Wahrscheinlichkeit, Statistik, Induktion

Christian Wurm
cwurm@phil.hhu.de

April 15, 2024

Contents

1 Zum Aufbau **10**

2 Induktion und Lernen – eine Begriffsklärung **12**

3 Wahrscheinlichkeiten - eine erste Intuition **13**
3.1 Wahrscheinlichkeit als eine Theorie rationalen Handelns 13
3.2 Wahrscheinlichkeit und Induktion 16
3.3 Bedeutung von Wahrscheinlichkeiten 17

4 Grundlagen der Wahrscheinlichkeitstheorie **18**
4.1 Desiderata 18
4.2 Boolesche Algebren 18
4.3 Einige Beobachtungen 19
4.4 Definition von Wahrscheinlichkeitsräumen 21
4.5 Ereignisse und Ergebnisse 21
4.6 Die Komplement-Regel 22
4.7 Die Summenregel 22
4.8 Die Produktregel 23
4.9 Das sog. Bayessche Gesetz und seine Bedeutung 24
4.10 Einige Beispiele von Wahrscheinlichkeitsräumen 25
4.10.1 Laplace-Räume 25
4.10.2 Bernoulli-Räume 25
4.10.3 Diskrete Wahrscheinlichkeitsräume 25
4.11 Produkträume 28

4.12 Unabhängige Ereignisse 29
4.13 Bedingte Wahrscheinlichkeit 31
4.14 Verbundwahrscheinlichkeiten und Marginalisierung 33
4.15 Wahrscheinlichkeitsgesetze – allgemeine Form 34

5 RX1 Erste Schritte in R **39**
5.1 Auf Daten zugreifen in R 39
5.2 Operationen auf dataframes 45
5.3 Korrelationen und Ähnliches 46
5.4 Rechnen mit dataframes 50

6 Zufallsvariablen **56**
6.1 Definition 56
6.2 Erwartungswert 57
6.3 Würfeln - mal wieder 58
6.4 Erwartungswerte bei Wetten – quantifizierte Gewißheit 60
6.5 Noch ein Beispiel: Erwartete Länge von Wörtern im Text . . . 62
6.6 Varianz und Standardabweichung 64
6.7 Varianz und Stichprobenvarianz 66
6.8 Kovarianz und Korrelation 68

7 RX2 Varianz etc. in R **71**
7.1 Variablen in Datensätzen 71
7.2 Verteilungen in Datensätzen 72

8 Wichtige Wahrscheinlichkeitsverteilungen **75**
8.1 Einleitung 75
8.2 n über k 77
8.3 Binomiale Verteilungen 78
8.4 Kategoriale Wahrscheinlichkeitsräume und Multinomiale Verteilungen 81
8.5 Normal-Verteilungen und der Zentrale Grenzwertsatz 83
8.6 Eine Anwendung des zentralen Grenzwertsatzes 85
8.7 Einige Illustrationen zu Binomial- und Normalverteilung . . . 88
8.8 Die geometrische Verteilung und die Exponentialverteilung . . 89
8.9 Die hypergeometrische Verteilung 91
8.10 Potenzgesetze 93
8.11 Zipfs Gesetz 95

8.12 Zipfs Gesetz und Wortlänge 97
8.13 Anmerkungen zu Zipf 98

9 R-Übung zu Verteilungen 99

10 RX3 Daten visualisieren 100

11 Hypothesen prüfen 108
11.1 Verteilungen und Vertrauensgrenzen in R 108
11.2 Sequentielle Überprüfung von Hypothesen 1 113
11.3 SÜH 2 – Unabhängig 116
11.4 p-Hacking 118

12 RX4a Übung 121

13 Sequentielle Bayesianische Hypothesenprüfung 123
13.1 Der Bayesianische Ansatz 123
13.2 Sequentielle Hypothesenprüfung – Bayesianisch 129
13.3 **RX5b** Die effektive Berechnung mit dynamischer Programmierung 132

14 Einseitige Tests 140
14.1 Ein zweites Beispiel – Fehlerquoten 140

15 RX6 Wahrscheinlichkeitsdichte, Wahrscheinlichkeitsmasse, Quantile 143

16 Mehrere Variablen und Tests für Unabhängigkeit 145
16.1 Vorspiel: Parameter schätzen 145
16.2 **RX7** Der Schwellentest 147
16.3 p-Werte und statistische Unabhängigkeit 153
16.4 Die χ^2-Verteilung und der χ^2-Test 156
16.5 χ^2 in R 160
16.6 t-test – ein Beispiel 161

17 Markov-Ketten 167
17.1 Vorgeplänkel 167
17.2 Markov-Ketten: Definition 169
17.3 (Teile der) Sprache als Markov-Prozess 171

17.4 Likelihood und Parameter-Schätzung bei für Markov-Ketten . 173

18 Wahrscheinlichkeiten schätzen 177
18.1 Die Likelihood-Funktion 177
18.2 Maximum Likelihood Schätzung I 179
18.3 Ein Beispiel 183
18.4 Definitionen 184
18.5 Hausaufgabe 10 189

19 Parameter glätten – Smoothing 1 (add one) 190
19.1 Add-one smoothing als ein bestimmes lineares Modell 192

20 Parameter glätten – Good-Turing smoothing (vereinfacht) 195

21 Parameter schätzen – Bayesianisch 199
21.1 Uniformes Apriori 199
21.2 Kein uniformes Apriori 202

22 Numerische Parameter und Alternativen zu ML 207

23 Parameter für offene Skalen schätzen 211
23.1 Einleitung 211
23.2 Apriori Verteilungen über diskrete offene Skalen 212
23.3 Schätzen von kontinuierlichen Skalenparametern 214
23.4 Jeffreys Apriori-Verteilung 215

24 Entropie, Kodierung, und Anwendungen 216
24.1 Definition 216
24.2 Kodierungstheorie und Entropie 220
24.3 Bedingte Entropie 225
24.4 Bedingte Entropie – ein Rechenbeispiel mit Anwendung 227
24.5 Kullback-Leibler-Divergenz 231
24.6 Kreuzentropie 232

25 Maximum Entropie Methoden 235
25.1 Definition 235
25.2 Ein einfaches Beispiel 237
25.3 ME in der Computerlinguistik 240

26 Klassifikation mittels Entropie: Entscheidungsbäume 242
26.1 (Boolesche) Entscheidungsfunktionen . . . 242
26.2 Entscheidungsbäume . . . 244
26.3 Overfitting I . . . 247
26.4 Overfitting II . . . 249
26.5 Implementierung eines Algorithmus . . . 251

27 Probabilistische Graphische Modelle I - Bayesianische Netze 256
27.1 Korrelation und Kausalität . . . 256
27.2 Einleitung 2: Entscheidungsfunktionen generalisieren . . . 261
27.3 Verdeckte Variablen/Rauschen . . . 263
27.4 Definitionen . . . 265
27.5 Rechnen mit BNs . . . 268
27.6 Konditionale (Un-)Abhängigkeit . . . 270
27.7 Minimalität und Direktionalität . . . 272
27.8 Von der Verteilung zum Graphen . . . 275
27.9 BN im Machine Learning . . . 278
27.10 BN implementieren (im Aufbau) . . . 279

28 Induktives Lernen 280
28.1 Der Rahmen . . . 280
28.2 Test und Trainigsdaten, Overfitting und Underfitting . . . 285

29 Lineare Regression 288
29.1 Der einfache lineare Fall . . . 288
29.2 Höhere Dimensionen . . . 290
29.3 Polynomiale Regression . . . 292
29.4 Probleme mit polynomialer Regression – Overfitting . . . 294
29.5 Polynomiale Regression in höheren Dimensionen . . . 295
29.6 Heuristische Optimierung: Gradientenbasierte Methoden . . . 296
29.7 Lernrate – flexibel . . . 298
29.8 Batch gradient descent . . . 299
29.9 Stochastic gradient descent . . . 300
29.10 Lineare Regression in R . . . 301

30 Regularisierung 303
30.1 Tikhonov Regularisierung/Ridge Regression . . . 304
30.2 Lasso Regression . . . 306

30.3 Frühes Aufhöhren 307

31 Logistische Regression – Aktivierungsfunktionen 308
31.1 Definitionen 308
31.2 Bedeutung 310
31.3 Kostenfunktion für LogReg 313

32 Softmax-Regression 314
32.1 Die Interpretation der logistischen Regression – ein Beispiel . . 319

33 Schema F der Regression/Klassifikation 322

34 Nearest neighbour Regression 325
34.1 Lineare Algebra (reine Wiederholung) 325
34.2 Die eigentliche Klassifikation (einfach) 327
34.3 Generalisierung: k nearest neighbour 328
34.4 Nearest Neighbour Regression und Logistische Regression . . . 329
34.5 Probleme und Perspektiven 330

35 Kurze Zusammenfassung für Regression und ML 331

36 Support Vector Machines für Klassifikation 332
36.1 Hyperebenen und lineare Separierung 332
36.2 Normalitätserwägungen 335
36.3 Das Problem der optimalen Separierung 336
36.4 Zwischenstück: Optimierung mit Lagrange Multiplikatoren . . 339
36.5 Regularisierung von SVM – Weiche Ränder 341

37 SVM: Merkmale und Kernel 343
37.1 Matrizen und lineare Abbildungen – eine Randbemerkung . . 344
37.2 Feature-maps 345
37.3 Der Kernel-Trick 346
37.4 Hinge-loss 350

38 Regression mit SVM 351

39 Wiederholung und Grundbegriffe 353

40 Metriken und Kostenfunktionen 354

41 k-means clustering **360**

42 "Curse of Dimensionality" **363**

43 Projektionen **365**

44 Mannigfaltigkeiten **366**

45 Locally Linear Embeddings **368**

46 Principal component analysis **371**
46.1 Kovarianz und Varianz 371
46.2 PCA mittels Eigenzerlegung, SWZ 375
46.3 Geometrische Darstellung 377
46.4 Algebraische Darstellung und Kodierung 379
46.5 Dimensionen reduzieren mit PCA – Varianz 383
46.6 PCA und Klassifikation/Regression 386
46.7 Kernel PCA 387

47 PAC-Lernen **388**
47.1 Einleitung 388
47.2 Definitionen 389
47.3 PAC-lernbare Probleme I 395
47.4 PAC-lernbare Probleme II 397
47.5 PAC-lernbare Probleme III 398

48 EM-Algorithmen: Parameter schätzen von unvollständigen Daten **400**
48.1 Einleitung 400
48.2 Ein Beispielproblem 401
48.3 Der EM-Algorithmus auf unserem Beispiel 403
48.4 Der Algorithmus (allgemeine Form) 406

49 Der EM-Algorithmus in der maschinellen Übersetzung **408**
49.1 Grundbegriffe der maschinellen Übersetzung 408
49.2 Wahrscheinlichkeiten schätzen 411
49.3 Der EM-Algorithmus: Vorgeplänkel 413
49.4 Der eigentliche Algorithmus 416
49.5 EM für IBM-Modell 1: Ein Beispiel 417

50 Naive Bayes Klassifikatoren (aka *idiot Bayes)* 420
50.1 Der naive Bayes Spamfilter 420
50.2 BN 420

51 Bayesian Learning 420

52 Monte Carlo Methoden 421
52.1 Gibbs-Sampling 421
52.2 MCMC 421

53 Hidden Markov Modelle 421

54 Zur Methodik des maschinellen Lernens 421
54.1 Abriß der Methode 421
54.2 Zwei Probleme 422

55 *No free lunch* – Gibt nix umsonst 424
55.1 Einleitung 424
55.2 Die NFL Theoreme und was sie bedeuten 425
55.3 Einige NFL Theoreme 431
55.4 Was bedeutet das also... 435
55.5 NFL und Probabilistische Optimierung 436
(Zur RX-Notation: damit sind alle Kapitel, die für den R-Teil relevant sind, markiert; RX damit es leicht zu finden ist mit Suchfunktion.)

Kursinhalte und Quellen

Das Ziel soll es sein, Methoden der Wahrscheinlichkeitstheorie, Statistik und des maschinellen Lernens zu verstehen. Insgesamt kann man das Thema umschreiben mit der Frage: wie können wir sinnvolle Schlüsse mit unsicherer und ungenügender Information ziehen? Was wichtig ist: es geht mir nicht um die einfache Anwendung fertiger Methoden (was oft genug sinnlos ist), sondern um Verständnis. Das hat natürlich Vor- und Nachteile, macht die Sache aber insgesamt nicht leichter.

Dieses Skript orientiert sich in Sachen Wahrscheinlichkeitstheorie in weiten Teilen am Skript von Marcus Kracht (zu finden online unter http://wwwhomes.uni-bielefeld.des/mkracht/html/statistics.pdf); das ist für

diejenigen, die es ganz genau wissen wollen. Dort finden sich ausführlichere Definitionen und Beweise, die ich hier meist auslasse.

Weiterhin benutze ich Edwin Jaynes' "Probability Theory: The Logic of Science". Jaynes war Physiker und einer der wichtigsten Wegbereiter der Bayesianischen Statistik.

Für den Teil um das maschinelle Lernen benutze hauptsächlich ich von Stuart Russell & Peter Norvig "Artificial Intelligence: A Modern Approach", ein Buch das gleichermaßen breit, gut informiert, gründlich wie leicht zugänglich ist, das also nur empfohlen werden kann.

Ein sehr neues Buch das ich benutze und empfehlen kann ist "Deep Learning" von Bengio et al., hier v.a. die ersten Kapitel. Hier werden einige Dinge sehr präzise umrissen; das meiste spielt aber hier erstmal keine Rolle.

1 Zum Aufbau

Das Skript zerfällt in zwei große Teile, und wird benutzt in zwei verschiedenen Kursen:

I Wahrscheinlichkeitstheorie (Abschnitt 3–27), Implementierung in R

II Machine learning (Abschnitt 28–47), Implementierung in Python

Teil I kann man wiederum aufteilen in verschiedene große Themengruppen, die verschiedene Kapitel umfassen:

I.1 Wahrscheinlichkeitstheorie (die reine Lehre) 3–10

I.2 Hypothesen und Tests (Plausibilität von Theorien, also die Umkehrung der reinen Lehre) 11–16

I.3 Parameter schätzen (im Prinzip ist das wie 1.2, nur mit unendlich vielen Hypothesen) (hier zugeschlagen Markov Ketten) (17–23)

I.4 Entropie und Information, Entscheidungsbäume als Anwendung (24–26)

I.5 Bayesianische Netze, Kausalität, konditionale Unabhängigkeit (hier gehören die Markov-Ketten eigentlich hin) 27

I.6 Probabilistische Sprachen und Grammatiken. Das ist im Aufbau. (also keine Kapitel!)

Für meinen Kurs Statistik und Wahrscheinlichkeit behandele ich normalerweise I.1–I.5, aber bei weitem nicht alles was unter diese Punkte fällt (insbesondere I.2, I.3 wird sehr selektiv behandelt).

Teil II Das kann man ebenso aufteilen. Dieser Teil beschäftigt sich fast ausschließlich mit Vektorbasierten Modellen, und hier fast nur mit Modellen die im weiteste Sinne linear sind.

II.1 Grundlagen und Methoden des maschinellen Lernens aka. Induktion 28

II.2 Lineare Regression und was darauf aufbaut (29–32)

II.3 Support Vector Machines (36–38)

II.4 Hauptkomponentenanalyse/PCA (46)

II.5 Weitere Vektormodelle: Clustering, Locally Linear Embeddings, Nearest Neightbour Regression 34, 41, 45

II.6 Modelle, die nicht direkt mit Vektoren und Optimierung zu tun haben. PAC-Lernen, EM-Algorithmen (47–49, weiteres in Arbeit.

Ich behandele im Machine Learning-Kurs normalerweise II.1-II.5, mehr oder weniger wie sie da stehen.

2 Induktion und Lernen – eine Begriffsklärung

In der Literatur ist oft etwas undifferenziert von Lernen und Induzieren die Rede. Dabei gibt es eine klare und sinnvolle Unterscheidung:
"**Lernen**" bedeutet: wir wissen, was gelernt werden soll, und uns interessiert, wie und ob jemand, der das Ziel nicht kennt, dorthin gelangt. In diesem Sinne kann man z.B. sagen: die Kinder haben Arithmetik gelernt, die Studenten haben Algebra gelernt etc.
"**Induktion**" bedeutet: wir möchten eine allgemeine Regel/System erstellen, die normalerweise für eine unendliche Menge von Beobachtungen gilt, für die wir aber nur eine endliche Menge von Beobachtungen haben. Der entscheidende Punkt ist: wir kennen nicht die korrekte Regel, wir wissen nur dass es eine gibt. Was immer wir am Ende haben, ist möglicherweise falsch. Beispiele sind:

- Die Wahrscheinlichkeit eines gewissen Satzes in einer gewissen Sprache (woher sollen wir das wissen)
- Die Theorie der Schwerkraft (kann ja immer noch falsch sein)
- Eine Grammatik (für eine unendliche Sprache) gegeben eine endliche Menge von Sätzen die wir beobachten

Die Beispiele zeigen schon: für praktische und wissenschaftliche Anwendungen ist der Begriff der Induktion tatsächlich viel interessanter und relevanter als der des Lernens. Der Begriff der Induktion ist eng mit dem der Wahrscheinlichkeit verknüpft, insbesondere in der Praxis. Deswegen werden wir uns zunächst damit beschäftigen.

3 Wahrscheinlichkeiten - eine erste Intuition

3.1 Wahrscheinlichkeit als eine Theorie rationalen Handelns

Praktisch alles, was wir in diesem Seminar machen, basiert auf Wahrscheinlichkeitstheorie. Deswegen ist es wichtig, dass wir eine gute Intuition dafür haben, was Wahrscheinlicheit bedeutet. Die Wahrscheinlichkeit ist erstmal ein Maß dafür, wie sicher/unsicher wir sind, dass ein Ereignis eintritt. Dabei bezeichnet man mit 1 die Sicherheit, dass es eintritt, mit 0 die Sicherheit, dass es nicht eintritt; Wahrscheinlichkeiten sind also Zahlen in [0,1]. Wir schreiben P(A) für die Wahrscheinlichkeit von A, wobei A für ein beliebiges Ereignis steht. Nehmen wir nun 2 Ereignisse A,B; nehmen wir weiterhin an, $P(A) > P(B)$. Dann bedeutet das soviel wie: wir gehen davon aus, A eher eintritt als B. Das hat eine sehr natürliche Interpretation, wenn wir z.B. von Risiken und Rationalität sprechen: nehmen wir an

A=Ein Fahrradfahrer stirbt in einem Unfall, weil er keinen Helm aufhat.

B=Ein Fahrradfahrer stirbt in einem Unfall, weil er den Radweg gegen die Fahrtrichtung fährt.

Nehmen wir weiterhin an, $P(A) < P(B)$ (das lässt sich mit Statistiken verifizieren). In diesem Fall würden wir sagen, ein Radfahrer, der mit Helm den Radweg gegen die Fahrtrichtung fährt, ist irrational (aber nicht unbedingt, wenn er ohne Helm fährt). Im Zusammenhang mit Risiken gilt also: Wahrscheinlichkeiten haben viel mit rationalem handeln zu tun, und in gewissem Sinne ist die Wahrscheinlichkeitstheorie so etwas wie eine **Theorie des rationalen Handelns**.

Um dieses Beispiel Konzept weiter zu klären, machen wir ein etwas komplexeres Beispiel. Sie fragen sich, ob Sie einen Fahrradhelm aufziehen sollen. Das macht per se noch keinen Unterschied, sondern erst im Falle eines Unfalls. Aber selbst im Falle eines Unfalls macht der Helm nicht zwangslufig einen Unterschied, sondern kann aus zweierlei Grnden Unütz sein:

1. Der Kopf ist nicht (genug) involviert
2. Der Kopf ist zuviel involviert (z.B. ein LKW fährt darüber)

Die Frage, ob mir der Fahrradhelm wirklich nutzt, ist also gar nicht so leicht zu beziffern! Wie geht das? Nehmen wir erstmal die Unfallwahrscheinlichkeit

$$P(F) = x$$

Wir suchen also $P(F)$, und diese Größe ist unbekannt. Sie *schätzen* diese Größe aber auf eine gewisse Art und Weise. *Schätzen* ist hier bereits ein technischer Begriff, und wir nennen bezeichnen die geschätzte Wahrscheinlichkeit mit

$$\hat{R}(F) = \hat{x}.$$

Geschätzte Wahrscheinlichkeiten haben also einen Hut auf. Man kann diese Größe schätzen z.B. indem man die Anzahl der zurckgelegten Fahrradkilometer teilt durch die Anzahl der Unfälle (vereinfacht gesagt).

Als nächstes braucht man die Wahrscheinlichkeit, dass, **gegeben einen Unfall**, der Helm mich rettet:

(1) $P(H|U) = y$

Auch diese Wahrscheinlichkeit kann man schätzen aus Unfallstatistiken mittels relativen Häufigkeiten.

Aber wie ist nun die Wahrscheinlichkeit, dass der Helm mich rettet? Gegeben x, y können wir das nun ausrechnen mit dem Wahrscheinlichkeitskalkül!

Wir haben gesehen, dass Wahrscheinlichkeiten mit Häufigkeiten zusammenhängen. Manchmal wissen wir aber nichts über Häufigkeiten. Allgemeiner gesagt: wir haben keine relevante Information um die Wahrscheinlichkeit eines Ereignisses zu schätzen. Das gilt z.B. für einen Wurf mit einer (uns unbekannten Münze. Was ist die Wahrscheinlichkeit, dass Sie auf den Kopf fällt? Oder wie sollen wir die Wahrscheinlichkeit schätzen, bei einem Unfall auf den Kopf zu fallen wenn wir keine Zahlen dazu kennen?

Hier nutzen wir ein wichtiges Prinzip:

$$\hat{P}(E) = 0.5$$

Das ist das **Prinzip der Indifferenz**: falls wir keinerlei Information haben ob ein Ereignis E eintritt oder nicht, dann schätzen wir $\hat{P}(E) = 0.5$. Dieses Prinzip muss man oft noch leicht generalisieren (dann wird Formulierung etwas abstrakter):

Prinzip der Indifferenz Sei **E** ein Zufallsexperiment mit n möglichen Ergebnissen, $E_1, ..., E_n$. Wenn wir keinerlei relevante Information haben über **E**, dann gilt für all $i : 1 \leq i \leq n$: $\hat{P}(E_i) = \frac{1}{n}$ (man nehme einen handelsüblichen Würfel, dann haben wir $n = 6$, und das Zufallsexperiment ist ein Wurf).

Noch eine weitere Sache kann man hier sehen: gegeben ein Ereignis E bezeichnen wir mit $\overline{E}$ sein **Komplement**, also die Tatsache dass es (im Rahmen des Zufallsexperimentes) *nicht* stattfindet. Mit E_1E_2 bezeichnen wir kurzerhand die Tatsache, dass zwei Ereignisse E_1 und E_2 stattfinden. Und hier kommt die große Aufgabe der Wahrscheinlichkeitstheorie:

Die Aufgabe Die logischen Operationen von Konjunktion ("und"), Negation ("nicht") etc. müssen wir transformieren in **numerische Operationen** auf Wahrscheinlichkeiten. Denn am Ende wollen wir *eine* Zahl haben, die unser geschätztes Risiko wiedergibt.

Genau diese Rechenregeln werden wir als nächstes besprechen. Bei diesen Regeln geht es darum, logische Verknüpfungen von Operationen umzuwandeln in numerische Operationen, anhand derer wir das Risiko quantifizieren können.

3.2 Wahrscheinlichkeit und Induktion

Hier haben wir Wahrscheinlichkeiten beschrieben als ein Mittel, um uns rational zu verhalten. Im Zusammenhang mit Induktion suchen wir etwas anderes, aber sehr ähnliches: nicht die rationalste Verhaltensweise, sondern die rationalste Theorie über die Natur der Dinge. Wir suchen also eine rationale Sicht der Dinge. Das ist in der Tat für uns die geläufigste Anwendung für Wahrscheinlichkeiten: sie sollen uns sagen:

Frage der Induktion Gegeben eine Reihe Beobachtungen, die wir gemacht haben, was ist die plausibelste Theorie der zugrundeliegenden Gesamtheit/Realität?

Mit *plausibel* wird üblicherweise gemeint: hat die höchste Wahrscheinlichkeit. Hierbei spielen normalerweise 2 Faktoren eine Rolle:

1. Wie plausibel sind unsere Beobachtungen unter der Annahme, dass die Theorie richtig ist?

2. Wie plausibel ist unsere Theorie in sich?

Denn es kann sein, dass unsere Beobachtungen sehr wahrscheinlich sind unter der Annahme, dass ein Troll sie mit einer gewissen Absicht generiert; aber diese Theorie ist in sich sehr unwahrscheinlich.

Mit "zugrundeliegender Realität" meinen wir meistens eine Wahrscheinlichkeitsverteilung oder eine zugrundeliegende Gesamtheit (Population), von der wir nur einzelne Stichproben beobachten können. Z.B. eine Fabrik stellt Fernseher her; wir prüfen davon eine Auswahl, z.B. 1000 Stück. Dann möchten wir wissen, wie viele der insgesamt produzierten Fernseher (Population) defekt sind, bzw. wie die Wahrscheinlichkeit ist, dass ein beliebiger produzierter Fernseher defekt ist (Wahrscheinlichkeitsverteilung).

Weiterhin möchten wir oft wissen: mit welcher Sicherheit können wir diesen Schluss ziehen? Natürlich ist jede Annahme dieser Art sicherer, je mehr Fernseher wir prüfen. Mit diesen Themen befasst sich **statistische Inferenz**, und wir werden oft Probleme dieser Art treffen.

3.3 Bedeutung von Wahrscheinlichkeiten

Eine Sache, die man gleich zu Anfang klären sollte, ist: *was bedeuten eigentlich Wahrscheinlichkeiten*? Üblicherweise ist man versucht zu sagen: wenn eine Münze mit einer Wahrscheinlichkeit von 1/2 auf Kopf fällt, dann heißt das, dass sie perfekt symmetrisch ist, und weiterhin: wenn wir sie oft genug werfen, wird sie in ca. der Hälfte der Fälle auf Kopf landen. Die Wahrscheinlichkeit beschreibt also eine physische Eigenschaft und in der Folge ein Verhalten.

Das klingt gut, ist aber problematisch: was ist die Wahrscheinlichkeit, dass es Leben auf dem Mars gibt? Und gegeben dass wir eine Münze finden und werfen, was ist die Wahrscheinlichkeit, dass sie auf Kopf landet? Hier haben wir einen anderen Begriff von Wahrscheinlichkeit: er drückt die Stärke unserer Überzeugung aus. Diese kann – im Falle des Mars – mehr oder weniger informiert sein. Im Falle der Münze sagen wir: die Wahrscheinlichkeit, dass sie auf Kopf landet, ist 1/2, denn wir haben keinerlei wissen, dass uns dahin bringen würde, Kopf oder Zahl vorzuziehen. Diese uniforme Verteilung ist also **Ausdruck unserer Ignoranz**. Darauf beruht das Prinzip der Indifferenz.

4 Grundlagen der Wahrscheinlichkeitstheorie

4.1 Desiderata

Wir haben gesehen dass wir für die Wahrscheinlichkeitstheorie 2 große Desiderata haben:

1. Wir wollen (aussagen)logische Operationen für Ereignisse; und
2. wir möchten die logischen Operationen *numerisch interpretieren*, d.h. in numerische Funktionen verwandeln.

4.2 Boolesche Algebren

Logische Operationen können wir in Booleschen Algebren interpretieren:

Definition 1 *Sei M eine Menge. Ein Mengensystem $\mathcal{M} \subseteq \wp(M)$ ist eine Boolesche Algebra über M, falls*

1. *$M \in \mathcal{M}, \emptyset \in \mathcal{M}$;*
2. *falls $N \in \mathcal{M}$, dann ist auch $\overline{N} := M - N \in \mathcal{M}$;*
3. *falls $N_1, N_2 \in \mathcal{M}$, dann sind auch $N_1 \cup N_2 \in \mathcal{M}$.*

NB: die Definition impliziert dass falls $N_1, N_2 \in \mathcal{M}$, dann ist auch $N_1 \cap N_2 \in \mathcal{M}$, da $N_1 \cap N_2 = \overline{\overline{N_1} \cup \overline{N_2}}$. Unsere Definition betrifft eigentlich nur einen Spezialfall von Booleschen Algebren, nämlich solchen über Mengensystemen. Allerdings kann jede endliche Boolesche Algebra auf diesen Spezialfall reduziert werden.

Übung: Mengenlehre und Partitionen

- $M \cap N = \overline{\overline{M} \cup \overline{N}}$ (Interdefinierbarkeit 1)
- $M \cup N = \overline{\overline{M} \cap \overline{N}}$ (Interdefinierbarkeit 2)
- $\overline{\overline{M}} = M$ (doppeltes Komplement)
- $(M \cup N) \cap O = (M \cap O) \cup (N \cap O)$ (Distribution)

Eine **Partition** einer Menge M ist eine Menge $X \subseteq \wp(M)$, d.h. $X = \{N_1, ..., N_i\}$ (im endlichen Fall), und es gilt:

1. $N_1 \cup ... \cup N_i = M$
2. für alle $N_i, N_j \in X$, entweder $N_i = N_j$ oder $N_i \cap N_j = \emptyset$.

4.3 Einige Beobachtungen

Wir haben bereits gesagt, dass Wahrscheinlichkeiten Zahlen in $[0, 1]$ sind, wobei wir die Korrespondenz haben

$0 \cong$ Unmöglichkeit
$1 \cong$ Sicherheit

Nun haben wir, aus logischen Gründen folgendes:

(1) $P(A) \leq P(A \text{ oder } B)$ und $P(B) \leq P(A \text{ oder } B)$

(In Zukunft schreiben wir: $P(A \cup B)$). Das ist klar: wann immer A eintritt, tritt auch A oder B ein, also ist das Ereignis wahrscheinlicher etc. Ebenso klar ist:

(2) $P(A \text{ und } B) \leq P(A)$ und $P(A \text{ und } B) \leq P(B)$

(In Zukunft schreiben wir: $P(A \cap B)$ oder einfach $P(AB)$). Das ist klar: die Wahrscheinlichkeit, dass sie bei Ihrer nächsten Radfahrt angefahren werden ist größer als die, dass sie angefahren werden und im Zuge dessen 50euro finden.

Gleichzeitig haben wir folgendes: sei $\bot$ ein Ereignis, das vollkommen unmöglich ist, z.B. Sie würfeln (mit einem handelsüblichen Würfel) eine 7. Dann haben wir natürlich:

(3) $P(A \cap \bot) = 0$; $P(A \cup \bot) = P(A)$

Also, in Worten: $\bot$ ist *absorbierend für Konjunktion* und *neutral für Disjunktion*.

Umgekehrt, sei $\top$ ein Ereignis, dessen Eintritt sicher ist, z.B. dass Sie eine Zahl zwischen 1 und 6 würfeln. Dann haben wir

(4) $P(A \cap \top) = P(A)$; $P(A \cup \top) = 1$

Also gilt: $\top$ ist absorbierend für Disjunktion, und neutral für Konjunktion. Nun haben wir, nach Annahme:

(5) $\quad P(\top) = 1;\ P(\bot) = 0$

Wir suchen also Operationen, für die 1, 0 jeweils neutral bzw. absorbierend sind. Das wird erfüllt von den Operationen + und $\cdot$:

(6) $\quad n + 0 = n \qquad n \cdot 0 = 0$

Ebenso haben wir:

(7) $\quad n \cdot m \leq n \qquad$ und $\qquad n \cdot m \leq m \qquad$, für $n, m \in [0, 1]$,

sowie:

(8) $\quad n \cdot 1 = n \qquad n + 1 \geq 1$

sowie:

(9) $\quad n \leq n + m$ und $m \leq n + m$, für $n, m \in [0, 1]$

Wir haben also folgende Korrespondenz:

Konjunktion $\cong \cdot$
Disjunktion $\cong +$

Das Problem ist, dass sich in dem einfachen Fall die Wahrscheinlichkeiten *nicht* auf 1 aufsummieren. Wir haben eine Korrespondenz, aber das ist noch zu einfach gedacht. Das sieht man auch an folgendem Beispiel:

(10) $\quad P(A \cap A) = P(A) \neq P(A) \cdot P(A)$

sowie

(11) $\quad P(A \cup A) = P(A) \neq P(A) + P(A)$

Konjunktion und Disjunktion sind also **idempotent**, im Gegensatz zur Addition und Multiplikation. Die Materie ist also durchaus komplex; es gibt allerdings eine wunderbar elegante Lösung, die uns mit allen nötigen Rechenregeln versorgt.

4.4 Definition von Wahrscheinlichkeitsräumen

Folgende Definition stammt von Kolmogorov, und ist das Ergebnis langer Überlegungen und Dispute.

Definition 2 *Ein Wahrscheinlichkeitsraum ist ein Tripel* $(\Omega, \mathfrak{A}, P)$*, wobei* $\mathfrak{A} \subseteq \wp(\Omega)$ *eine Boolesche Algebra ist, und* $P : \mathfrak{A} \to [0, 1]$ *eine Wahrscheinlichkeitsfunktion, so dass*

1. $P(\Omega) = 1$*;*
2. $P(\emptyset) = 0$*, und*
3. *falls* $A_1, A_2, ..., A_n$ *paarweise disjunkt sind, dann ist*

$$P(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} P(A_i)$$

Zur Erklärung: mit **paarweise disjunkt** meint man: für alle i, j so dass $1 \leq i, j \leq n$, falls $i \neq j$, dann ist $A_i \cap A_j = \emptyset$.

Die Bedingung der Booleschen Algebra ist wie folgt zu verstehen: falls $A, B \subseteq \Omega$ Ereignisse sind, die eine Wahrscheinlichkeit haben, dann haben auch die Ereignisse $A \cup B$ (d.h.: A oder B trifft ein), $A \cap B$ (d.h. beide A und B treffen ein) und $\overline{A}$ (d.h. A trifft nicht ein) eine Wahrscheinlichkeit.

4.5 Ereignisse und Ergebnisse

Wir nennen eine Menge $A \subseteq \Omega$ ein **Ereignis**; wir nennen $a \in \Omega$ ein **Ergebnis**. Meistens entspricht ein Ergebnis a einem Ereignis $\{a\}$. Aber nicht immer ist das intuitiv: nehmen wir an, wir würfeln mit zwei Würfeln, wobei unsere Ergebnisse die Form haben

$$\langle m, n \rangle$$

Nun ist "der erste Wurf ist eine 2" kein Ergebnis, sondern ein Ereignis, nämlich das Ereignis

$$\{\langle 2, 1 \rangle, ..., \langle 2, 6 \rangle\}$$

Daher weisen wir Wahrscheinlichkeiten normalerweise Ereignissen zu, nicht Ergebnissen.

4.6 Die Komplement-Regel

Wir kommen nun zu den Rechenregeln. Die Regel für die Berechnung des Komplementes $P(\overline{A})$ aus $P(A)$ lautet wie folgt:

(2) $P(\overline{A}) = 1 - P(A)$

Das lässt sich sehr einfach ableiten: wir haben

1. $P(A \cup \overline{A}) = P(\Omega) = 1$ und
2. $A \cap \overline{A} = \emptyset$;

also:

$$\begin{aligned} 1 &= P(A \cup \overline{A}) \\ &= P(A) + P(\overline{A}) \\ \Leftrightarrow 1 - P(A) &= P(\overline{A}) \end{aligned}$$

4.7 Die Summenregel

Die Summenregel erlaubt es uns, die logische Disjunktion rechnerisch aufzulösen. Die Summenregel lautet:

(3) $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

Intuitiv bedeutet das: um die Wahrscheinlichkeit einer Disjunktion zu berechnen, reicht es die Wahrscheinlichkeiten zu addieren, wenn man nur die Wahrscheinlichkeitsmasse abzieht, die auf die Konjunktion beider Ereignisse entfällt (illustrierbar mittels Venn-Diagramm). Das lässt sich wie folgt ableiten aus den Axiomen:

$$\begin{aligned} P(A \cup B) &= P(A \cup (B \cap \overline{A})) && \text{(Mengenlehre)} \\ &= P(A) + P(B \cap \overline{A}) && \text{(Disjunkte Mengen)} \\ &= P(A) + P(B \cap (\overline{A} \cup \overline{B})) && \text{(Mengenlehre)} \\ &= P(A) + P(\overline{\overline{B \cap (\overline{A \cap B})}}) && \text{(Mengenlehre)} \\ &= P(A) + P(\overline{\overline{B} \cup (A \cap B)}) && \text{(Mengenlehre)} \\ &= P(A) + (1 - P(\overline{B} \cup (A \cap B))) && \text{(Subtraktionsregel)} \end{aligned}$$

$$
\begin{aligned}
&= P(A) + (1 - P(\overline{B}) + P(A \cap B)) && \text{(Disjunkte Mengen)} \\
&= P(A) + (1 - (1 - P(B)) + P(A \cap B)) && \text{(Disjunkte Mengen)} \\
&= P(A) + (1 - (1 - P(B))) - P(A \cap B)) && \text{(Arithmetik)} \\
&= P(A) + (1 - 1) + P(B) - P(A \cap B)) && \text{(Arithmetik)} \\
&= P(A) + P(B) - P(A \cap B)) && \text{(Arithmetik)}
\end{aligned}
$$

4.8 Die Produktregel

Um die Konjunktion sinnvoll zu interpretieren, brauchen wir die Definition der *bedingten Wahrscheinlichkeit*. Wir definieren

$$(4) \quad P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Nehmen Sie nun an wir suchen die Wahrscheinlichkeit $P(A \cap B)$; wir bekommen sie durch eine ganz simple Termumformung:

$$(5) \quad P(A \cap B) = P(A|B)P(B)$$

Da $P(A \cap B) = P(B \cap A)$ ($\cap$ ist kommutativ), bekommen wir also die Produktregel:

$$(6) \quad P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

Intuitiv ist das wie folgt zu verstehen: wenn A und B eintreffen, dann bedeutet dass das 1. A eintrifft und 2. unter dieser Voraussetzung B eintrifft (oder umgekehrt). Wichtig ist: $P(A|B)$ sagt nichts über zeitliche Reihenfolge! So ist die Formel intuitiv richtig. Wir werden später noch mehr zum Konzept der bedingten Wahrscheinlichkeit erfahren.

Diese Umformung mag einem auf Anhieb nicht sehr hilfreich erscheinen, allerdings ist sie eines der zentralen Gesetze. Denn in der Praxis kennen wir oft bedingte Wahrscheinlichkeiten besser als unbedingte, so dass wir uns das Gesetz leicht zunutze machen können. Es gibt übrigens noch eine allgemeinere Form der Produktregel:

$$(7) \quad P(A \cap B|X) = P(A|BX)P(B|X) = P(B|AX)P(A|X)$$

Das generalisiert die letzte Formel, da X beliebig (auch leer) sein kann.

4.9 Das sog. Bayessche Gesetz und seine Bedeutung

Das Bayessche Gesetz bzw. Theorem ist im Prinzip auch nichts anderes als eine Term-Umformung, vorausgesetzt alle Definitionen soweit. Es sieht wie folgt aus:

(8)
$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A \cap B)}{P(B)} \frac{P(A)}{P(A)} = \frac{P(B \cap A)}{P(A)} \frac{P(A)}{P(B)} = P(B|A) \frac{P(A)}{P(B)}$$

Die Bedeutung ist folgende: wir haben Wahrscheinlichkeitstheorie eingeführt als ein Werkzeug, um uns rational zu verhalten. Noch häufiger werden wir sie benutzen, um eine rationale Sicht der Dinge zu bekommen. Die Frage wird sein: gegeben unsere Beobachtungen, was ist die wahrscheinlichste Annahme über die Natur der Dinge? Seien

- B unsere Beobachtungen,
- H eine Hypothese (über zugrundeliegende Wahrscheinlichkeiten);
- wir kriegen immer $P(B|H)$
- wir suchen aber $P(H|B)$!

Das lässt sich aber nicht ohne weiteres errechnen; wir bekommen normalerweise nur $P(B|H)$! Das Bayessche Gesetz erlaubt uns aber, von $P(B|H)$ zu $P(H|B)$ zu gelangen – mit einen Seitenannahmen, doch dazu später mehr.

4.10 Einige Beispiele von Wahrscheinlichkeitsräumen

4.10.1 Laplace-Räume

In einem Laplace-Raum gilt folgendes: wir haben $\mathfrak{A} = \wp(\Omega)$, das heißt zunächst, jedes mögliche Ereignis bekommt eine Wahrsccheinlichkeit. Außerdem haben wir, für alle $A \in \wp(\Omega)$,

$$P(A) = |A|/|\Omega| \tag{9}$$

Das bedeutet soviel wie: alle *Ergebnisse*, also alle "atomaren" Ereignisse, sind gleich wahrscheinlich. Das beste Beispiel für einen Laplace Raum ist ein fairer Würfel mit n Zahlen (n ist beliebig, muss aber endlich sein!). Natürlich bedeutet das nicht, dass alle Ereignisse gleich wahrscheinlich sind, denn wenn wir einen handelsüblichen Würfel mit 6 Zahlen nehmen, dann ist das Ereignis $\{2, 4, 6\}$ eines geraden Ergebnisses natürlich wahrscheinlicher als das Ereignis $\{2\}$ dass wir eine 2 werfen.

4.10.2 Bernoulli-Räume

Ein Bernoulli Raum hat nur zwei Ergebnisse: wir haben $\Omega = \{1, 0\}$, außerdem haben wir wie vorher: $\mathfrak{A} = \wp(\Omega)$, und $P(1) = 1 - P(0)$. Das typische Beispiel für einen Bernoulli-Raum ist der Wurf einer Münze, die möglicherweise auch unfair ist.

4.10.3 Diskrete Wahrscheinlichkeitsräume

Diskrete Wahrscheinlichkeitsräume sind eine Generalisierung von Laplace und Bernoulli-Räumen. Ein Wahrscheinlichkeitsraum ist diskret, falls

$$\mathfrak{A} = \wp(\Omega),$$

also wenn jedes denkbare Ereignis eine Wahrscheinlichkeit hat.

Ein wichtiges Ergebnis ist das folgende (das wir hier nur informell erklären): jeder endliche Wahrscheinlichkeitsraum kann als ein diskreter Raum "aufgefasst werden". Mit der Wendung "aufgefasst werden" meinen wir soviel wie: kann darauf abgebildet werden, ohne dass wir irgendwelche Information verlieren.

Nicht diskrete Wahrscheinlichkeitsräume sind v.a. im Unendlichen interessant. Man nehme z.B. einen Wahrscheinlichkeitsraum über $\Omega = \mathbb{N}$. Wir

nehmen an, dass dieser Wahrscheinlichkeitsraum fair ist, das bedeutet: alle $n \in \mathbb{N}$ sind gleich wahrscheinlich. Das bedeutet aber:

$$(10) \quad P(n) = \frac{1}{|\mathbb{N}|} = 0$$

Die Wahrscheinlichkeit einer Zahl muss also 0 sein - sobald sie grösser ist, würden sich Wahrscheinlichkeiten auf ∞ addieren. Wie sind jetzt die Wahrscheinlichkeiten von Ereignissen definiert, z.B. der geraden Zahlen? Hier gibt es verschiedene Möglichkeiten; wir nehmen folgende: für $M \subseteq \mathbb{N}$ gilt

$$(11) \quad P(M) = lim_{n\to\infty} \frac{|M \cap \{1, ..., n\}|}{n}$$

Damit bekommen wir z.B.

$$\begin{aligned} P(\{2n : n \in \mathbb{N}\} &= \frac{1}{2} \\ P(\{3n : n \in \mathbb{N}\} &= \frac{1}{3} \\ &... \end{aligned}$$

Weiterhin: falls $|M| < \omega$, dann ist $P(M) = 0$; sprich: jede endliche Menge hat die Wahrscheinlichkeit 0. Das Gegenstück ist aber nicht der Fall: nicht jede unendliche Menge hat eine Wahrscheinlichkeit, denn: der Grenzwert ist nicht in jedem Falle definiert (es gibt viele Mengen, die haben keinen Grenzwert, weil der Bruch nicht konvergiert).

Dieser Wahrscheinlichkeitsraum ist noch aus einem anderen Grunde interessant: wir haben

$$(12) \quad P(1) = P(3) = P(5) = P(7) = ... = 0$$

aber:

$$(13) \quad P(\{1, 3, 5, 7, ...\}) = 0.5$$

Daraus folgt:

$$(14) \quad P(\bigcup_{i=1}^{\infty} \{2i - 1\}) \neq \sum_{i=1}^{\infty} P(\{2i - 1\})$$

Das ist kein Widerspruch zu Axiom 3 in Definition 2 (Wahrscheinlichkeitsraum), denn in Def2 geht es nur um endliche Vereinigungen/Summen. Dennoch ist es schön, wenn diese Eigenschaft auch im Unendlichen gilt; man nennt das σ-Additivität. Unser Raum ist also nicht σ-additiv.

Gibt es einen diskreten, σ-additiven Raum über $\mathbb{N}$? Erstmal natürlich gibt es relativ triviale Räume mit Wahrscheinlichkeitsfunktionen wie

$$P(M) = \begin{cases} 1, \text{ falls } 1 \in M \\ 0 \text{ andernfalls} \end{cases} \tag{15}$$

Ein interessanterer Raum ist der, in dem *jede* Zahl eine streng positive Wahrscheinlichkeit bekommt. Wie ist das möglich? Z.B. mit folgender Funktion:

$$P_2(n) = \frac{1}{2^n} \tag{16}$$

Dieser Raum ist diskret und σ-additiv. Solche Räume sind insbesondere in der Computerlinguistik interessant: jedes Sprachmodell (= Wahrscheinlichkeitsverteilung über einer Sprache wie Deutsch, Englisch etc.) ist ein diskreter, σ-additiver Wahrscheinlichkeitsraum über Σ^* (einer unendlichen Menge).

4.11 Produkträume

Produkträume sind eine intuitiv sehr einfache Erweiterung von Wahrscheinlichkeitsräumen. Nehmen wir an wir haben einen Würfel und kennen seine Wahrscheinlichkeiten. Wir möchten jetzt aber Wahrscheinlichkeiten wissen dafür, dass wir mit demselben Würfel zweimal in Folge eine gewisse Zahl Würfeln; uns interessiert also beispielsweise das Ereignis $\langle 2, 3\rangle$ (die spitzen Klammern stehen hier für geordnete Paare, also hier für das Ereignis: erster Wurf 2, zweiter Wurf 3). Das Ereignis $\{\langle 2, 3\rangle\}$ ist allerdings *kein* Element unseres Wahrscheinlichkeitsraums. Wie geben wir ihm dennoch eine Wahrscheinlichkeit?

Hier hilft uns das Produkt zweier Räume, oder, in diesem konkreten Fall, das Produkt eines Raumes mit sich selbst. Wir nehmen zwei Räume $(\Omega_1, \mathfrak{A}_1, P_1)$ und $(\Omega_2, \mathfrak{A}_2, P_2)$. Die möglichen Ergebnisse des Produktraumes sind einfach definiert als $\Omega_1 \times \Omega_2$, das kartesische Produkt der beiden Ergebnismengen. Im obigen Beispiel wäre das also die Menge

$$\{1, 2, 3, 4, 5, 6\} \times \{1, 2, 3, 4, 5, 6\}.$$

Die Menge der Ereignisse stellt uns allerdings vor einige technische Schwierigkeiten, denn das kartesische Produkt zweier Booleschen Algebren ist nicht notwendig eine Boolesche Algebra. Wir brauchen also eine etwas kompliziertere Definition:

$$(17)\quad \mathfrak{A}_1 \bigotimes \mathfrak{A}_2 := \{\bigcup_{i=1}^{p} A_i \times B_i : \text{für alle } i,\ A_i \in \mathfrak{A}_1, B_i \in \mathfrak{A}_2\}$$

Wahrscheinlichkeiten im Produktraum werden wie folgt definiert:

$$(18)\quad (P_1 \times P_2)(A \times B) := P_1(A) \cdot P_2(B)$$

Natürlich muss $(P_1 \times P_2)$ alle Bedingungen einer Wahrscheinlichkeitsfunktion erfüllen (s.o.). Dann lässt sich mit einiger Mühe folgendes zeigen:

Lemma 3 *Seien $\mathcal{P}_1 = (\Omega_1, \mathfrak{A}_1, P_1)$ und $\mathcal{P}_2 = (\Omega_2, \mathfrak{A}_2, P_2)$ zwei Wahrscheinlichkeitsräume. Dann ist $\mathcal{P}_1 \times \mathcal{P}_2 := (\Omega_1 \times \Omega_2, \mathfrak{A}_1 \times \mathfrak{A}_2, P_1 \times P_2)$, der **Produktraum** der beiden, auch ein Wahrscheinlichkeitsraum.*

4.12 Unabhängige Ereignisse

Zwei Ereignisse sind unabhängig von einander, falls in unserem Wahrscheinlichkeitsraum gilt:

$$P(A|B) = P(A)$$

(das impliziert übrigens dass $P(B|A) = P(B)$. Warum?). Daraus wiederum können wir mithilfe der Definition der bedingten Wahrscheinlichkeiten direkt ableiten:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = P(A)$$
$$\Longleftrightarrow$$
$$P(A \cap B) = P(A|B) \cdot P(B) = P(A) \cdot P(B).$$

Wir können also die Wahrscheinlichkeit von $A \cap B$, falls A, B unabhängig sind, mittels $P(A) \cdot P(B)$ berechnen.

Ein typisches Beipiel für zwei unabhängige Ereignisse ist folgendes: wir werfen einen Würfel zweimal, und uns interessiert die Wahrscheinlichkeit dass wir beim ersten Wurf eine 1, beim zweiten Wurf eine 2 werfen. Woher wissen wir dass die beiden Ereignisse unabhängig sind? Zunächst betrachten wir unseren Wahrscheinlichkeitsraum. Sei $\mathcal{W} = (\Omega, \mathfrak{A}, P)$ der Wahrscheinlichkeitsraum (Bernoulli-Raum) eines einfachen Wurfes eines (gerechten) Würfels. Uns interessiert dann der Produktaum $\mathcal{W} \otimes \mathcal{W}$. Was sind die beiden Ereignisse A=erster Wurf 1, B=zweiter Wurf 2 in diesem Wahrscheinlichkeitsraum? Zunächst gilt: unsere Ergebnisse, d.h. atomare Ereignisse, sind geordnete Paare, und Ereignisse sind Teilmengen von $\Omega \times \Omega$. Daher gilt: $A = \{1\} \times \Omega$, und $B = \Omega \times \{2\}$; die Ereignisse sind also jeweils das kartesische Produkt einer 1-elementigen Menge mit der Menge Ω, wobei Ω einmal zur linken, einmal zur rechten Seite steht. (Warum?)

Wenn wir davon ausgehen, dass die beiden Ereignisse unabhängig sind, können wir leicht deren Wahrscheinlichkeit berechnen: $P(A) = P \times P(\{1\} \times \Omega) = P(\{1\} \cdot 1$; $P(B) = P \times P(\Omega \times \{2\}) = P(\{1\} \cdot 1$. Woher wissen wir, dass die beiden Ereignisse unabhängig sind in unserem Produktraum $\mathcal{W} \times \mathcal{W}$? Wir können das kurz prüfen:

(19)

$$P(\{1\}\times\Omega|\Omega\times\{2\}) = \frac{P(\{1\}\times\Omega)\cap(\Omega\times\{2\})}{\Omega\times\{2\}} = \frac{P(\langle 1,2\rangle}{\Omega\times\{2\}} = \frac{\frac{1}{36}}{\frac{1}{6}} = \frac{1}{6} = P(\{1\}\times\Omega)$$

NB: wir zeigen hier bloß Dinge, die nach unserer Intuition offensichtlich sind. Allerdings ist es wichtig zu wissen, dass der Formalismus mit unseren Intuitionen übereinstimmt.

4.13 Bedingte Wahrscheinlichkeit

Nehmen wir an, Hans hat drei Kinder, und die Wahrscheinlichkeit, einen Jungen zu haben ist $\frac{1}{2}$. Die Wahrscheinlichkeit, dass Hans genau einen Jungen hat, ist $\frac{3}{8}$. (Warum?) Angenommen aber, wir wissen dass Hans eine Tochter hat, wie ist dann die Wahrscheinlichkeit dass er genau einen Sohn hat? Gleich sollte sie nicht sein, denn wir haben die Menge der möglichen Ereignisse reduziert - es ist unmöglich, dass er drei Söhne hat! Also hat sich die Menge der möglichen Ergebnisse geändert, statt 8 Ergebnissen finden wir nur noch 7. Wir nehmen an, dass die Wahrscheinlichkeiten weiterhin gleich verteilt sind. Außerdem gilt nach wie vor: in 3 der 7 Ereignisse hat Hans genau einen Sohn. Also schließen wir: sei A das Ereignis: genau ein Sohn; B das Ereignis: mindestens eine Tochter. Dann ist die Wahrscheinlichkeit von A gegeben B, geschrieben $A|B$, $\frac{3}{7}$.

Das war eine sehr intuitive Art Rechnung. Etwas genauer ist es wie folgt. Wenn wir zwei Ereignisse A, B betrachten, dann gibt es vier die beiden zu kombinieren: (1) A und B treffen ein, (2) A trifft ein, B nicht, (3) B trifft ein, A nicht, (4) keines von beiden trifft ein. Wenn wir nun nach der Wahrscheinlichkeit von $A|B$ fragen, dann haben wir bereits Möglichkeiten (2) und (4) eliminiert, es bleiben also nur (1) und (3). Wir verringern also den Raum der Möglichkeiten; diese sind: $P(A \cap B)$ und $P((\overline{A}) \cap B)$. Wir bekommen also als Wahrscheinlichkeit:

$$(20) \quad P(A|B) = \frac{P(A \cap B)}{P(A \cap B) + P((\overline{A}) \cap B)} = \frac{P(A \cap B)}{P(B)}$$

Die letzte Gleichung folgt, da $(A \cap B) \cup ((\overline{A}) \cap B) = B$, und $(A \cap B) \cap ((\overline{A}) \cap B)) = \emptyset$. Dies definiert die bedingte Wahrscheinlichkeit, und ist bekannt als **Bayes Gesetz der bedingten Wahrscheinlichkeit**.

Bedingte Wahrscheinlichkeiten sind von großer Wichtigkeit nicht nur für die Stochastik, sondern auch für die Statistik. Eine wichtige Konsequenz ist die folgende: wir können die Wahrscheinlichkeit eines Ereignisses $A \cap B$ errechnen durch

$$(21) \quad P(A \cap B) = P(A|B)P(B).$$

Weiterhin haben wir natürlich $A = (A \cap B) \cup (A \cap (\overline{B})$. Da $(A \cap B) \cap$

$(A \cap (\overline{B}) = \emptyset$, gilt also auch $P(A) = P(A \cap B) + P(A \cap \overline{B}$. Daraus folgt:

(22) $P(A) = P(A|B)P(B) + P(A|\overline{B})P(\overline{B})$

Das bedeutet, leicht verallgemeinert, wenn wir eine Partition M von Ω haben, dann müssen wir nur die bedingten Wahrscheinlichkeiten $A|B_i : B_i \in M$ kennen, um die Wahrscheinlichkeit von A zu berechnen.

Der Grund warum bedingte Wahrscheinlichkeiten eine so große Rolle für die Statistik spielen ist der sogenannte **Satz von Bayes**. Oftmals ist unser Ziel, die Ordnung von bedingten Wahrscheinlichkeiten umzukehren. Was wir leicht berechnen können ist die Wahrscheinlichkeit eines Ereignisses in einem gegebenen Wahrscheinlichkeitsraum. In der Statistik verhält es sich aber umgekehrt: wir haben nur ein gewisses Ereignis, und wir möchten Rückschlüsse auf zugrundeliegende Wahrscheinlichkeiten machen. Wir möchten also von P(Ereignis|Wahrscheinlichkeitsraum) zu P(Wahrscheinlichkeitsraum|Ereignis). Der Satz von Bayes gibt uns dazu die Möglichkeit:

(23)
$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A \cap B)}{P(B)} \cdot \frac{P(A)}{P(A)} = \frac{P(B \cap A)}{A} \cdot \frac{P(A)}{P(B)} = P(B|A)\frac{P(A)}{P(B)}.$$

4.14 Verbundwahrscheinlichkeiten und Marginalisierung

Eine wichtiger und grundlegender Begriff der Wahrscheinlichkeitstheorie sind die sog. **marginalen Wahrscheinlichkeiten**. Die marginale Wahrscheinlichkeit von A ist einfach $P(A)$. Das Konzept ist sehr einfach, das Problem ist aber dass wir oft nur die Wahrscheinlichkeit von 2 (oder mehr) gleichzeitig eintretenden Ereignissen oder bedingten Wahrscheinlichkeiten beobachten können. Z.B.:

- Die Wahrscheinlichkeit, dass unser Besuch zu spät kommt, wenn er mit der Bahn kommt ($\cong$ Wahrscheinlichkeit, dass die Bahn Verspätung hat)
- Die Wahrscheinlichkeit, dass unser Besuch zu spät kommt, wenn er mit dem Auto kommt ($\cong$ Wahrscheinlichkeit dass Stau ist)

Angenommen, wir haben (mehr oder weniger korrekte) Wahrscheinlichkeiten für die beiden Ereignisse $P(V|B)$ und $P(V|A)$. Wie kommen wir zur marginalen Wahrscheinlichkeit dass unser Besuch zu spät kommt?

Hierfür brauchen wir noch etwas, nämlich die Wahrscheinlichkeit dass unser Besuch das Auto/die Bahn nimmt, nämlich $P(A)$ und $P(B)$. Nun können wir folgende Tatsache nutzen:

1. A, B schließen sich gegenseitig aus; und
2. eines von beiden muss der Fall sein (nehmen wir mal an).

Das bedeutet: A, B **partitionieren** Ω. Das bedeutet aber auch: $V \cap A$ und $V \cap B$ partitionieren V, also:

$$P(V) = P((V \cap A) \cup (V \cap B)) \tag{24}$$

$$= P(V \cap A) + P(V \cap B) \tag{25}$$

(Axiom 3 der Wahrscheinlichkeitsräume!) Das bedeutet, im Fall einer Partition vereinfacht sich die Summenregel in wesentlich, so dass wir einfach eine Addition bekommen. Mit der Multiplikationsregel können wir den Term folgendermaßen auflösen:

$$P(V) = P(V \cap A) + P(V \cap B) = P(V|A)P(A) + P(V|B)P(B) \tag{26}$$

Da wir diese Werte (nach Annahme) kennen, können wir also die Wahrscheinlichkeit berechnen. Das funktioniert, solange wir Partitionen des Wahrscheinlichkeitsraumes bilden (endlich und sogar unendlich viele); so bekommen wir

die allgemeine Form der Marginalisierung: wir sagen $B_1, ..., B_n$ partitionieren$A$ gdw. gilt:

1. $A \subseteq B_1 \cup ... \cup B_n$ und
2. für alle i, j, falls $i \neq j$, dann $B_i \cap B_j = \emptyset$

Dann gilt: $\{B_1 \cap A, ..., B_n \cap A\}$ ist eine Partition von A.

(27) $$P(A) = \sum_{i=1}^{n} P(A|B_i)P(B_i), \text{ vorausgesetzt } B_1, ..., B_n \text{ partitionieren} A$$

Das funktioniert übrigens sogar mit reellwertigen Parametern, nur brauchen wir dann Integrale (z.B. $B_r : r \in \mathbb{R}$).

4.15 Wahrscheinlichkeitsgesetze – allgemeine Form

Wir haben oben die wichtigsten Regeln der Wahrscheinlichkeitstheorie eingeführt. Zusammen mit dem Begriff der bedingten Wahrscheinlichkeit kann man sie noch allgemeiner formulieren. Zu Übersichtszwecken fügen wir hier nochmal alle zusammen:

Komplementregel: $P(\overline{A}|X) = 1 - P(A|X)$

Summenregel: $P(A \cup B|X) = P(A|X) + P(B|X) - P(A \cap B|X)$

Produktregel: $P(A \cap B|X) = P(A|X)P(B|A, X) = P(B|X)P(A|B, X)$

Bayes Gesetz: $P(A|B, X) = P(B|A, X)\frac{P(A|X)}{P(B|X)}$

Marginalisierung $P(A|X) = \sum_{i=1}^{n} P(A|B_i, X)P(B_i, X)$,
vorausgesetzt $B_1, ..., B_n$ partitionieren A

Übungsaufgabe 1

Nehmen wir Mensch-ärgere-dich-nicht; Sie haben alle Männchen draußen. Wie ist die Wahrscheinlichkeit, dass wir mit 3 Würfen mindestens eine 6 werfen, also ein Männchen ins Spiel bekommen?

Hier gibt es jetzt verschiedene Rechenwege!

Übungsaufgabe 2

Reinhold Messner muss einen steilen Eishang unter einem hängenden Gletscher queren. Die Wahrscheinlichkeit, dass sich während der Dauer seiner Querung von oberhalb eine Schneemasse löst und ihn in die Tiefe reißt, schätzt er auf 1/4. Die Wahrscheinlichkeit, dass er selbst (als erfahrener Eisgeher) bei der Querung ausgleitet und abstürtzt, schätzt er auf 1/20.

1. Wie schätzt er also seine Überlebenschancen für den Fall einer Querung ein? (Vorsicht: bilden Sie die richtigen Partitionen!)

2. Messner hat in seinem Leben 100mal einen vergleichbaren Eishang unter einem vergleichbaren Hängegletscher gequert und hat überlebt. Gleichzeitig beträgt die Wahrscheinlichkeit, auf einer Himalaya-Expedition den Yeti zu sehen, nach Messners Einschätzung $0,000001 = \frac{1}{1.000.000}$. Was war wahrscheinlicher (immer nach Messners Einschätzung) – dass Messner seine 100 Eisquerungen überlebt oder dass er auf einer seiner 25 Himalaya-Expeditionen den Yeti sieht? Begründen Sie!

3. Wir als außenstehende sagen: die Wahrscheinlichkeit, dass Messners Einschätzung bezüglich der Yeti-Wahrscheinlichkeit stimmt, beträgt ebenfalls nur $\frac{1}{1.000.000}$, während es mit einer Wahrscheinlichkeit von $\frac{999.999}{1.000.000}$ den Yeti gar nicht gibt. Was ist also für uns die Wahrscheinlichkeit, dass Messner auf seinen 25 Expeditionen den Yeti wirklich gesehen hat?

Lösung

1. $(\frac{57}{80})$ (auf zwei Wegen: 1-Todeswahrscheinlichkeit, oder P(keine Lawine)·P(kein Ausgleiten))

2. Überlebenswahrscheinlichkeit: $(\frac{57}{100})^{100} \approx 1.89 \cdot 10^{-15}$. Yeti-Wahrscheinlichkeit (nach Messner): $1 - (\frac{999.999}{1.000.000})^{25} \approx 2.5 \cdot 10^{-5}$.

3. Yeti-Wahrscheinlichkeit (nach uns) ist Messners Yeti-Wahrscheinlichkeit mal $\frac{1}{1.000.000}$, also $2.5 \cdot 10^{-5} \cdot 10^{-6} = 2.5 \cdot 10^{-11}$. Also immer noch wahrscheinlicher als sein Überleben!

Hausaufgabe 1a - Beim Metzger

Abgabe bis zum 26.4.2021 *vor dem Seminar*, per email.

Nehmen Sie an, Sie haben Hackfleich vom Metzger im Kühlschrank, das noch gut aussieht, aber Sie wissen nicht mehr, wann Sie es gekauft haben. Der Metzger weiß es auch nicht mehr, aber sagt Ihnen, dass die Wahrscheinlichkeit, dass man von leicht verdorbenem Hackfleich (also solchem, das noch gut aussieht) Bauchweh kriegt, bei 1/3 liegt. Er sagt aber auch, dass Hackfleisch, das noch gut aussieht, allgemein nur in 1/100 aller Fälle (leicht) verdorben ist, und davon abgesehen auch der Verzehr von unverdorbenem Hackfleich in 1/50 aller Fälle zu Bauchweh führt. Sie lassen es sich also schmecken.

1. Wie groß ist die Wahrscheinlichkeit, dass Sie Bauchweh bekommen?
2. Nehmen Sie an, prompt nach dem Essen bekommen Sie Bauchschmerzen. – Wie hoch ist die Wahrscheinlichkeit, dass das Hackfleisch tatsächlich verdorben war?

Hausaufgabe 1b - Eine Krankheit

Abgabe bis zum 26.4.2021 *vor dem Seminar*, per email.

Es geht um eine Krankheit, die durchschnittlich einen von 100.000 Menschen trifft. Um die Krankheit zu diagnostizieren gibt es einen Test. Der Test liefert ein positives Resultat (sagt also aus, dass die Testperson die Krankheit hat) mit einer Wahrscheinlichkeit von 0.99, wenn die Testperson krank ist (Sensitivität 0.99). Auch wenn die Testperson gesund ist, kommt es mit einer Wahrscheinlichkeit von 0.007 zu einem positiven Resultat, also Spezifizität von 0.993)

Sie lassen diesen Test machen und das Ergebnis ist positiv. Wie groß ist die Wahrscheinlichkeit, dass Sie tatsächlich krank sind?

Übung a - Covid

Abgabe bis zum 19.5.2020 *vor dem Seminar*, per email.

Nehmen Sie an, wir haben eine (reale) Fallzahl von 250 pro 100.000 Menschen. Ein Corona-Schnelltest hat

- Sensitivität 0.8
- Spezifizität von 0.98

Sie machen den Test, das Ergebnis ist positiv. Wie ist die Wahrscheinlichkeit, dass Sie tatsächlich an Covid erkrankt sind?

Sie machen den Test, das Ergebnis ist negativ. Wie ist die Wahrscheinlichkeit, dass Sie tatsächlich *nicht* an Covid erkrankt sind?

Übung b - Covid

Wir haben also eine verbliebende Unsicherheit. Können wir diese Unsicherheit verringern, indem wir wiederholt testen?

Übung c - Covid

Nehmen wir an, wir haben ein Gruppe (Familie, Kollegen) mit engem Kontakt von 10 Personen, die wir (schnell)testen. Wir haben 2 positive Tests darunter. Wie ist die Wahrscheinlichkeit, dass es positive Fälle gibt?

Übung d - Covid

Die AstraZeneca-Impfung ist tödlich in $\approx$ 1/1,000,000 Fälle. Eine Covid19 Erkrankung ist tödlich in $\approx$ 1/1000 Fällen. Damit scheint die Sache klar – stimmt aber nicht: denn Sie haben ja nicht die Wahl zwischen Impfung und Covid.

- Wie rechnet man realistisch aus, ob eine Impfung lohnt?
- Wie *klein* müsste die Chance sein, an Covid zu erkranken, damit sich die Impfung *nicht* lohnt?
- Die Covid-Mortalität $\approx$ 1/1000 ist natürlich stark altersabhängig. Nehmen wir (recht willkürlich) an, die Wahrscheinlichkeit an Covid19 zu erkranken beträgt 1/20.

Angenommen, das Impfrisiko ist unabhängig vom Alter und wir denken streng egoistisch (denn Impfungen schützen ja nicht nur uns, sondern immer auch die anderen!!): ab welchem fallbezogenen Todesrisiko lohnt die Impfung? (Wir klammern hier auch alle anderen Probleme, long covid etc. aus)

Übung e - Covid

(Achtung: teilweise Fantasiezahlen!)

Teil 1 Wir haben 7 Fälle von Hirnvenenthrombose (HVT) im zeitlichen Zusammenhang mit der AZ-Impfung, bei 1,7 Millionen Impfungen. Wenn man eine Vergleichsgruppe von 1.7 ungeimpften nimmt, findet man 4 Fälle von HVT. Hieraus ergibt sich nun folgende Frage: kann das Zufall sein?

Teil 2 Es gibt aber folgendes zu beachten: auf 1.7M *Frauen* würde man 7 Fälle von HVT erwarten, auf 1.7M *Männer* nur eine 1. Es stellt sich heraus, dass der Frauenanteil an den geimpften deutlich größer ist (Pflege, Lehr und Betreuungspersonal!), nämlich 0.8. Wie ändert sich die Einschätzung?

Übung f - Covid Die Urstrünge des Virus Sars-CoV19 sind immer nur umstritten, eine umfassende Aufarbeitung wird es aus politischen Gründen wohl in absehbarer Zeit nicht geben. Es bleibt u.a. die Tatsache, dass in Wuhan, wo das Virus zuerst bei Menschen beobachtet wurde, ein großes virologisches Institut steht, in welchem eben mit Coronaviren und Fledermäusen gearbeitet wurde.

Das alleine ist aber keine starke Evidenz für die "Laborhypothese" zum Ursprung des Virus. Welche zuätzliche Info bräuchten wir, um die Stärke der Evidenz einzuschätzen?

5 RX1 Erste Schritte in R

R ist eine freie Statistik-Software. R funktioniert zunächst wie ein Taschenrechner: wir können alle möglichen einfachen und komplexen Berechnungen ausführen. In in der Kommandozeile R arbeitet man mit Befehlen, die unmittelbar kompiliert werden mit 'enter'. Man sieht das am Zeichen '>'. Wenn man einen Befehl eingibt der nicht vollständig ist, erscheint ein '+', was bedeutet: das 'enter' wird als reiner Zeilenumbruch gewertet, der Befehl kann beendet werden. Falls man einen Fehler gemacht hat und zurück möchte zur normalen Kommandozeile, kann man eingeben:

+ 'esc'
+ 'strg-c'

Das ist etwas mühsam, deswegen ist es besser eine Plattform wir RStudio zu nutzen, wo man erst ganze Skripte erstellt, die man dann kompiliert.

R-Aufgabe 1

Implementieren Sie, nur unter Nutzung elementarer arithmetischer Operationen, die Funktion $\binom{n}{k}$, wobei

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{28}$$

Dazu müssen Sie natürlich zunächst die Funktion $n!$ implementieren.

R-Aufgabe 2

Schreiben Sie eine Funktion f so dass $f(n)$ die $n$te ersten Fibonacci Zahl ist (zur Erinnerung: 1,1,2,3,5,8,13,...).

5.1 Auf Daten zugreifen in R

Zunächst installieren wir die Datensätze die wir brauchen:

> install.packages("languageR", repos ="http://cran.r-project.org")

(Vorsicht beim copy-paste: evtl. müssen Sie die Anführungszeichen von Hand korrigieren, sonst gibt es eine Fehlermeldung). Nun laden wir die Daten:

> library(languageR)

.Rprofile in home directory mit dem Befehl! Wir besprechen nun unseren ersten Datensatz, den Datensatz *verbs* über die englische Dativalternation

> head(verbs, n=10)

Wir können die Daten auch in ein lesbares Format exportieren:

> write.table(verbs, file = "path/name.txt")

und wiederum importieren:

> verbs = read.table("path/name.text)

Hilfe findet man wie folgt:

> help(verbs)
> example(verbs)

Unser Datensatz ist eine Tabelle. Also greifen wir nach Zeile und Spalte darauf zu:

> verbs[2,4]

Damit greift man auf Zellen zu: name[Zeile,Spalte]

> verbs[,5]

liefert die Spalte 5, da alle Zeilen! Ebenso:

> verbs[4,]

liefert Zeie 4. Also:

> verbs[,] # soviel wie verbs

Man kann auch nach dem header auf Spalten zugreifen:

> verbs$LengthOfTheme # soviel wie verbs[,5]

Wir können hier auch beliebig neue Variablen deklarieren:

> row1 = verbs[1,]
> col5 = verbs[,5]
> head(col5, n=5)

NB: diese Variablen stehen wieder für Datensätze, aber diesmal eindimensionale:

> row1[1]
> col5[4]

row1 ist eine Variable mit header, wir können also auch abrufen:

> row1["RealizationOfRec"]

Hier wird eigentlich ein Integer erwartet, der String muss also als solcher markiert werden mit "...".

col5 hat keinen header, also gibt es diese Möglichkeit nicht.

Wir haben nun gesehen, wie wir Zellen, Zeilen und Spalten extrahieren können. Manchmal möchten wir größere Teile einer Tabelle extrahieren. Dafür müssen wir Vektoren kreieren. Das kann man machen mit z.B.

> 1:5

Wenn wir bestimmte Werte einsetzen wollen, geht das auch einfacher mit

> c(5,23,1,17)

Wir können natürlich Variablen deklarieren

> rs = c(5,23,1,17)

Nun können wir einen Teil der Tabelle extrahieren:

> verbs[rs,]

NB: wir extrahieren nicht nur, wir ordnen die Zeilen auch nach unserem Vektor.

Wir können auch abrufen:

> verbs[rs,1:4]

Man kann auch einen Vektor mit Namen der Spalten nutzen:

> verbs[rs, c("RealizationOfRec","Verb","AnimacyOfRec")]

Bislang haben wir Daten nach ihren "Koordinaten" extrahiert, unabhängig von ihrem Inhalt. Uns interessieren aber normalerweise gerade die Inhalte. In unserem Beispiel kann das sein: wir möchten unterscheiden nach AnimacyOfTheme (entweder animate oder inanimate)

> verbs[verbs$AnimacyOfTheme == "animate",]

Das nimmt also die Zeilen mit Wert "animate", und davon alle Spalten. verbs$AnimacyOfTheme == "animate" ist also eine Bedingung an die Zeile. Man kann das auch schreiben:

> subset(verbs, AnimacyOfTheme == "animate")

Es handelt sich also hierbei um eine Art Kontrollstruktur (if ... else ...) mit einem Wert in Boolean. Dementsprechend können wir auch beliebige Boolesche Operatoren nutzen:

> verbs[verbs$AnimacyOfTheme == "animate" & verbs$LengthOfTheme >2,]

Logisches und ist &, oder |, und nicht ist !
Es gibt 2 Funktionen colnames und rownames:

> head(rownames(verbs))
> head(colnames(verbs))

Die geben jeweils die Namen der Zeilten/Spalten. NB: beides sind Strings, auch wenn die Zeilen nummeriert sind!

> verbs["1",]

Das ist aus folgendem Grund nützlich: wir definieren:

> verbs.rs <- verbs[rs,]
> verbs.rs["23",]
> verbs["23",]

Das Resultat ist in diesem Fall dasselbe, sonst nicht. $ greift auf Spalten zu, wie wir gesehen haben. Als einfaches Kommando geht das so:

> verbs.rs$AnimacyOfRec

Hier ist "levels" die Menge der mögliches Werte in den Zellen.

Wichtig: Es gibt Zeilen- und Spaltennamen, das sind strings Alle anderen nicht-numerischen Eingaben werden automatisch in sog. factors bzw. levels konvertiert: das sind soz. atomare Prädiktoren (dazu später mehr). Formal bedeutet das: in einem dataframe ist die Menge der Objekte, die stehen kann, geschlossen - nämlich die Menge der Werte die stehen. Dementsprechend stehen die Werte nicht mit "..." - sind also keine Strings!
Wenn man levels wieder als Strings lesen will, geht das mit dem Befehl as.character:

> verbs.rs$AnimacyOfRec = as.character(verbs.rs$AnimacyOfRec)
> verbs.rs$AnimacyOfRec

Zurück geht es mit as.factor():

> verbs.rs$AnimacyOfRec = as.factor(verbs.rs$AnimacyOfRec)

Hausaufgabe 2

Zu bearbeiten bis zum 3.5.2021

1. Erstellen Sie einen dataframe, der alle Zeilen von verbs enthält, in denen das Thema mehr als 5 Buchstaben hat.

2. Erstellen Sie einen dataframe, der alle Zeilen von verbs enthält, in denen das Verb mehr als 5 Buchstaben hat. Hier brauchen Sie einige zusätzliche Befehle, die wir nicht besprochen haben: nchar() gibt die Länge eines Wortes; allerdings müssen Sie zunächst die levels zu strings transformieren (as.character()).

5.2 Operationen auf dataframes

Soweit haben wir auf Daten zugegriffen. Was wir jetzt wollen ist Datensätze manipulieren. Als erstes wollen wir Daten **sortieren**. Das geht mittels des Befehls **order()**, der sich jeweils auf eine Spalte oder Zeile beziehen kann:

```
> verbs.rs[order(verbs.rs$RealizationOfRec),]
```

In diesem Fall haben wir die verbs.rs erst sortiert, dann visualisiert. NB: sortieren muss nicht nach numerischen Werten geschehen; es geht auch nach levels oder strings; in diesem Fall wird alphabetisch sortiert. Man kann auch gleichzeitig nach mehreren Spalten sortieren:

```
> verbs.rs[order(verbs.rs$Verb, verbs.rs$LengthOfTheme), ]
```

Die etwas verwickelte Syntax erklärt sich so: Was order() tatsächlich liefert ist ein Vektor von Zeilennummern, in dem zuerst die Zeilennummer der zuerst zu listenden Zeile steht, dann die zweite etc. Also:

```
> order(verbs.rs$Verb)
```

Damit kann man z.B. auch einfache Vektoren alphabetisch sortieren:

```
> vec = c("stehen", "gehen", "laufen", "rennen", "humpeln")
> order(vec)
> vec[order(vec)]
```

Um Vektoren zu sortieren gibt es noch eine andere Funktion, nämlich **sort()**:

```
> sort(vec)
> vec = sort(vec)
```

Wir können auch Einträge in dataframes ändern:

```
> verbs.rs[1, ]$RealizationOfRec = "NP"
```

Das ist natürlich etwas umständlich. Man kann auch ganze Spalten und

Zeilen ändern. Z.B. wenn wir in LengthOfTheme aus den Logarithmen die tatsächliche Länge rekonstruieren wollen geht das so:

> verbs.rs$LengthOfTheme

liefert den Vektor;

> exp(verbs.rs$LengthOfTheme)

die Exponentialfunktion davon. Wir können also Funktionen $f : \mathbb{R} \to \mathbb{R}$ auf Vektoren applizieren; das Ergebnis ist die **punktweise Applikation**.

We können auch neue Spalten zu unserem dataframe hinzufügen, z.B. eine Spalte mit der Länge des Verbs. Es gibt eine Funktion **nchar()** von Strings zu Integers; die lässt sich wie vorher auch einfach auf Vektoren applizieren:

> nchar("Sonnenschein")
> nchar(c("Sonnenschein", "als"))

Wir können nun mit dem Operator $ eine neue Spalte erstellen:

verbs.rs$Length =nchar(as.character(verbs.rs$Verb))

Das Ergebnis kan man sich anschauen:

> verbs.rs[1:4,]
> verbs.rs[1:4,c("Verb", "Length")]

5.3 Korrelationen und Ähnliches

Wir kommen nun zu einem anderen Thema, nämlich wie man sog. **contingency tables** aus einem dataframe extrahiert. Solche Tabellen erlauben uns, die Kookurrenzen von gewissen Werten (*levels*) einfach darzustellen bzw. abzurufen. Wir wissen dass RealizationOfRec und AnimacyofRec beide binär sind; man kann Kookurrenzen also in einer 2×2 Tabelle darstellen.

> levels(verbs$RealizationOfRec)

> levels(verbs$AnimacyofRec)

Mit dem Befehl **xtabs()** kann man nun die Werte gegeneinander darsellen:

> xtabs(~ RealizationOfRec + AnimacyOfRec, data = verbs)

xtabs() nimmt 2 Argumente. 2 ist offensichtlich der Datensatz; das erste Argument ist eine sog. **Formel**. Formeln haben im Allgemeinen die Struktur

abhängige Variable ~ Prädiktor1 + Prädiktor2 + Prädiktor3 ...

Die **abhängige Variable** wäre der Wert, den wir vorhersagen möchten; die Prädiktoren dagegen die Werte, die wir dafür als gegeben betrachten müssen. Man nennt diese dann die **unabhängige Variable**. Wichtig ist: ob eine Variable (un)abhängig ist hat *nichts* mit der Realität und auch nichts mit unseren Daten zu tun; es hängt davon ab, was uns interessiert, also was wir mittels unserer Daten vorhersagen möchten.

In verbs-dataframe wäre die "abhängige Variable" Realization of Recipient, alles andere wäre unabhängig. Das sagt aber *nichts aus* darüber, welche Abhängigkeiten zwischen den verschiedenen Faktoren bestehen.

(Als eine private Anmerkung: ich finde diese Bezeichnung furchtbar und würde sie selber nie benutzen; sie ist aber gängig in gewissen statistischen Schulen, also erkläre ich sie.)

Wir kommen zurück zu unserer Datentabelle. In diesem Fall ist die abhängige Variable nichts – wir möchten einfach nur Datenpunkte gegeneinander plotten und noch nichts vorhersagen, es gibt also keine abhängige Variable, wir möchten einfach nur Variablen gegeneinander plotten.

Das geht übrigens auch mit mehr als zwei Spalten ("Variablen"):

> verbs.xtabs =
+ xtabs(~ AnimacyOfRec + AnimacyOfTheme + RealizationOfRec, data=verbs)
> verbs.xtabs

Hier sehen wir, wie die dritte Dimension dargestellt wird: R pickt sich RealizationOfRec heraus und unterscheidet zwei Fälle (binäres Merkmal); dementsprechend lassen sich die Daten in zwei Tabellen darstellen.

Wir sehen auch, dass belebte Themen (AnimacyOfTheme="animate") sehr selten ist. Deswegen macht es eventuell Sinn, diese Fälle aus Ausnah-

men auszuschliessen. Das geht wie folgt:

```
> verbs.xtabs =
+ xtabs( ~ AnimacyOfRec + RealizationOfRec,
+ data=verbs, subset = AnimacyOfTheme != "animate")
> verbs.xtabs
```

Hier sehen wir ganz klar: Recipients scheinen eher als NP realisiert zu werden von belebt, als PP wenn unbelebt.

Diese Tabelle liefert uns absolute Häufigkeiten von Ko-okkurrenzen; es wäre vielleicht ganz nett wenn wir diese wiederum transformieren könnten zu relativen Häufigkeiten. Dazu müssen wir die Gesamtzahl der Beobachtungen kennen, die der Tabelle zugrundeliegen, also die Gesamtzahl der Einträge mit AnimacyOfTheme=inanimate. Das geht mit folgendem Befehl:

```
> norm <- sum(verbs.xtabs)
> norm
```

Hier bekommen wir die Summe aller Zelleneinträge der Tabelle. Nun können wir folgendes machen:

Nun können wir die Tabelle "normalisieren", indem wir folgendes Eingeben:

```
> verbs.xtabs/norm
```

Die Operation wird also *punktweise* auf den Zellen ausgeführt. Wir können das ganze auch als Prozente transformieren:

```
> verbs.xtabs.proz <- (verbs.xtabs/norm)*100
```

Solche **Proportionen** machen die Lage häufig überschaubarer. Es gibt auch einen eigenen Befehl hierfür, nämlich *prop.table()*. Das nimmt zwei Argumente: eine Tabelle, und $\{1, 2\}$

```
> prop.table(verbs.xtabs, 1)      # Zeilen summieren auf 1
> prop.table(verbs.xtabs, 2)      # Spalten summieren auf 1
```

Zwischen den beiden Tabellen sehen wir jetzt deutliche Unterschiede:

1. In diesem Fall sind die Werte annähernd gleich verteilt.

2. In diesem Fall finden sich in der unteren Zeile ein Unterschied um den Faktor > 2.

Diese Tabellen sind die Grundlage für spätere **Signifikanztests**, mit denen man zu belegen sucht, dass diese Verteilung *kein Zufall* ist.

Block-Aufgabe 2

1. Erstellen Sie einen dataframe, der alle Zeilen von verbs enthält, in denen das Thema mehr als 5 Buchstaben hat.

2. Erstellen Sie einen dataframe, der alle Zeilen von verbs enthält, in denen das Verb mehr als 5 Buchstaben hat. Hier brauchen Sie einige (und nur) Befehle, die wir besprochen haben: nchar() gibt die Länge eines Wortes; allerdings müssen Sie zunächst die levels zu strings transformieren (as.character()).

3. Erstellen Sie einen dataframe, der alle Zeilen von verbs enthält, die folgende Bedingung erfüllen: Verb und Theme haben *zusammen* mehr als 10 Buchstaben.

4. Sortieren Sie den Datensatz verbs nach der Länge der Verben!

Erinnern Sie sich auch daran: wenn man ein komplexes Problem hat, lst man erstmal die einfachen Bestandteile des Problems; der Rest ergibt sich meistens ganz gut!

Übrigens gibt es ein Problem: die Werte in LengthOfTheme sind Logarithmen, als solche aber natürlich gerundet (Logarithmen sind blicherweise irrationale Zahlen). Wenn sie exponentieren, werden sie nicht immer genau den Ausgangswert treffen. Um Fehler diesbezüglich zu vermeiden, nutzen Sie am besten die Funktion `round()`, die alle Zahlen rundet zu integer-Werten.

5.4 Rechnen mit dataframes

Eine weitere Frage die man sich stellen kann, ist folgende: in welchem Maße ist die *Länge des Themas* (in Buchstaben) bestimmt bzw. beeinflusst durch AnimacyOfRecipient? Könnte es z.B. sein dass belebte Recipients eher komplexe Themen präferieren, im Gegensatz zu unbelebten Recipients?

In diesem Fall ist die Situation etwas anders als vorher, denn wir wollen nicht ein (binäres) Merkmal (*level*) vorhersagen, sondern einen numerischen Parameter, der zumindest theoretisch beliebig viele Werte annehmen kann (auch wenn er es praktisch nicht tun wird). Um also überhaupt Zugang zu dem Problem zu haben, brauchen wir den **Mittelwert** der Länge der Themen für belebte bzw. unbelebte Recipients. Der Mittelwert eines Vektors $\vec{v} \in \mathbb{R}^n$ ist defiert durch

$$\mu(\vec{v}) = \sum_{i=1}^{n} v_i \cdot \frac{1}{n} \tag{29}$$

In R gibt es eine eingebaute Funktion hierfür, nämlich **mean()**. Eingabe ist ein Vektor, Ausgabe ist eine Zahl.

```
> mean(1:10)
```

Hiermit können wir sehr einfach die Mittelwerte der Länge des Themas berechnen:

```
> mean(verbs[verbs$AnimacyOfRec == "animate", ]$LengthOfTheme)
> mean(verbs[verbs$AnimacyOfRec == "inanimate", ]$LengthOfTheme)
```

Hier haben wir Logarithmen; wir können die Werte natürlich wieder rücktransformieren:

```
> exp(mean(verbs[verbs$AnimacyOfRec == "animate", ]$LengthOfTheme))
> exp(mean(verbs[verbs$AnimacyOfRec == "inanimate", ]$LengthOfTheme))
```

Etwas bequemer kann man diese Werte berechnen mit der Funktion **tapply()**, die drei Argumente nimmt:

1. Ein Vektor mit Zahlen aus einer Tabelle, dessen Mittelwert (z.B.) wir suchen

2. Ein Vektor mit levels aus derselben Tabelle, in Abhängigkeit von dem wir Mittelwerte suchen

3. Die Funktion auf dem Vektor, in unserem Fall also **mean()**

> tapply(verbs$LengthOfTheme, verbs$AnimacyOfRec, mean)

Die Ausgabe ist eine Tabelle mit zwei Mittelwerten, je nach *level* des zweiten Argumentes. Hier haben wir wieder einen Datensatz, auf dem wir die typische Frage der orthodoxen Statistik stellen können: kann das wirklich Zufall sein?

Man kann diese Funktion auch allgemeiner anweden; dafür brauchen wir **Listen**. Listen sind wie Vektoren geordnete Strukturen, mit dem Unterschied dass die Elemente einer Liste **komplex** sein können. Man erstellt Listen mit dem Befehl **list()**. Argumente können z.B. Spalten unseres Datensatzes sein; dann bekommen wir:

> tapply(verbs$LengthOfTheme, list(verbs$AnimacyOfRec, verbs$AnimacyOfTheme), mean)

Hier wird also wiederum nach zwei Merkmalen der Mittelwert bestimmt, so dass wir das als eine 2×2 Tabelle lesen können. Für mehr als zwei Merkmale wird die Visualisierung schwieriger.

An diesem Beispiel kann man auch die Funktion **with()** illustrieren. Alle unsere Daten sind Teil des Datensatzes *verbs*. Das hat zur Folge, dass wir ziemlich of "verbs$" schreiben müssen. Mit **with()** geht das einfacher:

> with(verbs, tapply(LengthOfTheme, list(AnimacyOfRec, AnimacyOfTheme), mean))

Hier werden also alle Argumente, die einen Datensatz denotieren, automatisch auf *verbs* bezogen.

Eine letzte Operation lässt sich am besten illustrieren mit dem Datensatz **heid** aus *languageR*.

> heid[1:5,]

Hier handelt es sich um das Ergebnis eines sog. **lexical decision tasks**: Versuchspersonen müssen entscheiden, ob ein "Wort" ein echtes Wort ist oder nicht; hierbei die Zeit gemessen, die sie für diese Entscheidung brauchen. Es gibt also für jedes Paar

Versuchsperson – Wort

eine Reaktionszeit. In unserem Datensatz sind die Worte Neologismen, die einem

Wortstamm + *heid*-Suffix

gebildet werden. Zusätzlich gibt es noch die Häufigkeit des Wortstammes. Die zugrundeliegende Motivation dfür das sammeln dieser Daten ist folgende Hypothese:

> **Hypothese**: Je höher die Frequenz des Wortstammes, desto geringer die Reaktionszeit, natürlich gemittelt.

Nun kann man aber auf mehrere Arten mitteln:

- Mitteln über Versuchspersonen
- Mitteln über Worte
- ...

Die Funktion hierfür ist **aggregate()**. Die Syntax ist ähnlich der **tapply()**-Funktion:

1. Argument 1 ist ein numerischer Vektor
2. Argument 2 ist die Liste der *levels*, nach denen wir ihn unterteilen
3. Argument 3 ist die Funktion die wir applizieren

Der Unterschied zwischen tapply() und aggregate() besteht ist folgender:

- tapply() erstellt eine Tabelle, dient also in erster Linie der Visualisierung. Weitere Modifikationen mit dieser Tabelle sind aber schwer zu machen.

- aggregate() erstellt einen neuen dataframe, mit dem wir weiter arbeiten können. Deswegen brauchen wir im folgenden die Funktion aggregate(); mit tapply() wäre nichts zu machen!

Wir berechnen also den Mittelwert über Personen:

```
> heid2 <- aggregate(heid$RT, list(heid$Word), mean)
> heid2[1:5,]
```

Diese Funktion präserviert *nicht* die Spaltennamen; wir müssen sie also hinzufügen wenn wir die Daten verständlich halten wollen:

```
> colnames(heid2) = c("Word", "MeanRT")
```

Ebenso haben wir nun eine wichtige Information verloren, nämlich Base-Frequency; genau diese Information brauchen wir aber für unsere Hypothese. Wir müssen also unseren dataframe rekonstruieren:

```
> items = heid[, c("Word","BaseFrequency")]
```

Da jedes Subjekt zu denselben Worten gestestet wurde, haben wir ständig sich wiederholende Einträge in beiden Spalten. Man kann diese Woederholungen löschen mit **unique()**:

```
> nrow(items)
> items = unique(items)
> nrow(items)
> items[1:5,]
```

Jetzt haben wir zwei dataframes: einen mit Worten und BaseFrequency, und einen mit Worten und RT. Jetzt müssen wir die beiden nur noch zusammenbringen. Das geht mit der Funktion **merge()**. Das nimmt 4 Argumente:

- Den ersten dataframe
- Den zweiten dataframe
- Die "Schlüssel" für das *matching*: der Schlüssel in dataframe 1
- Den Schlüssel in dataframe 2

Das ganze sieht also so aus:

```
> heid3 <- merge(heid2, items, by.x = "Word", by.y = "Word")
> head(heid3, n=10)
```

In beiden frames heißt die matchende Spalte "Word", ist also gleich. Hier werden dann Zeilen, in denen es einen match gibt, zusammengefügt.

Dasselbe kann man natürlich auch auf andere Art und Weise machen, ohne dass man dataframes mergen muss!

```
> heid4 = aggregate(heid$RT, list(heid$Word, heid$BaseFrequency), mean)
> colnames(heid4) = c("Word", "BaseFrequency", "MeanRT")
> heid4 = heid4[order(heid4$Word),]
> head(heid4, n=10)
```

Was können wir also sagen über MeanRT und BaseFrequency? Wir werden es sehen!

Block-Aufgabe 3a

Nehmen Sie den Datensatz heid3, den wir definiert haben. Transformieren Sie die Spalte BaseFrequency wie folgt: aus $x \in \mathbb{R}$ wird "high", falls x größer ist als der Mittelwert der Spalte, und auf "low" andernfalls.

Als nächstes berechnen Sie den Mittelwert von MeanRT in Abhängigkeit von BaseFrequency mit den Befehlen, die wir besprochen haben.

Tipp: für den ersten Teil der Aufgabe können Sie eine *for*-Schleife über den Datensatz nutzen; es gibt aber auch andere Mittel und Wege.

Es gibt hier noch ein mögliches Problem, das manchmal auftritt: wenn man einen Eintrag in einer Tabelle in einen string umwandelt, werden *alle* Einträge als strings aufgefasst, Vergleichsoperationen sind also möglicherweise undefiniert. Falls das Problem auftritt, kann man es lösen indem man den Datensatz klont – Modifikation auf dem Klon, Prüfung der Bedingung auf dem Original.

Block-Aufgabe 3b

LengthOfTheme ist logarithmisch, daher die komischen Zahlen.

- Invertieren Sie das.
- Implementieren Sie eine Funktion, die fr jede Länge die Anzahl der Datenpunkte mit dieser Länge liefert.
- Das Ergebnis knnen Sie als Funktion plotten.
- Nun transformieren Sie die Anzahl (nicht die Länge!) logarithmisch.
- Wie sieht das Ergebnis aus?

6 Zufallsvariablen

6.1 Definition

Erinnern Sie sich dass für eine Funktion f wir mit f^{-1} soviel meinen wie die Umkehrfunktion. Da die einfache Umkehrung einer Funktion nicht unbedingt eine Funktion ist (wegen fehlender Eindeutigkeit), ist die formale Definition wie folgt:

(30) $f^{-1}(a) = \{b : f(b) = a\}$

$f^{-1}(x)$ ist also immer eine Menge, und falls es kein b gibt, so dass $f(b) = a$, dann gilt

(31) $f^{-1}(a) = \emptyset$

Mit diesem Wissen und dem Wissen dass $\emptyset$ in jedem Wahrscheinlichkeitsraum enthalten ist, werden Sie die folgende Definition besser verstehen.

Sei $\mathcal{P} = (\Omega, \mathfrak{A}, P)$ ein Wahrscheinlichkeitsraum, und

$$X : \Omega \to \mathbb{R}$$

eine Funktion. X ist eine **Zufallsvariable** falls für alle $x \in \mathbb{R}$,

$$X^{-1}(x) \in \mathfrak{A}.$$

In einem *diskreten* Wahrscheinlichkeitsraum ist jede Funktion $X : \Omega \to \mathbb{R}$ eine Zufallsvariable. Das bedeutet:

$$P(X^{-1}(x)) \in [0, 1]$$

ist eine definierte Wahrscheinlichkeit; wir schreiben das oft einfach

$$P(X = x),$$

und sagen: die Wahrscheinlichkeit dass X den Wert x annimmt.

NB: eine Zufallsvariable ist keine Variable, sondern eine Funktion; der irreführende Name wurde aus dem Englischen *random variable* rück-übersetzt. Der eigentliche Deutsche Begriff Zufallsgröße ist aber (meines Wissens) nicht mehr gebräuchlich.

Zufallsvariablen werden oft benutzt, um Wahrscheinlichkeitsräume zu vereinfachen. Nehmen wir das obige Beispiel mit den Verspätungen in Auto

und Bahn: es können viele Dinge geschehen mit einer gewissen Wahrscheinlichkeit. Wir können nun eine Zufallsvariable definieren, die alle Ereignisse auf die Verspätung abbilden (=Zahl der Minuten), in der sie resultieren. $P(X = 30)$ wäre dann die Wahrscheinlichkeit, dass unser Besuch 30min Verspätung hat.

6.2 Erwartungswert

Zufallsvariablen wecken gewisse Erwartungen. Der **Erwartungswert** über einer Zufallsvariablen ist wie folgt definiert:

$$\mathbf{E}(X) := \sum_{x \in \mathbb{R}} x \cdot P(X^{-1}(\{x\})) \tag{32}$$

Statt $P(X^{-1}(\{x\}))$ schreibt man meist $P(X = x)$, d.h. die Wahrscheinlichkeit, mit der X den Wert $x \in \mathbb{R}$ zuweist. Wenn wir beispielsweise die Werte von X als Geldwerte auffassen, die wir in einem Spiel gewinnen (oder im Fall von negativen Zahlen verlieren), dann ist der Erwartungswert soviel wie der Geldbetrag, den wir in einem durchschnittlichen Spiel gewinnen/verlieren (gemittelt sowohl über den Betrag als auch die Wahrscheinlichkeit des Gewinns/Verlustes!).

Wenn wir eine diskrete Wahrscheinlichkeitsfunktion haben, also jedes Ergebnis (nicht Ereignis!) eine Wahrscheinlichkeit bekommt, dann gibt es eine wesentlich einfachere Definition:

$$\mathbf{E}(X) := \sum_{\omega \in \Omega} X(\omega) \cdot P(\omega) \tag{33}$$

6.3 Würfeln - mal wieder

Nehmen wir an, wir werfen zwei faire Würfel. Das führt zu einem Produktraum zweier Laplace-Räume, der wiederum ein Laplaceraum ist. Wir definieren nun eine Zufallsvariable X auf unserem Produktraum durch

$$X(\langle x, y\rangle) = x + y.$$

D.h. z.B. $X(\langle 3, 4\rangle) = 7$, wobei $\langle 3, 4\rangle$ das Ergebnis "erster Wurf 3, zweiter Wurf 4" darstellt. X entspricht einem Spiel, indem nur die Summe der Würfel eine Rolle spielt.

Die Zufallsvariable X eröffnet uns jetzt einen neuen Wahrscheinlichkeitsraum, nämlich

$$(X[\Omega], \wp(X[\Omega]), P \circ X^{-1}).$$

$X[-]$ ist die punktweise Erweiterung von X auf eine Menge, z.B.

$$X[\{a, b\}] = \{X(a), X(b)\}.$$

D.h. also in unserem Beispielfall $X[\Omega] = \{2, 3, ..., 12\}$. Mit $P \circ X^{-1}$ meinen wir die Komposition der beiden Funktionen, also

$$P \circ X^{-1}(x) = P(X^{-1}(x)).$$

Das sieht komplizierter aus als es ist. Was ist bespielsweise die Wahrscheinlichkeit von 2 in unserem neuen Raum? Nun, wir haben $X^{-1}(2) = \{\langle 1, 1\rangle)\}$, und $P(\langle 1, 1\rangle) = \frac{1}{36}$. Was ist die Wahrscheinlichkeit von 5? Intuitiv sollte die höher sein, denn es gibt ja einige Möglichkeiten mit zwei Würfeln 5 Augen zu werfen. Und tatsächlich haben wir

$$P \circ X^{-1}(5) = P(\{\langle 1, 4\rangle, \langle 2, 3\rangle, \langle 3, 2\rangle, \langle 4, 1\rangle\}) = 4 \cdot \tfrac{1}{36} = \tfrac{1}{9}.$$

Wir sehen also: der neue Wahrscheinlichkeitsraum ist kein Laplace-Raum!

Was ist der Erwartungswert? Auch diesmal können wir (zum Glück!) die einfachere Formel benutzen, da wir ja im alten Wahrscheinlichkeitsraum das kleine p haben - jedes atomare Ergebnis hat ja die Wahrscheinlichkeit $\frac{1}{36}$. Wir können also diesen Term ausklammern. Wir machen die Summe etwas einfacher, indem wir die Mengen

$$\{(1, 1), ..., (6, 1)\}$$

etc. zusammenfassen; wir haben

1+2+...+6=21

zu dieser Zahl addieren wir 6 · zweite Komponente; so bekommen wir:

$$\begin{aligned} \mathbf{E}(X) &= \sum_{\omega \in \Omega} X(\omega) \cdot p(\omega) \\ &= \frac{27 + 33 + 39 + 45 + 51 + 57}{36} \\ &= 7 \end{aligned} \tag{34}$$

Das bedeutet, wir erwarten im Schnitt mit einem Wurf 7 Augen zu bekommen.

6.4 Erwartungswerte bei Wetten – quantifizierte Gewißheit

Stellen Sie sich vor, Sie wetten um einen Euro über einen gewissen Sachverhalt, z.B.: Sie sagen, `Al Pacino war der Hauptdarsteller im Paten`, Ihr Gegenüber sagt: `es war Robert de Niro`. Nun sind Sie nicht zu 100% sicher, aber zu 90%. Wir haben:

- Einen diskreten Wahrscheinlichkeitsraum mit drei Ergebnissen, A(l),R(obert),K(einer der beiden).
- Ihre subjektive Wahrscheinlichkeitsverteilung $P(A) = 0.9$, $P(R) = 0.05$, $P(K) = 0.05$.
- Die Variable $X : \{A, R, K\} \to \mathbb{R}$, mit der Definition $X(A) = 1$, $X(R) = -1$, $X(K) = 0$.

In diesem Fall gilt: der Erwartungswert ist das, was Sie zu gewinnen erwarten, ihr erwarteter Gewinn bzw. Verlust. Hier sehen wir, dass Wahrscheinlichkeiten subjektiv sind, denn Ihr Gegenüber wird natürlich ganz andere Wahrscheinlichkeiten verteilen.

Man kann dasselbe Szenario noch ausbauen und Erwartungswerte nutzen, um **Gewißheiten zu quantifizieren**. Z.B. könnte Ihr gegenüber sagen: Die Wette, wie wir sie oben besprochen haben, gehe ich nicht ein. Aber wenn Du so sicher bist, dann können wir ja folgende Wette machen:

- Falls A stimmt, gebe ich Dir einen Euro;
- falls R stimmt, gibst Du mir 10;
- andernfalls bleiben wir bei 0.

Die Frage ist: ist das eine gute Wette? Wir haben

$$0.05 \cdot -10 + 0.05 \cdot 0 + 0.9 \cdot 1 = -0.5 + 0.9 = 0.4 \tag{35}$$

Die Antwort wäre also: ja, wir erwarten einen Gewinn von 0.4 Euro. Alternativ gäbe es folgende Wette:

- Falls A stimmt, gebe ich Dir einen Euro;
- falls A nicht stimmt, gibst Du mir 10;

In diesem Fall gilt:

(36) $0.1 \cdot -10 + 0.9 \cdot 1 = -0.1$

Die Wette wäre also nicht mehr gut für uns, denn unsere Sicherheit beträgt 1:9.

Was wir hier sehen ist lehrreich aus folgendem Grund: wir sind oft schlecht darin, unsere subjektive (Un)sicherheit in Zahlen auszudrücken. Mittels des Wettmodels sind wir aber in der Lage zu quantifizieren:

Meine (subjektive) Gewißheit, dass Eregnis A eintritt, ist $> n/m$ (wobei $n < m$, genau dann wenn ich mir bei einer Wette vom letzten Typ, bei einer Verteilung $X(A) = n$, $X(\overline{A}) = -(m - n)$, einen Gewinn erwarten (= positiver Erwartungswert).

6.5 Noch ein Beispiel: Erwartete Länge von Wörtern im Text

Nehmen wir an wir haben eine Sprache mit endlich vielen Wörtern (im Gegensatz zum Deutschen), also etwa das Englische. Nehmen wir ebenfalls an, wir kennen für jedes englische Wort w die Wahrscheinlichkeit, mit der w in irgendeinem zufälligen Text an einer zufälligen Stelle auftritt; wir haben also eine diskrete Wahrscheinlichkeitsverteilung, gegeben durch die diskrete Verteilung

$$P : \Sigma^* \to [0, 1]$$

Was uns interessiert ist folgendes: wenn wir immer wieder einen zufälligen Text aufschlagen und ein zufälliges Wort heraussuchen, wie *lang* wird dieses Wort im Durchschnitt sein? Oder anders gesagt, wie lang ist das durchschnittliche englische Wort?

Um diese Frage zu beantworten, brauchen wir zunächst die Funktion

$$|-| : \Sigma^* \to \mathbb{N}$$

wobei $|w|$ die Länge von w denotiert. Eine erste Antwort auf die Frage nach der erwarteten Länge eines durchschnittlichen englischen Wortes wäre wie folgt: denotieren wir das englische Lexikon mit L; erinnern Sie sich außerdem dass für Mengen (statt Ketten) $|-|$ die Kardinalität denotiert.

$$\frac{\sum_{w \in L} |w|}{|L|} \tag{37}$$

Wir summieren also alle Längen von den verschiedenen Wörtern auf, und teilen sie durch die Anzahl der Wörter. Das gibt uns die durchschnittliche Länge der Worte in L, aber nicht die durchschnittliche Länge der Worte im Text, denn es beachtet nicht die unterschiedliche Wahrscheinlichkeit, mit der die einzelnen Worte im Text verteilt sind. Um die zu berücksichtigen, müssen wir $P(w)$ in unsere Formel einbauen:

$$\sum_{w \in L} |w| \cdot P(w) \tag{38}$$

Wir müssen in diesem Fall nicht mehr durch $|L|$ dividieren, da eine ähnliche Funktion bereits von $P(w)$ übernommen wird; denn $\sum_{w \in L} P(w) = 1$. Wie sie vielleicht schon erraten haben, ist $|-|$ eine Zufallsvariable, und die Formel in (38) ist nichts anderes als ihr Erwartungswert.

Hausaufgabe 3

1 Praxis Bitte bearbeiten bis zum 12.5.2020

In R gibt es die Funktionen mean() (Mittelwert bzw. Erwartungswert), var(), sd() (Varianz und Standardabweichung).

Implementieren Sie die Funktionen von Hand mittels elemenarer Operationen (Sie müssen sie entsprechend anders benennen.

Wundern Sie sich bitte nicht, wenn Sie leicht andere Ergebnisse bekommen bei var(),sd(), das passiert öfters!

2 Theorie Bitte bearbeiten bis zum 11.5.2020

Wir nehmen eine probabilistische Sprache: $P : \{a\}^+ \to [0, 1]$, wobei

$$P(w) = \frac{1}{2^{|w|}} \tag{39}$$

Sprich: $P(a) = 1/2$, $P(aa) = 1/4$, $P(aaa) = 1/8$ etc. Wir nehmen nun die Zufallsvariable $|-| : \Sigma^* \to \mathbb{R}$. Was ist ihr Erwartungswert? Also was ist

$$\sum_{w \in a^+} |w| \cdot P(w) \tag{40}$$

Natürlich ist die Summe unendlich, man kann das also nicht einfach ausrechnen. Aber Sie können das sehr gut approximieren, vielleicht finden Sie den Wert dabei. Viel Spass!

6.6 Varianz und Standardabweichung

Man muss sich darüber klar sein, dass der Erwartungswert nicht zwangsläufig ein Wert sein muss, der überhaupt vorkommt (ebenso wie etwa der Durchschnittswert). Wenn wir eine faire Münze haben,

$$X(K) = 1, X(Z) = -1,$$

dann ist

(41) $\mathbf{E}(X) = 0$

– also kein Wert, der irgendeinem Ergebnis entspricht. Es gibt noch einen weiteren Punkt, der sehr wichtig ist. Der Erwartungswert gibt uns eine Art Mittelwert im Hinblick auf die Wahrscheinlichkeit. Wir wissen aber nicht, wie die Ergebnisse um den Erwartungswertverteilt sind: sie können sich zum Erwartungswert hin häufen (sie das Beispiel der zwei Würfel); sie können sich aber auch auf beiden Seiten des Erwartungswertes häufen: siehe das letzte Beispiel der Münze.

Der Erwartungswert ist also für gewisse wichtige Fragen nicht informativ. Hier brauchen wir das Konzept der **Varianz**. Die Definition der Varianz einer Zufallsvariable ist zunächst nicht sehr erhellend:

(42) $\mathbb{V}(X) = \mathsf{E}((X - \mathsf{E}(X))^2)$

Was bedeutet diese Definition? Um sie zu verstehen, muss man zunächst wissen dass für zwei Zufallsvariablen X, Y, $X + Y$ und $X \cdot Y$, definiert durch

(43) $$X + Y(\omega) = X(\omega) + Y(\omega)$$

(44) $$X \cdot Y(\omega) = X(\omega) \cdot Y(\omega)$$

wiederum Zufallsvariablen sind. Also ist $X - \mathsf{E}(X)$ eine Zufallsvariable, und dann ebenso $(X - \mathsf{E}(X))^2$, und dementsprechend können wir wiederum deren Erwartungswert bestimmen. Die Zufallsvariable $X - \mathsf{E}(X)$ bildet ein Ergebnis ω auf die Differenz $X(\omega) - \mathsf{E}(X)$; es sagt uns also, wie weit ein Ergebnis von der Erwartung abweicht. Als nächstes wird dieses Ergebnis quadriert zu $(X(\omega) - \mathsf{E}(X))^2$, um alle Werte positiv zu machen (uns interessiert nur die Abweichung, nicht die Richtung der Abweichung). Wir haben also eine Zufallsvariable, die uns die Abweichung eines Ergebnisses vom Erwartungswert

von X im Quadrat liefert. Die Varianz ist schließlich der Erwartungswert dieser Variable. In einem Satz, die Varianz ist die erwartete Abweichung der Zufallsvariablen von ihrem Erwartungswert im Quadrat.

Die **Standardabweichung** $\sigma(X)$ einer Zufallsvariable X ist die Wurzel der Varianz:

$$\sigma(X) = \sqrt{\mathtt{V}(X)} \tag{45}$$

Die Standardabweichwung gibt grob gesagt die durchschnittliche Abweichung eines Ergebnisses (unter der Zufallsvariable) vom Erwartungswert (das stimmt nicht ganz, gibt aber eine Idee). Es gibt ein sehr wichtiges Ergebnis für die Standardabweichung, mittels dessen seine Bedeutung sofort klar wird:

Für eine Zufallsvariable X mit Erwartungswert $\mathsf{E}(X)$ und Standardabweichung σ gilt immer: für alle $t \in \mathbb{R}$ und die Wahrscheinlichkeitsmasse R, die zwischen $\mathsf{E}(X) - t\sigma$ und $\mathsf{E}(X) + t\sigma$ liegt, also

$$R = P(X \in [\mathsf{E}(X) - t\sigma, \mathsf{E}(X) + t\sigma]) \tag{46}$$

Dann gilt:

$$R \geq 1 - \frac{1}{t^2} \tag{47}$$

Z.B. $t = 2$ bedeutet, dass $3/4$ der Wahrscheinlichkeitsmasse in $[\mathsf{E}(X) - 2\sigma, \mathsf{E}(X) + 2\sigma]$. Also 2 Standardabweichungen decken $3/4$ der Wahrscheinlichkeit ab usw.

6.7 Varianz und Stichprobenvarianz

In einem Datensatz (array von Zahlen) wird der Erwartungswert zum Mittelwert, die Varianz wird zur mittleren quadratischen Abweichung vom Mittelwert. Es ist aber üblich, die Varianz in einem Datensatz ("Stichprobe") etwas anders zu definieren. Sie $\vec{x}$ ein Zahlenvektor,

- $|\vec{x}|$ seine Länge,
- $\sum \vec{x}$ die Summe seiner Komponenten
- $\vec{x}^2$ die *punktweise* Quadratoperation auf den Komponenten
- $\mu(\vec{x}) = \sum(\vec{x})/|\vec{x}$ sein Mittelwert

Dann haben wir, für die obige Definition der Varianz,

$$V(\vec{x}) = \mu((\vec{x} - \mu(\vec{x}))^2) = \frac{\sum((\vec{x} - \mu)^2)}{|\vec{x}|} \tag{48}$$

Die Stichprobenvarianz ist etwas anders definiert:

$$SV(\vec{x}) = \frac{\sum((\vec{x} - \mu)^2)}{|\vec{x}| - 1} \tag{49}$$

Wir subtrahieren also 1 im Nenner, sonst ist alles gleich. Man nennt das auch die **Bessel Korrektur**, und wenn z.B. R die Varianz eines arrays berechnet, benutzt es diese Formel. Bleibt die Frage: **Warum**?

Die Frage hat mit dem Konzept der **Erwartungstreue** zu tun: ein Erwartungswert ist der konvergierende Mittelwert aller Ergebnisse, wenn wir das Experiment (im limes) unendlich wiederholen. Die Varianz ist ebenfalls ein Erwartungswert, nämlich die erwartete Abweichung vom Erwartungswert. Das bedeutet:

> Um diesen Varianz-Erwartungswert überhaupt zu berechnen, *müssen wir den Erwartungswert bereits kennen.*

Um den Erwartungswert aber zu schätzen, brauchen wir bereits einen Datenpunkt. Und dieser Datenpunkt ist genau derjenige, der in der Bessel Korrektur im Nenner fehlt.

Um also die (Stichproben)Varianz als konvergierenden Mittelwert zu beschreiben, nehmen wir folgendes Experiment:

- Nimm einen Zahlenvektor $\vec{x} = (x_1, ..., x_n)$,
- ziehe zufällig ein x_i (entspricht Erwartungswert)
- ziehe zufällig ein x_j, wobei $i \neq j$ (entspricht Abweichung)
- berechne $(x_i - x_j)^2$

Hier ist klar, das es nur $n - 1$ Möglichkeiten gibt für x_j.

Das Experiment wird (im limes) unendlich iteriert, die Varianz wäre dann der konvergierende Mittelwert. Setze $x_i = x^m$ um zu sagen: das m-te Experiment zieht als erstes x_i; ebenso $x_j = y^m$. Dann haben wir (ohne dass es damit bewiesen wäre)

$$(50) \quad lim_{m\to\infty} \frac{\sum (x^m - y^m)^2}{m} = SV(\vec{x})$$

Hier konvergiert der Wert

$$(51) \quad lim_{m\to\infty} \frac{sumx_m}{m}$$

gegen den Erwartungswert, was zumindest erklärt, warum für die Varianz ein Datenpunkt immer "fehlt", obwohl immer neu gezogen wird.

6.8 Kovarianz und Korrelation

Die Varianz einer Variable ist die erwartete Abweichung von Erwartungswert. Die Kovarianz ist definiert für die *gemeinsame Varianz* zweier Variabeln.

$$\mathtt{V}(X) = \mathcal{E}((X - \mathcal{E}(X))^2) \tag{52}$$

$$Cov(X_1, X_2) = \mathcal{E}((X_1 - \mathcal{E}(X_1)) \cdot (X_2 - \mathcal{E}(X_2))) \tag{53}$$

Also anstatt die Abweichung vom Erwartungswert zu quadrieren werden die beiden Abweichungen multipliziert. Zuletzt wird der entsprechende Erwartungswert gebildet. Man beachte dass die Kovarianz durchaus negativ sein kein, im Gegensatz zur Varianz! Dabei gilt:

- Eine positive Kovarianz bedeutet: wenn die eine Variable positiv von ihrem Erwartungswert abweicht, dann auch die andere (tendenziell)
- Eine negative Kovarianz bedeutet: wenn die eine Variable positiv von ihrem Erwartungswert abweicht, dann die andere negativ, also X_1 groß$\sim$ X_2 klein (tendenziell)
- Eine Kovarianz um die 0 bedeutet: wir können keine Vorhersagen dieser Art treffen.

Die Kovarianz ist einfach, aber nicht besonders informativ: die konkreten Zahlen hängen von der Varianz der einzelnen Variablen ab, und erlauben keine Rückschlüsse darauf, wie stark die Variablen tatsächlich korrelieren. Dafür müssen normalisieren, also die individuelle Varianz herausdividieren. Pearsons Korrelation ist die Kovarianz der standardisierten Variablen, also

$$Korr(X_1, X_2) = \mathcal{E}((\hat{X_1} - \mathcal{E}(\hat{X_1})) \cdot (\hat{X_2} - \mathcal{E}(\hat{X_2}))) \tag{54}$$

Da – qua Konstruktion – $\mathcal{E}(\hat{X_1}) = \mathcal{E}(\hat{X_2}) = 0$, vereinfacht sich das nochmal zu

$$Korr(X_1, X_2) = \mathcal{E}(\hat{X_1} \cdot \hat{X_2}) = \frac{Cov(X_1, X_2)}{\sigma(X_1)\sigma(X_2)} \tag{55}$$

(die letzte Gleichung folgt aus Satz für die lineare Transformationen von Kovarianzen; Verschiebungen, also Addition/Substraktion, ändern die Kovarianz nicht).

Hausaufgabe ?

Abgabe bis 15.5.2018 vor dem Seminar.

Im Spielcasino gilt folgendes: Spiele entsprechen Zufallsexperimenten, Wetten entsprechen Zufallsvariablen. Der Ausspruch "Die Bank gewinnt immer" soviel wie: für jedes Spiel hat die Bank einen positiven Erwartungswert. Nun gibt es aber folgende Strategie: ich setze beim Roulette 1euro auf rot. Wenn ich gewinne, fange ich die Strategie von vorne an. Wenn ich verliere, setze ich den doppelten Betrag auf rot, immer weiter, bis ich irgendwann gewinne. Wenn ich schließlich gewinne, war mein Einsatz immer höher als meine Verluste bisher, ich mache also unterm Strich Gewinn.

1. Wie passt das zusammen damit, dass die Bank immer gewinnt? Wie entwickelt sich der Erwartungswert im Lauf des Spiels?

2. Unter welchen Annahmen könnten Sie tatsächlich das Casino sprengen (d.h. beliebig Gewinn machen)?

Hausaufgabe ?

Bitte bearbeiten bis zum 2.6.2020 Wir gehen ins Casino, und nehmen ein Beispiel aus der Sitzung: Roulette mit Farben und Ziffern.

1. Wir setzen Ziffern, die 1 sagen wir, und wir setzen einen Euro. Dann haben wir $P(1) = ... = P(37) = 1/37$. Die entsprechende Zufallsvariable X_1 verhält sich wie folgt: $X_1(1) = 35$ (Einsatz mal 36, minus Einsatz), $X_1(2) = ...X_2(37) = -1$ (Geld futsch).

2. Wir setzen Farben, rot sagen wir, ebenfalls einen Euro. Dann haben wir $P(r) = 18/37$, $P(\overline{r} = 19/37$. Die Zufallsvariable ist $X_2(r) = 1$ (Einsatz mal 2, minus Einsatz), $X_2(\overline{r}) = -1$ (Geld futsch).

Wir spielen jeweils 20 Runden, in denen wir jeweils einen Euro setzen, separat für jede der beiden Strategien.

1. Berechnen Sie die Erwartungswerte der beiden Strategien (für 20 Spiele auf Zahl, und für 20 Spiele auf Farbe).

2. Berechnen Sie Varianz und Standardabweichung.

7 RX2 Varianz etc. in R

7.1 Variablen in Datensätzen

Wie verhalten sich unsere Daten in languageR zu diesen Begriffen? Zunächst ist es wichtig, zu sehen wie die Spalten unserer Tabelle im Prinzip Zufallsvariablen sind:

	RealizationOfRec	VerbAnimacyOfRec	AnimacyOfTheme	LengthOfTheme
1	NP	feed animate	inanimate	2.6390573
2	NP	give animate	inanimate	1.0986123

Wir können diesen Datensatz als eine Art Wahrscheinlichkeitsverteilung auffassen:

- Unsere **Ergebnisse** haben die Form

  (NP, feed, animate, inanimate, 2.6390573)

  Wir haben also einen Produktraum $\Omega \subseteq \Omega_1 \times ... \times \Omega_5$! Diese Teilräume werden jeweils durch die möglichen Werte bestimmt, d.h. *levels* oder $\mathbb{R}$.

- Jedes Merkmal entspricht einer Zufallsvariable, d.h. einer Funktion $X_i : \Omega \to \Omega_i$. Derartige Konstrukte finden wir häufig!

- Z.B.: RealizationOfRec((NP, feed, animate, inanimate, 2.6390573) = NP

- Auf diese Art und Weise können wir den Raum transformieren, z.B. bedingte und unbedingte Verteilungen für gewisse Variablen betrachten, z.B.

  P(RealizationOfRec=NP)
  P(RealizationOfRec=NP | AnimacyOfRec=animate)
  etc.

Z.B. haben wir

(56)

$$\text{P(RealizationOfRec=NP)} = \frac{\texttt{nrow(verbs[verbs\$RealizationOfRecipient=="NP",])}}{\texttt{nrow(verbs)}}$$

Wir bekommen also die Wahrscheinlichkeiten aus der Annahme, dass der Raum Laplace ist, alle Zeilen gleich wahrscheinlich. Weiterhin bekommen wir:

$$= \frac{\text{P(RealizationOfRec=NP | AnimacyOfRec=animate)}}{\text{P(RealizationOfRec=NP \& AnimacyOfRec=animate)}}{\text{P(AnimacyOfRec=animate)}}$$

was wir nach obigen Schema einfach ausrechnen können.

Wir sehen hieran warum Zufallsvariablen ein so wichtiges Werkzeug sind: sie erlauben uns fragen wie oben sehr kompakt schreiben, auch wenn sie sich natürlich ohne "Variablen" als Ereignisse darstellbar wären.

7.2 Verteilungen in Datensätzen

Eine besondere Variable ist dabei LengthOfTheme, da sie eine Funktion nach $\mathbb{R}$ ist. Nennen wir diese Variable LOT. Wie ist die Verteilung von LOT definiert? Sei r eine reelle Zahl. Dann bekommen wir

(57) $P(LOT = r) = n(r)/|verbs|$

wobei $|verbs|$ die Anzahl der Einträge in verbs ist (=Zeilen), und $n(r)$ die Anzahl der Einträge $\vec{x}$ so dass $LOT(\vec{x}) = r$. Wir können diese Variable also tatsächlich als eine (diskrete) **Wahrscheinlichkeitsverteilung**

$$f_X : \mathbb{R} \to \mathbb{R}$$

auffassen, denn es gilt

(58) $$\sum_{r \in \mathbb{R}} P(LOT = r) = 1$$

Dementsprechend kann man nun fragen: wie ist der Erwartungswert dieser Variable, wie die Varianz und die Standardabweichung? Das sind wichtige Kenngrößen einer Verteilung, die uns Aufschluss geben über ihre Natur. Wie können wir das berechnen?

Hierfür gibt es eine Reihe eingebauter Befehle in R, die jeweils einen numerischen Vektor als Eingabe nehmen, nämlich:

- **mean()** – gibt den Mittelwert des Vektors, was in diesem Fall der Erwartungswert der Variable ist!
- **median()** – gibt den sog. Median des Vektors, also den Wert der in der Mitte liegt (genausoviele darüber wie darunter)
- **range()** liefert die Weite der Verteilung, also den maximalen und minimalen Wert den sie annimmt (nicht die Wahrscheinlichkeit, sondern das kleinste/größte r so dass $P(X = r) > 0$!
- **var()** gibt die Varianz der Variable
- **sd()** gibt die Standardabweichung

Das sieht also wie folgt aus:

> var(verbs$LengthOfTheme)

> sd(verbs$LengthOfTheme)

Für $r \in [0, 1]$ ist dass r-Quantil besagt nun derjenige Wert in $x \in X$ so dass $r * 100$ Prozent des Datensatzes kleiner ist als x. Also das 0.9-Quantil ist derjenige Wert, der größer ist als 90 Prozent des Daensatzes. Hierfür gibt es einen Befehl **quantile()** in R:

> quantile(verbs$LengthOfTheme,0.5)

> quantile(verbs$LengthOfTheme,0.9)

Die Standardabweichung ist, wie oben gesagt, eng verknüpft mit dem Begriff des **Quantils**: $t\times$ Standardabweichung deckt $1 - \frac{1}{t^2}$ der Wahrscheinlichkeitsmasse ab.

Block-Aufgabe 4

1. Implementieren Sie eine Funktion Varianz() mit elementarer Arithmetik und Kontrollstrukturen, die für einen gegeben Zahlenvektor die Varianz berechnet! (sie dürfen aber benutzen Funktionen wie length(), nrow(), sum()).

2. Ebenso mit Standardabweichung!

3. Implementieren Sie eine Funktion Zentral(Vektor,r) mit Vektor einem Zahlenvektor, $r \in [0, 1]$, die Ihnen zwei Werte liefert, nämlich folgende: den Wert $x \in X$ so dass anteilig r/2 Einträge kleiner sind als x, und den Wert $y \in X$ so dass (anteilig) r/2 Einträge größer sind als y.

8 Wichtige Wahrscheinlichkeitsverteilungen

8.1 Einleitung

Im letzten Beispiel war unser Wahrscheinlichkeitsraum der Raum zweier Würfe mit einem fairen Würfel. Wir haben gesehen dass die Zufallsvariable $X : \Omega \to \mathbb{R}$,

(59) $X(\langle i, j\rangle) = i + j$

aus einem Wahrscheinlichkeitsraum $\mathcal{P}_1 = (\Omega, \wp(\Omega), P)$ einen neuen Wahrscheinlichkeitsraum macht, nämlich den Raum

$$\mathcal{P}_2 = (X[\Omega], \wp(X[\Omega]), P \circ X^{-1})$$

Beide Räume sind diskret, aber der Raum $\mathcal{P}_1$ hat eine wichtige Eigenschaft, die $\mathcal{P}_2$ fehlt: er ist Laplace, d.h. alle Ergebnisse sind gleich wahrscheinlich. $\mathcal{P}_2$ ist natürlich nicht Laplace; dennoch sieht man ihm auf gewisse Weise an, dass er aus einem Laplace-Raum entstanden ist. Wir werden uns zunächst mit den sog. Binomialverteilungen beschäftigen.

Definition 4 *Binomialverteilungen sind Verteilungen, die aus einem (vielfachen) Produkt eines Bernoulli-Raums mit sich selbst konstruiert werden mittels einer additiven Zufallsvariable wie in (59).*

Danach werden wir uns den allgemeineren Multinomialverteilungen zuwenden, für die unser Würfelraum ein Beispiel liefert.

Wir haben gesagt dass Zufallsvariablen Funktionen in die rellen Zahlen sind. Eine wichtige Konsequenz ist, dass wir, gegeben einen Wahrscheinlichkeitsraum mit Wahrscheinlichkeitsfunktion P und eine Zufallsvariable X, eine Funktion

$$f_X : \mathbb{R} \to \mathbb{R}$$

bekommen, die definiert ist durch

(60) $f_X(x) = P(X = x) = P(X^{-1}(x))$

(letztere Gleichung qua unserer Konvention; verwechseln Sie nicht das große X und das kleine x!). Diese Funktion ist die **Wahrscheinlichkeitsverteilung** von X. NB: die Wahrscheinlichkeitsverteilung ist eindeutig definiert durch

den Wahrscheinlichkeitsraum und die Zufallsvariable. Deswegen wird oft von Wahrscheinlichkeitsfunktionen P gesprochen als wären sie eine Wahrscheinlichkeitsverteilungen, und umgekehrt. Das kann manchmal zu Verwirrung führen, denn es ist ja nicht gesagt dass die Ergebnisse in Ω reelle Zahlen sind, und daher kann man von keiner Verteilung für P selbst sprechen. Falls aber $\Omega \subseteq \mathbb{R}$, dann ist die Identitätsfunktion id, wobei

$$\text{f.a. } x \in \mathbb{R},\ id(x) = x,$$

eine Zufallsvariable. Und da

$$P \circ id = P \circ id^{-1} = P,$$

kann man auch von einer Wahrscheinlichkeitsverteilung von P sprechen.

Eine Wahrscheinlichkeitsverteilung f_X heißt **diskret**, wenn es nur endlich oder abzählbar unendlich viele $x \in \mathbb{R}$ gibt, so dass $f_X(x) \neq 0$ (erinnern Sie sich: falls $X^{-1}(x) = \emptyset$, dann ist $P(X^{-1}(x)) = 0$, also $f_X(x) = 0$.

8.2 n über k

Die Formel $\binom{n}{k}$ (sprich n über k) ist von zentraler Bedeutung für die Wahrscheinlichkeitstheorie und Statistik. Sie ist wie folgt definiert:

(61)
$$\binom{n}{k} = \frac{n}{1} \cdot \frac{n-1}{2} \cdot \ldots \cdot \frac{n-(k-1)}{k} = \frac{n \cdot (n-1) \cdot \ldots \cdot (n-k+1)}{k!} = \frac{n!}{k!(n-k)!}$$

Die letze Gleichung gilt nur unter der Voraussetzung dass n, k positive ganze Zahlen sind, und $n \geq k$. In unseren Beispielen wird diese Voraussetzung immer erfüllt sein. Die intuitive Bedeutung dieser Formel ist die folgende:

- nehmen wir an, wir haben eine Menge M, so dass $|M| = n$.
- $\binom{n}{k}$ ist die Anzahl von verschiedenen Mengen $N \subseteq M$, so dass $|N| = k$.

Warum brauchen wir diese Formel?

- Nehmen wir einen Raum, der das n-fache Produkt eines Wahrscheinlichkeitsraumes darstellt; etwa: ein n-facher Münzwurf.
- Wir möchten nun die Wahrscheinlichkeit des Ereignisses: k-mal Kopf.
- Dieses Ereignis umfasst alle Ergebnisse (Ergebnisse sind n-tupel), von denen k-Komponenten Kopf sind. Wieviele Ereignisse sind das?
- Die Antwort ist $\binom{n}{k}$.

Diese Formel ist also sehr wichtig um Wahrscheinlichkeiten von Ereignissen der Art zu berechnen: k von n Ergebnissen sind x (x irgendein Ergebnis), egal welche.

8.3 Binomiale Verteilungen

Zur Erinnerung: ein Bernoulli-Raum ist ein Wahrscheinlichkeitsraum mit $|\Omega| = 2$. Wir setzen kanonisch

1. $\Omega = \{0, 1\}$ (denn die Bezeichnung der Ereignisse ist natürlich willkürlich); außerdem
2. $p = P(1)$, $q = (1 - p)$

Nehmen wir Einfachheit halber an, dass $\mathcal{P}$ Bernoulli und Laplace ist, z.B. der Raum zum Wurf einer fairen Münze. Wir denotieren das Ereignis "Kopf" mit 0, "Zahl" mit 1. Da also unsere Ereignisse reelle Zahlen sind, nehmen wir kurzerhand die Zufallsvariable id, d.i. die Identitätsfunktion. Wir erweitern jetzt den Raum zu einem n-fachen Produktraum, d.h. zu dem Raum eines n-fachen Münzwurfes; und wir nehmen eine Zufallsvariable

$$X : \{0, 1\}^n \to \mathbb{R},$$

so dass

$$X(\langle \omega_1, .., \omega_n \rangle) = \textstyle\sum_{i=1}^{n} \omega_i;$$

d.h. nichts anderes als dass uns X für irgendein Ergebnis sagt wie oft wir Zahl geworfen haben, unabhängig von der Reihenfolge der Ergebnisse.

Wir wissen bereits, wie wir die Wahrscheinlichkeit für das Ereignis ausrechnen, dass wir von den n Würfen k-mal Zahl werfen; beachten Sie, dass in der neuen Terminologie wir dieses Ereignis mit $X^{-1}(k)$ bezeichnen können!

$$P(X^{-1}(k)) = \binom{n}{k} p^k (1 - p)^{n-k} \tag{62}$$

(p ist die Wahrscheinlichkeit von Zahl, $1 - p$ die Wahrscheinlichkeit von Kopf.) Wenn wir nun die Wahrscheinlichkeitsverteilung haben wollen für das n-fache Produkt des Bernoulli-Raumes und unserer Variable X, dann kriegen wir folgende Funktion $f : \mathbb{R} \to \mathbb{R}$:

$$f_X(x) = \begin{cases} \binom{n}{x} p^x (1 - p)^{n-x}, \text{ falls } x \in \{0, 1, ..., n\} \\ 0 \text{ andernfalls} \end{cases} \tag{63}$$

Dies ist die Formel für die sogenannte **Binomialverteilung**, die wohl wichtigste diskrete Wahrscheinlichkeitsverteilung. Diese Verteilung ist symmetrisch genau dann wenn $p = 0.5$, $p = 1$ oder $p = 0$. In beiden letzten Fällen gibt die Funktion für alle Eingaben bis auf eine 0 aus, wie Sie leicht prüfen können. In allen anderen Fällen ist die Funktion asymmetrisch.

Die Binomialverteilung, wie wir sie geschrieben haben, ist eine Funktion, d.i. eine Abbildung von reellen Zahlen in die reellen Zahlen. Eigentlich handelt es sich aber um eine *Familie* von Funktionen, da wir für p und n uns nicht allgemein festlegen müssen (aber mit p auch q festlegen!). Die Funktion ändert sich aber je nach den Werten die p und q nehmen, daher sagt man p und q sind die **Parameter** der Funktion. Wir schreiben also die Familie der Binomialverteilungen als

$$\mathbf{B}(x|p,n) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x}, \text{ falls } x \in \{0,1,...,n\} \\ 0 \text{ andernfalls} \end{cases} \tag{64}$$

Hier können wir p, n entweder als zusätzliche Argumente der Funktion betrachten, oder als konkrete Instanziierungen für ein Element der Familie von Funktionen. Wichtig ist aber dass

1. $0 \leq p \leq 1$, und
2. $n \in \mathbb{N}$,

sonst ist die Funktion (bis auf weiteres) nicht definiert. Wir haben folgende Konvention: wir sagen Binomialverteilung, wenn wir die ganze Familie von Funktionen meinen, und Binomialfunktion, wenn wir eine konkrete Funktion betrachten. Eine wichtige Eigenschaft der Binomialverteilung ist die folgende:

Lemma 5 *Der Erwartungswert einer Binomialfunktion gilt immer* $\mathbf{E}(\boldsymbol{B}(x|p,n)) = pn$.
Die Varianz einer Binomialverteilung ist immer $\mathbf{V}(\boldsymbol{B}(x|p,n)) = p(1-p)n$

Den Beweis lasse ich an dieser Stelle aus, da er an vielen Stellen nachgelesen werden kann. Ein berühmter und wichtiger Satz ist der Satz von Moivre-Laplace, der besagt dass für $n \to \infty$ (also für immer öfter Würfeln) die Binomialverteilung gegen die Gauss'sche Normalverteilung konvergiert.

In R gibt es die Binomialverteilung natürlich eingebaut, mit dem Befehl **dbinom()**, mit drei Argumenten: x (Anzahl ein Ergebnis) n (Anzahl Iterationen), p (Wahrscheinlichkeit des Ergebnisses)

> dbinom(23,50,0.5)

8.4 Kategoriale Wahrscheinlichkeitsräume und Multinomiale Verteilungen

Die Generalisierung von $|\Omega| = 2$ auf $|\Omega| = n : n \in \mathbb{N}$, also von Bernoulli-Räumen auf beliebige endliche Räume, sind *kategoriale* Räume und Wahrscheinlichkeitsfunktionen. Ebenso wie Binomialverteilungen aus der Iteration von Bernoulli-Räumen entstehen (d.h. durch ein endliches Produkt eines Bernoulli Raumes $\mathcal{P}$ mit sich selbst, auch $\mathcal{P}^k$ geschrieben), entstehen **Multinomialverteilungen** durch ein endliches Produkt eines kategorialen Raumes mit sich selbst. Multinomialverteilungen sind komplizierter als Binomialverteilungen aus folgendem Grund: nehmen wir an, $|\Omega| = n$, und als Konvention

$$\Omega = \{0, 1, ..., n-1\}.$$

Wir notieren

$$P(i) = p_i.$$

Für die Multinomialverteilung ist nun jedes

$$p_i : 0 \leq i \leq n-1$$

ein Parameter. Auch die Kombinatorik dieser Räume ist wesentlich komplizierter, weswegen es (meines Wissens nach) keine geschlossene Formel für Multinomialfunktionen gibt. Wichtig ist folgendes: sei X eine Zufallsvariable auf Ω, dem einfachen Raum; X_n die Zufallsvariable definiert durch

$$X_n(\langle \omega_1, .., \omega_n \rangle) = \sum_{i=1}^{n} X(\omega_i) \tag{65}$$

Also wieder die alte Summenvariable. Diese Variable ist multinomial verteilt. Nun gilt:

$$\mathcal{E}(X_n) = \mathcal{E}(X) \cdot n \tag{66}$$

$$\mathtt{V}(X_n) = \mathtt{V}(X) \cdot n \tag{67}$$

$$\sigma(X_n) = \sqrt{\mathtt{V}(X_n)} \tag{68}$$

Wir können also die Verteilung selbst vielleicht nicht sehr gut berechnen, aber diese Parameter sind sehr leicht zu bekommen, da X normalerweise eine einfache Verteilung ist. Das ist extrem wichtig für die Approximation mittels der Normalverteilung!

Im Grenzwert (für $n \to \infty$) konvergiert aber auch die Multinomialverteilung auf die Gauss'sche Normalverteilung. Das ist eine Folge des Zentralen Grenzwertsatzes, der wiederum eine Generalisierung des Satzes von Moivre-Laplace darstellt. Das bedeutet also: wenn wir mit n Würfeln spielen und die Verteilung für die Summe der Augen suchen, dann wird diese Verteilung immer ähnlicher der Normalverteilung, je größer n ist. Das zeigt Ihnen auch, wie außerordentlich wichtig die Normalverteilung ist für Stochastik und Statistik - auch wenn Sie sie noch nicht kennen.

Für die Multinomialverteilung gibt es keine geschlossene analytische Form, dementsprechend gibt es keine Funktion in R. Wenn wir also wissen wollen: wie wahrscheinlich ist es, mit 100 Würfen 436 Augen zu werfen, dann müssen wir erstmal rechnen!

8.5 Normal-Verteilungen und der Zentrale Grenzwertsatz

Wir werden Normalverteilungen nur sehr kurz anreißen, weil deren Funktion ziemlich kompliziert ist, und sie in der statistischen Sprachverarbeitung keine herausragende Rolle spielen. Wenn wir sie dennoch kurz besprechen, liegt das an der herausragenden Rolle die sie in der gesamten Statistik spielen, und insbesondere ihrer Bedeutung für die beiden zuletzt besprochenen Binomial- und Multinomialverteilungen. Die Normalverteilung ist eine Familie von Funktionen mit zwei Parametern, dem **Mittelwert** μ und der **Standardabweichung** σ; deren Formel ist

$$(69) \quad f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2\right)$$

Die Normalverteilung ist eine stetige Funktion über reelle Zahlen, im Gegensatz zu den anderen Verteilungen die wir hier betrachten. Ich werde diese Funktion hier nicht erklären, aber es ist wichtig zu wissen dass die Normalverteilung *die* statistische Verteilung schlechthin ist. Ihre Bedeutung versteht man vielleicht am besten aus dem **zentralen Grenzwertsatz**, den ich hier auch nur informell beschreibe: nehmen wir an, wir haben einen Wahrscheinlichkeitsraum, über dem wir n unabhängige, gleich verteilte Zufallsvariablen definieren können (z.B. n-mal Münze werden/würfeln etc., wobei jede Zufallsvariable X_i uns das Ergebnis des i-ten Wurfes liefert). Wir nennen wir diese Zufallsvariablen also

$$X_i : 1 \leq i \leq n.$$

Wir definieren nun eine neue Zufallsvariable

$$(70) \quad Y = X_1 + X_2 + ... + X_n$$

(erinnern Sie sich wie die Addition von Funktionen definiert ist: $f + g(x) := f(x) + g(x)$.

Der zentrale Grenzwertsatz Der zentrale Grenzwertsatz besagt: je größer n ist, desto stärker gleicht sich Y an die Normalverteilung an.

Das ist aus mindestens zwei Gründen wichtig: 1. die Binomialfunktion ist für große n gar nicht mehr berechenbar; wir können sie aber, je größer n, desto genauer mit der Normalverteilung approximieren. 2. Fehler in komplizierten Messungen oder Berechnungen, oder allgemeiner gesagt: Reihen von zufälligen Prozessen, verhalten sich genau so wie unsere Multinomialverteilungen; sie können also durch die Normalverteilung modelliert werden. Insbesondere bedeutet das: Reihen von Meßfehlern (etwa in der Physik, Astronomie) summieren sich *nicht* auf!

In R kann man die Normalverteilung abrufen mit

```
> dnorm(1,0,1)
```

oder plotten mit

```
> f<- function(x)dnorm(x,0,1)
> plot(f,-5,5)
```

8.6 Eine Anwendung des zentralen Grenzwertsatzes

Wie ist die Wahrscheinlichkeit, mit 100 Würfen mit einem fairen Würfel 370 Augen zu werfen? Naja, theoretisch können wir das einfach berechnen:

1. Jede Sequenz von 100 Würfen hat die gleiche Wahrscheinlichkeit $1/6^{100}$
2. Jetzt müssen wir nur noch ausrechnen, wie viele Folgen von Würfen der Länge 100 es gibt, deren Augen sich auf 370 aufaddieren.

Punkt 2 ist aber problematisch: es sind sehr viele, zuviele um sie einfach mit der Hand aufzuschreiben – wir werden nachher sehen wieviele (ich glaube mehr als es Teilchen in unserem Universum gibt, wenn ich richtig schätze). Andererseits haben wir auch keine Formel zur Hand, die uns das Ergebnis liefert!

Man kann das Problem mit kombinatorischen und analytischen Mitteln etwas runterbrechen, aber für mich sieht das aus wie ein Tag voll Arbeit.

Was machen wir also? Wir nutzen den zentralen Grenzwertsatz. Iterierte, unabhñagige Laplace-Experimente liefern uns eine Normalverteilung, 100 ist bereits eine vernünftig große Zahl, also können wir unser Ergebnis mit der Normalverteilung approximieren. Die Normalverteilung hat zwei Parameter, Erwartungswert und Standardabweichung. Die müssen wir nun für unsere Multinomialverteilung berechnen.

Erwartungswert Der Erwartungswert für einen Würfelwurf liegt bei

$$\frac{1+2+3+4+5+6}{6} = 3.5 \tag{71}$$

Das ist der Wert, den wir bei einem Wurf (im Mittel) erwarten. Wenn wir nun hundert Mal werfen, erwarten wir in jedem Wurf 3.5, wir bekommen also einen Erwartungswert von $3.5 \cdot 100 = 350$. Das ist genau wie bei der Binomialverteilung!

Varianz und Standardabweichung Die Varianz ist die erwartete quadratische Abweichung vom Erwartungswert. In unserem Fall berechnen wir, für X_n die Variable von n Würfen,

(72)

$$\mathtt{V}(X_1) = \frac{(1-3.5)^2 + (2-3.5)^2 + (3-3.5)^2 + (4-3.5)^2 + (5-3.5)^2 + (6-3.5)^2}{6} = 2.916667$$

Da die Experimente unabhängig sind und wir mit jedem Wurf eine solche Abweichung erwarten, gilt auch hier: die Varianz bei 100 iterierten und unabhängigen Würfen ist die Summe der Einzelvarianzen, also:

(73) $\mathtt{V}(X_{100}) = \mathtt{V}(X_1) \cdot 100 = 291.6667$

Dementsprechend bekommen wir eine Standardabweichung

(74) $\sigma(X_{100}) = \mathtt{V}(X_{100})^{1/2} = 17.07825$

Jetzt haben wir alles zusammen: wir berechen einfach

(75) $dnorm(370|350, 17.07825) = 0.01176697$

Ebenso z.B.:

(76) $dnorm(425|350, 17.07825) = 1.515746 \cdot 10^{-6}$

Das sind wahrscheinlich nicht die korrekten Wahrscheinlichkeiten, aber ich gehe davon aus dass die korrekten Wahrscheinlichkeiten sehr sehr nahe liegen!

Wir konnten uns also einen Tag voll kombinatorischer und analytischer Arbeit sparen. Danke, Normalverteilung! Wir können nun auch ganz einfach ausrechnen, wie viele Möglichkeiten es z.B. gibt, 370 Augen mit 100 Würfen zu werfen. Setze

(77)
$C(100, 370) =$ Anzahl der Möglichkeiten 370 Augen mit 100 Würfen zu werfen

Wir wissen dass

(78) $dnorm(370|350, 17.07825) = 0.01176697 \approx C(100, 370) \cdot (\frac{1}{6})^{100}$

Damit bekommen wir also:

(79) $C(100, 370) \approx \frac{0.01176697}{(1/6)^{100}} \approx 7.687581 \cdot 10^{75}$

Also eine 75 stellige Zahl!!!

Übrigens, damit wir sehen wie erstaunlich diese Tatsache ist (dass das funktioniert), bedenken wir folgendes: die Normalverteilung (egal mit welchen Parametern) ist eine stetige, differenzierbare Funktion. Damit eine stetige differenzierbare Funktion eine Wahrscheinlichkeitsverteilung ist, muss sie folgendes erfüllen:

$$\int_{-\infty}^{\infty} dnorm(x|\mu,\sigma)dx = 1 \tag{80}$$

Wir haben also ein unbestimmtes Integral von 1. Die Normalverteilung erfüllt das natürlich. Hierbei gilt es zu beachten dass die Normalverteilung (wieder egal mit welchen Parametern niemals den Wert 0 annimmt: sie konvergiert (abhängig von σ) relativ rasch gegen 0, wenn wir vom Erwartungswert abweichen (egal in welche Richtung), aber sie nimmt den Wert niemals an.

Soviel dazu; jetzt kehren wir wieder zurück zu unserer Multinomialverteilung, welche entsteht durch 100 Würfe mit einem fairen Würfel. Die möglichen Ergebnisse sind alle Zahlen von

100 bis 600

Wir brauchen also, unabhängig von den konkreten Wahrscheinlichkeiten, folgendes:

$$\sum_{n=100}^{600} dnorm(n|350, 17.07825) = 1 \tag{81}$$

Einfach damit wir die Multinomialverteilung konsistent approximieren. Wie sich (mit einem kleinen Programm) leicht überprüfen lässt, erfüllt *dnorm* die Gleichung 81. Die Normalverteilung ist also wirklich ein kleines Wunder der Mathematik (entdeckt durch C.F. Gauss).

8.7 Einige Illustrationen zu Binomial- und Normalverteilung

So berechnet man einfache Wahrscheinlichkeiten:
> `dbinom(1,4,0.5)`
> `0.25`

Man kann schön illustrieren wie sich die relative Varianz verringert:

```
f1 <- function(x){dbinom(x,10,0.5)}
plot(f1,0,10)

f2 <- function(x){dbinom(x,100,0.5)}
plot(f2,0,100)

f3 <- function(x){dbinom(x,1000,0.5)}
plot(f3,0,1000)
```

Aber die absolute Varianz erhht sich (siehe oben, wächst linear im zweiten Parameter)!

```
f1(5)
f2(50)
f3(500)
plot(f3,450,550)
```

Übrigens, eine Darstellung einer asymmetrischen Binomialverteilung:

```
f4 <- function(x){dbinom(x,100,0.7)}
plot(f4,0,100)
```

Ähnliche Spielereien kann man natürlich auch mit der Normalverteilung machen:

```
g1 <- function(x){dnorm(x,0,1)}
```

8.8 Die geometrische Verteilung und die Exponentialverteilung

Es gibt folgende Frage, die zur der geometrischen Verteilung führt:

- Gegeben ein (iteriertes) Bernoulli-Experiment, wie ist die Wahrscheinlichkeit genau $n-1$ Iterationen vor einem Erfolg zu haben?

Diese Frage ist ebenfalls wichtig für die Computerlinguistik, denn es ist die äquivalente Frage:

- Gegeben eine probabilistische Sprache, wie ist die Wahrscheinlichkeit, dass der n-te Buchstabe ein bestimmter Buchstabe a ist?

Die Antwort lautet in beiden Fällen gleich; sei p die Wahrscheinlichkeit unseres "ausgezeichneten" Ereignisses; dann ist die Wahrscheinlichkeit

$$P(X = n) = p(1-p)^{n-1} \tag{82}$$

Das bildet eine sog. **geometrische Folge** (für n), was soviel bedeutet dass der Quotient für zwei benachbarte Folgeglieder konstant ist durch die Kette.

Diese Variable hat folgenden Erwartungswert:

$$\mathcal{E}(X) = \frac{1}{p} \tag{83}$$

Die Varianz wird dann

$$\mathtt{V}(X) = \frac{1-p}{p^2} = \frac{1}{p^2} - \frac{1}{p} \tag{84}$$

Eine wichtige, sie eindeutig bestimmende Eigenschaft ist die sog. **Gedächtnislosigkeit**:

$$P(X = n + k | X > n) = P(X = k) \tag{85}$$

wobei n, k beliebige natürliche Zahlen sind. D.h. sie ist invariant für jegliche Verschiebungen. Die geometrische Verteilung spielt eine wichtige Rolle bei Markov-Ketten.

Wie die Normalverteilung als stetige Funktion korrespondiert mit der Binomialverteilung, so korrespondiert die **Exponentialverteilung** mit der geometrischen Verteilung.

$$f_\lambda(x) = \begin{cases} \frac{1}{\mu} e^{-\lambda x} \text{ falls } x \geq 0 \\ 0 \text{ falls } x < 0 \end{cases} \tag{86}$$

Diese Funktion hat einen Parameter λ, der bestimmt wie steil die Funktion abfällt.

Diese Funktionen kann man in R aufrufen mit folgenden Befehlen; die geometrische Verteilung hat den Befehl

> dgeom(x, prob)

wobei prob der Parameter p ist. Wir haben die üblichen Variantion $pgeom(), qgeom()$. Parallel die exponentielle Verteilung:

> dexp(x, rate = lambda)

Hier ist lambda der Parameter der Verteilung; der default ist 1; ebenso $pexp(), qexp()$.

8.9 Die hypergeometrische Verteilung

Eine (diskrete) Variable X ist hypergeometrisch verteilt, falls gilt: es gibt M, N, n, so dass

$$P(X = k) = \frac{\binom{M}{k}\binom{N-M}{n-k}}{\binom{N}{n}} \tag{87}$$

Diese Verteilung gilt für Stichproben, bei denen die Proben nicht zurückgelegt werden. Z.B.:

- von N Kugeln in einer Urne
- sind M rot (also $N \geq M$).
- Dann werden n Kugeln gezogen (also nicht wieder zurückgelegt!).
- Die hypergeometrische Verteilung liefert die Wahrscheinlichkeit, dass k ($\leq n$) rote Kugeln gezogen werden.

Das ist natürlich verwandt mit der Binomialverteilung: auch hier wird ein Bernoulli-Experiment n iteriert, und wir suchen die Wahrscheinlichkeit dass wir k mal ein bestimmtes Ergebnis finden. Die hypergeometrische Verteilung unterscheidet sich dadurch, dass jedes Ergebnis die Verteilung ändert: falls wir eine rote Kugel ziehen, sinkt die Wahrscheinlichkeit als nächstes wieder eine rote zu ziehen, da der Anteil der roten Kugeln sinkt; ebenso für die nichtroten Kugeln. In der Binomialverteilung bleiben die Wahrscheinlichkeiten immer gleich!

Damit lässt sich auch ein (komplexer) mathematischer Zusammenhang erklären: wenn wir N gegen ∞ konvergieren lassen, wobei

- M proportional mitwächst
- aber n, k gleich bleiben,

dann konvergiert die hypergeometrische Verteilung gegen die Binomialverteilung. Das ist weil in diesem Fall der Unterschied, den das ziehen einer roten/nichtroten Kugel für die darauffolgende Verteilung hat, gegen 0 konvergiert; wir bekommen also (im unendlichen) ein iteriertes Bernoulli Experiment. Aber Vorsicht: falls $N = \infty$, dann gelten eigene und andere Gesetze – wir erinnern uns an die Probleme des Unendlichen!

Die hypergeometrische Verteilung in R Der Grundbefehl für diese Verteilung lautet `hyper`; es gibt also `dhyper`, `phyper`,`qhyper`. Wir betrachten hier nur `dhyper`; das berechnet genau die Funktion in 87. Wir haben dann

```
> dhyper(k, M, N-M, n)
```

Also: k mal ziehen, M rote Kugeln, N Kugeln insgesamt (also N-M nicht rote), und wir bekommen die Wahrscheinlichkeit, x rote Kugeln zu ziehen.

8.10 Potenzgesetze

Bisher haben wir von Verteilungen (realwertigen Funktionsgraphen) gesprochen, die von gewissen Wahrscheinlichkeitsräumen und Zufallsvariablen induziert werden. Wir können aber auch aus einer anderen Perspektive von Verteilungen sprechen: nämlich als der Verteilung von gewissen Daten, die wir tatsächlich in der Realität beobachtet haben. In diesem Fall kennen wir die Werte (natürlich nur endlich viele), aber wissen nichts über die zugrundeliegenden Wahrscheinlichkeiten. Die erste Perspektive ist die Perspektive der Stochastik, die letztere ist die Perspektive der Statistik. Die **Zipf-Verteilung** ist sehr wichtig für die Linguistik, weil wir sie sehr häufig beobachten; aus diesem Grund werden wir jetzt die statistische Perspektive einnehmen.

Nehmen wir an, wir haben einen Datensatz, der aus Paaren von reellen Zahlen besteht, oder sich daraufhin auflösen lässt. Paare von reellen Zahlen sind deswegen so wichtig, weil Funktionen *extensional* betrachtet nichts anderes sind als Paare von Zahlen

$$(x, f(x)).$$

Man nennt diese Zahlenpaare auch den **Graphen** von f.

Beispiel: types und token bei Goethe Nehmen wir beispielsweise eine Menge von Wörtern, wie sie in einem Text vorkommen (z.B. *Die Wahlverwandtschaften*). Eine wichtige Unterscheidung, an die wir uns zunächst gewöhnen müssen, ist die von *type* und *token*. Als *type* bezeichnet man ein Wort als abstraktes Objekt (aber durchaus die konkrete Form, also nicht das Lemma/Lexem!). Als *token* bezeichnet man jedes Vorkommen dieses Objektes. Wenn ich also in einem Abschnitt zweimal das Wort isst finde, dann ist es derselbe *type*, aber zwei verschiedene *token*. Uns interessieren zunächst die *types*, die in dem Text vorkommen. Das sind keine Zahlenpaare; aber wir ordnen jedem Wort (*type*) ein Zahlenpaar zu: die erste Zahl gibt an, das wievielte Wort es ist in einer Liste, in der alle Worte (*types*) unseres Textes nach ihrer Häufigkeit (Anzahl der *tokens*) im Text geordnet sind, also etwa 1 wenn es das häufigste Wort ist. Die zweite Zahl gibt an, wie viele *token* von diesem Type es in unserem Text gibt. Die erste Zahl nennen wir den *Rang* des Wortes, die zweite die Häufigkeit.

Wir bekommen also eine Tabelle (Fantasiezahlen und -worte!)

Rang: $r(w)$	Häufigkeit: $f(w)$	(Wort)
1	10,362	(der)
2	6,295	(das)
3	3,981	(ist)
...	...	

Die dritte Spalte lassen wir weg.
Wir haben also einen Datensatz

$$D \subseteq \mathbb{R} \times \mathbb{R}$$

aus Paaren von Zahlen (die Worte selbst kommen nicht mehr vor).

Wir nehmen nun an, diese Paare sind eine Teilmenge des Graphen einer Funktion; aber wir wissen natürlich nicht welche! Unsere Aufgabe ist es nun, eine Funktion zu finden, die gute Eigenschaften hat (z.B. einfach ist), aber dennoch unsere Daten gut *approximiert*. Potenzgesetze findet man dann, wenn es eine Polynomfunktion gibt, die unsere Daten beschreibt.

Wir sagen also der Datensatz D folgt einem **Potenzgesetz**, wenn es ein Polynom

$$a_1 \cdot x^b + a_2 \cdot x^{b-1} + ...$$

gibt, so dass für alle $(x, y) \in D$,

(88) $y \approx a_1 \cdot x^b + a_2 \cdot x^{b-1} + ... + a_b$,

wobei $\approx$ eine näherungsweise Gleichheit bedeutet. Wichtig für das Polynom ist dass b der größte Exponent ist; alle Terme bis auf $a_1 \cdot x^b$ werden dann weggelassen, und man schreibt:

(89) $y \propto a \cdot x^b$,

was bedeutet dass die beiden miteinander korrelieren. Der Datensatz, den wir betrachtet haben, folgt tatsächlich einem Potenzgesetz, und noch genauer gesagt einer Zipf-Verteilung.

8.11 Zipfs Gesetz

In unserem Fall ist klar, dass Rang und Häufigkeit miteinander invers korrelieren: je niedriger der Rang eines Wortes ist, desto größer ist seine Häufigkeit, denn 1 ist der Rang des häufigsten Wortes etc. **Zipfs Gesetz** ist eigentlich kein Gesetz, sondern eine empirische Beobachtung, die aber durch ihre Regelmäßigkeit fast den Status eines Gesetzes hat; denn sie bestätigt sich für alle Arten von Texten. Wir kürzen den Rang eines Wortes mit $r(w)$ ab; seine Häufigkeit bezeichnen wir mit $f(w)$ (das f kommt von Frequenz). Zipfs Gesetz ist ein Potenzgesetz, und in seiner einfachsten Fassung besagt es:

$$f(w) \propto \frac{1}{r(w)} \tag{90}$$

Das ist ein Potenzgesetz, da $\frac{1}{x} = x^{-1}$. Was besagt diese Formel? Beachten Sie dass durch das Zeichen $\propto$ wir einen weiteren Term weglassen können; aber dieser Term darf keinen Exponenten haben. Was die Formel also besagt ist: es gibt eine Zahl k, so dass

$$f(w) \approx a_0(r(w))^{-1} + a_1 = a_0\frac{1}{r(w)} + a_1 \tag{91}$$

Durch einfache Termumformung erfahren wir, dass es a_0, a_1 gibt, so dass

$$f(w) \cdot r(w) \approx a_0 + a_1 r(w) \tag{92}$$

für alle Worte w, die in unserem Korpus vorkommen. Wenn wir das ganze etwas vereinfachen und $a_1 = 0$ setzen (d.h. wir gehen von $\approx$ zu $\propto$), sehen wir dass

$$f(w) \cdot r(w) \propto a_0, \tag{93}$$

d.h. Rang und Frequenz eines Wortes sind genau **invers proportional** zueinander. Z.B. werden wir das 10-häufigste Wort in etwa doppelt so oft finden wie das 20-häufigste Wort, und 10 mal häufiger als das 100-häufigste Wort. Das häufigste Wort wird sogar 100 mal häufiger sein als das 100-häufigste, etc.

Die Bedeutung von Zipfs Gesetz für die Computerlinguistik ist immens. Um zu sehen warum, betrachten wir ein Beispiel: nehmen wir an in unserem Text kommen 10000 Wörter (*types*) vor. Das häufigste Wort kommt 2000mal vor. Das bedeutet dann, dass das 2000-häufigste Wort nur einmal vorkommen sollte - und das wiederum heißt dass 8000 (von 10000!) Wörtern (*types*) überhaupt nur einmal vorkommen! Diese Wörter nennt man auch *hapax legomena* ("einmal gelesen"), und diese machen in den meisten Texten tatsächlich die große Mehrheit der *types* aus. Umgekehrt können wir daraus schließen, dass wenn wir die 100 häufigsten Wörter (*types*) abgedeckt haben, wir bereits den größten Teil der tokens abgedeckt haben!

Es gibt also zwei wichtige Konsequenzen für die Computerlinguistik: wenn wir beispielsweise ein Lexikon zur Worterkennung oder Übersetzung schreiben möchten, dann bekommen wir schon relativ gute Abdeckungen wenn wir nur die häufigsten Wörter abdecken. Wenn wir aber umgekehrt mit statistischen Methoden Informationen über Wörter erfahren wollen, z.B. in welchen Kontexten sie vorkommen können, dann haben wir für die allermeisten Wörter ein Problem, denn *fast alle* Wörter sind selten, und wenn ein Wort selten vorkommt, dann ist es schwierig mit statistischen Mitteln etwas zuverlässiges darüber zu erfahren.

8.12 Zipfs Gesetz und Wortlänge

Wir haben bereits die Funktion $|-| : \Sigma^* \to \mathbb{N}$ besprochen, die einem Wort seine Länge zuweist. Zipf hat ebenfalls beobachtet, dass es eine inverse Korrelation gibt von Wortlänge zu Worthäufigkeit. Wir haben also

$$f(w) \propto \frac{1}{|w|}. \tag{94}$$

Anders gesagt, je länger ein Wort, desto seltener ist es, und ein Wort mit Länge 5 sollte etwa 3-mal häufiger sein als ein Wort mit Länge 15 (sehr grob gesprochen). Zipf maßseinen Beobachtungen eine sehr große Bedeutung bei, und er führte sie alle zurück auf das Prinzip der kleinsten Anstrengung, die wir im Hinblick auf ein Ziel hin aufbringen möchten (*principle of least effort*), welches er allem menschlichen handeln zugrunde legte. Während seine Beobachtungen allgemein anerkannt sind, sind seine Hypothesen über die Ursachen der Beobachtungen weitestgehend zurückgewiesen worden.

Tatsächlich gibt es gute Argumente gegen seine Hypothesen. Erinnern Sie sich an die zufälligen Texte, von denen wir gesprochen haben. Ein solcher Text ist eine Zeichenkette über

$$(\Sigma \cup \square)^*,$$

wobei $\square$ für das Leerzeichen steht. In diesem Text hat jeder Buchstabe (und das Leerzeichen) eine Wahrscheinlichkeit, und diese ist vollkommen unabhängig von der Umgebung, in der er steht. Wir haben also beispielsweise

$$P(a) = 0.1, P(b) = 0.2, \ldots, P(\square) = 0.05$$

Ein Wort in diesem Text ist eine maximale Teilkette, die kein $\square$ enthält; d.h. eine Teilkette, die kein $\square$ enthält, aber links und rechts von $\square$ begrenzt ist.

Nehmen wir also an, wir generieren einen rein zufälligen Text nach unseren Wahrscheinlichkeiten. Hier gilt:

$$P(W = a_1 ... a_n) = P(a_1) ... P(a_n) P(\square)^2 \tag{95}$$

Hier ist W eine Zufallsvariable, die für ein beliebig ausgewähltes Wort steht. Eine merkwürdige Tatsache ist nun, dass wir auch in diesem rein zufälligen Text eine Zipf-Verteilung finden werden! D.h.

$$f(w) \propto \frac{1}{|w|} \tag{96}$$

gilt auch für die rein zufälligen Worte in unserem rein zufälligen Text – unter der Annahme dass

(97) $f(w) \approx P(W = w)$

also dass die Häufigkeit sich annähernd mit der Wahrscheinlichkeit deckt (das hängt zusammen mit dem Gesetz der grossen Zahlen).

Diese Zipf-Verteilung scheint also weniger durch besondere Eigenschaften natürlicher Sprache bedingt, sondern eine Folge allgemeinerer mathematischer Regelmäßigkeiten. Aber welche sind das? Nun, wir haben bereits einmal ausgerechnet, wie man die Wahrscheinlichkeiten von Worten in einem solchen Zufallstext berechnet. Die Wahrscheinlichkeit, dass wir irgendein Wort mit k Buchstaben treffen ist

(98) $P(\square)^2(1 - P(\square))^k$

Wir haben hier also eine **geometrische Verteilung**. Es ist klar, dass diese Zahl kleiner wird, je größer k wird. Daraus folgt, dass die Wahrscheinlichkeit von Worten immer weiter abnimmt, je länger sie werden, ganz unabhängig von den einzelnen Buchstaben aus denen sie bestehen und deren Wahrscheinlichkeiten. Wir haben also notwendig eine inverse Korrelation von Länge und Wahrscheinlichkeit.

8.13 Anmerkungen zu Zipf

Zipf-Verteilungen sind nicht nur in der Sprache allgegenwärtig. Sie gelten z.B. auch für Städte (Rang nach größe und Einwohnerzahl), Einkommensverhältnisse (zumindest in Italien, siehe Pareto-Verteilung) und viele andere Dinge.

9 R-Übung zu Verteilungen

R-Übung 1

Bitte bearbeiten bis zum 18.5.2021.

Wir gehen ins Casino, und nehmen ein Beispiel aus der Sitzung: Roulette mit Farben und Ziffern.

Nehmen Sie die folgenden Ereignisse und ordnen Sie sie Verteilungen zu. Dann schreiben Sie die Funktion (von Hand oder in R), mit der Sie deren Wahrscheinlichkeit berechnen können.

1. Die Wahrscheinlichkeit, dass Sie beim n-ten Spiel zum ersten Mal gewinnen von Roulette auf Zahl, mit $P(i) = 1/37$, für $i \in \{1, ..., 37\}$). Welche Zahlen Sie nehmen spielt natürlich keine Rolle.

2. Die Wahrscheinlichkeit, dass Sie bei 50 Spielen k mal gewinnen (von Roulette auf Zahl, mit den Wahrscheinlichkeiten wie in der letzten Aufgabe).

3. Die Wahrscheinlichkeit, dass Sie bei 20 Spielen kein einziges Mal gewinnen (von Roulette auf Zahl, mit den Wahrscheinlichkeiten wie in der letzten Aufgabe).

4. Sie spielen Poker. Wie ist die Wahrscheinlichkeit, dass Sie im französischen Blatt (32 Karten) genau 3 Damen bekommen (für Ihre 5 Karten total)?

R-Übung 2

Befehle für die Binomialverteilung sind `dbinom`; für Normalverteilung `dnorm`.

1. Plotten Sie den Graphen für die Binomialverteilung, mit $p = 0.5$, und $n = 1g0, 1500$ und 15000 (also drei Graphen).

2. Für jeden dieser Graphen erzeugen Sie einen Graphen der **Normalverteilung**, der diesen Graphen bestmöglich approximiert; Sie müssen also die Parameter μ, σ (Erwartungswert, Standardabweichung) entsprechen setzen.

   Tipp: Wenn Sie gut aufgepasst haben, kommen Sie mit ein bisschen Überlegen/Nachschlagen auf die korrekte Lösung. Sonst müssen Sie ein bisschen rumprobieren!

10 RX3 Daten visualisieren

Viele statistische Tests basieren auf gewissen Verteilungen. Aber: wenn wir einen Vektor wie verbs$LengthOfTheme betrachten, dann ist das in der trivialen Visualisierung eben einfach ein Zahlenstrang (Teilmenge von $\mathbb{R}$) mit Punkten darauf; wenn es schlecht läuft, haben wir sogar niemals zwei Punkte an derselben Stelle. Mit welchem Recht können wir sagen: die Daten sind annähernd normalverteilt/binomialverteilt etc.? Hier hilft es erstmal, wenn wir die Daten visualisieren; auch hierfür gibt es eine Reihe von nützlichen Funktionen in R. Was zunächst nützlich sein kann ist ein Balkendiagramm; das visualisiert man mit **barplot()**.

> barplot(verbs$LengthOfTheme)

Das Ergebnis ist aber noch nicht wirklich informativ: wir haben zu viele "Sprünge" in den Daten, um sie effektiv einer Verteilung zuzuordnen. Wir müssen die Balken größer machen, wobei ihre Höhe die Anzahl der darunterliegenden Datenpunkten angibt. Das nennt man ein **Histogramm**. Die entsprechende Funktion ist (mit vielen anderen) im Paket MASS, das wir zunächst laden müssen:

> library(MASS)
> truehist(verbs$LengthOfTheme)

Das Ergebnis ist bereits wesentlich informativer für uns: wir sehen hier dass wir eine Art Normalverteilung heben, wobei diese leicht asymmetrisch nach links ist. Wir können uns das also viell. vorstellen als eine asymmetrische Binomialverteilung. truehist() nimmt noch weitere Argumente:

> truehist(verbs$LengthOfTheme, xlab = "Length of theme", ylab = "Frequency", col = "grey")

Besonders wenn wir Grafiken exportieren möchten, können diese Befehle sinnvoll sein (sonst interessiert die Farbe wohl keinen). Es gibt verschiedene Formate in die wir exportieren können: png(), pdf(), jpeg(). Das ganze geht wie folgt:

> jpeg("barplot.jpeg", width =400, height = 400)

> truehist(verbs$LengthOfTheme, xlab = "Length of theme", ylab = "Frequency", col = "grey")
> dev.off()

jpeg() liefert den Rahmen, wir können damit auch bereits die Anzahl der Pixel bestimmen. RStudio erlaubt uns auch, direkt über den editor Grafiken zu exportieren.

Das Problem bei Histogrammen ist (wie am Beispiel deutlich zu sehen) ist dass wir diskrete Sprünge haben, aber eigentlich eine kontinuierliche Funktion approximieren wollen. Das macht R mit der Funktion **density()**, die die Werte interpoliert, also versucht durch eine Funktion zu approximieren. Zunächst aber nochmal zu Histogrammen. Es gibt eine weitere Funktion um Histogramme zu erstellen, nämlich **hist()**. Der Unterschied ist folgender:

- **truehist()** ist in erster Linie eine Funktion zur Visualisierung, die man z.B. nicht als Variablenbelegung speichert.
- **hist()** ist eine Funktion die besser geeignet ist um eine Variable zu belegen und weiter zu bearbeiten.

Dasselbe gilt auch für **density**: es produziert nicht in erster Linie einen Funktionsgraphen, sondern eine Funktion (was ja etwas anderes ist!).
> density(verbs$LengthOfTheme)

Hier bekommen wir zunächst die wichtigsten Parameter der Funktion mitgeteilt. Wir definieren uns nun zwei Variablen:

> d=density(verbs$LengthOfTheme)

> h=hist(verbs$LengthOfTheme)

Nun können wir die beiden Funktionen plotten:

> plot(d)
> plot(h)

Was sehen wir hier? Insbesondere erkennen wir gut dass unsere Verteilung nicht wirklich kontinuierlich ist: wir haben Spitzen allen $x \in \mathbb{R}$ s.d. $x \in log(\mathbb{N})$.

Was sehr nützlich ist, ist der Parameter **breaks** der Funktion hist():

```
> h=hist(verbs$LengthOfTheme, breaks = seq(0,6, by = 0.5))
```

Die Funktion **seq()** (mit drei Parametern) gibt hierbei einen Vektor an:

```
> seq(0,1, by = 0.1)
```

Wir haben also die Grenzwerte und die Schritte, mit denen wir dadurch gehen. Also:

```
> seq(0,10, by=1)
```

liefert dasselbe Ergebnis wie 1:10. Hiermit können wir also unsere Histogramme beliebig skalieren.

Zuletzt, bevor wir den Datensatz verlassen, noch folgende Anmerkung. verb$LengthOfTheme ist ein vektor mit log-transformierten Längen. Was ist eigentlich nochmal der Grund hierfür? Schauen wir mal an, was passiert wenn wir die Transformation rückgängig machen.

```
> verbL <- exp(verbs$LengthOfTheme)
> truehist(verbL)
> d2 = density(verbL)
> plot(d2)
```

Was wir sehen ist folgendes: wir haben eine Verteilung mit einem langen Rattenschwanz nach rechts: es gibt einfach sehr wenige sehr lange Worte, die die Symmetrie stören. In der log-Transformation bleibt natürlich eine Asymmetrie, aber in gewissen Sinne wird sie natürlicher, da die Werte eher zentriert sind, also weniger verteilt auf (seltene) Extremwerte. Wenn wir einen Datensatz haben, der unter einer gewissen Transformation "natürlicher" wird, dann kann man diese Transformation applizieren – aber nur unter der Voraussetzung, dass diese Transformation **bijektiv** ist, also rückgängig gemacht werden kann, und die Ordnungsverhältnisse ($\leq$ etc.) respektiert. Eine weitere Transformation, die man öfters benutzt ist die Transformation

$$\text{verbL} \mapsto \text{-1000/verbL}$$

Gründe für solche Transformationen liegen üblicherweise in der besseren Interpretierbarkeit der Daten.

Der lexdec Datensatz (kann man überspringen) Wir schauen uns nun einen etwas anderen Datensatz an aus languageR, nämlich den Datensatz **lexdec**. Das sind die Daten zu einem *lexical decision task* (wie heid).

> head(lexdec, n=5)

Hier haben wir ziemlich viele "Variablen". Das Prinzipt ist folgendes: Personen müssen entscheiden, ob ein Wort Englisch ist, und die Zeit hierfür wird gemessen (logarithmisch im Datensatz); hierzu gibt es eine Reihe von Eigenschaften des Wortes, Frequenz, Länge uvm. Uns interessiert erstmal nur die Spalte lexdec$RT:

> lexdec[1:5,"RT"]

Wir haben hier einen Vektor von (logarithmischen) Reaktionszeiten. Das eignet sich etwas besser für unsere Zwecke als verbs$LengthOfTheme.

> logRT <- lexdec$RT

> h <- hist(logRT)

> d <- density(logRT)

> plot(d)

Das sieht nun deutlich aus wie eine (leicht asymmetrische) Binomialverteilung:

> f <- function(x)dbinom(x,45,0.35)
> plot(f)

Funktionen dieser Form nennt man im Allgemeinen **Wahrscheinlichkeitsdichtefunktion** (Englisch: *probability density function, pdf*). Diese Funktionen sagen uns, wie "dicht" die Wahrscheinlichkeit über einem gewissen Bereich liegt. Das "d" in dbinom(), dnorm() etc.

Wir werden nun eine etwas andere Form von Verteilungen betrachten, nämlich Verteilungen, die uns sagen, wie die Wahrscheinlichkeitsmasse unter $f(x)$ **wächst** mit wachsendem x. Man nennt diese Funktion die **Wahrscheinlichkeitsmassefunktion** (Englisch: *probability mass function, pmf*). Jede pdf f hat eine assoziierte pmf die eindeutig bestimmt ist.

Falls f diskret ist, dann gilt:

$$(99) \quad pmf(f)(x) = \sum_{y:y \leq x} f(y)$$

Falls f eine stetige Funktion ist, dann gilt:

$$(100) \quad pmf(f)(x) = \int_{-\infty}^{x} f(x)$$

Die pmf gibt also an, wieviel Masse bis zu einem gewissen Punkt belegt wurde. Man ruft diese Funktionen auf mit den Befehlen **pnorm()**, **pbinom()** etc. Diese Funktionen geben also an, in welchem Maße die Wahrscheinlichkeitsmasse zunimmt.

```
> fmass <- function(x){pnorm(x,0,1)}
> plot(fmass,-3,3)
```

Hier sehen wir, dass solche Funktionen eine typische sog. Sigmoid-Kurve haben. Diese spielen als Aktivierungsfunktionen eine wichtige Rolle im maschinellen Lernen. Wir können hier z.B. ablesen dass praktisch 100% der Wahrscheinlichkeitsmasse ≤ 2 ist. Was uns erstmal interessiert ist eine Variante dieser Funktion, nämlich $qnorm(), qbinom()$:

```
> fq <- function(x){qnorm(x,0,1)}
> plot(fq,0,1)
```

Das q steht hier für *quantile*. Denn diese Massefunktionen gibt uns tatsächlich, für eine beliebige Wahrscheinlichkeit $p \in [0, 1]$, das p-Quantil. Wir möchten jetzt diese Quantile für unsere Daten plotten. Betrachten wir zunächst folgendes:

```
> plot(logRT)
```

Wenn plot() einen einfachen Vektor als Argument bekommt (statt einer Funktion), dann nimmt es an, die Nummer der Komponente ist x, der Wert der Komponente ist $f(x)$. Wir haben also eine (partielle) Funktion $vec : \mathbb{N} \to \mathbb{R}$. Diese rohen Daten sehen schon viel ansprechender aus, wenn wir den Vektor vorher sortieren:

> plot(sort(logRT))

Hieran sehen wir bereits, wie sich die Quantitäten verhalten: wir haben die typische (umgedrehte) Sigmoidkurve, die einer Normalverteilung (Massefunktion) entspricht. Wenn wir stattdessen die Quantile plotten, ändert sich die Kurve nicht sonderlich:

> plot(quantile(logRT))

Das einzige was uns daran stören sollte ist, dass die Punkte zu dünn sind. Das kann man aber leicht beheben:

> plot(quantile(logRT, seq(0,1,0.05)))

Hier sehen wir, dass diese Kurve der (sortierten) Datenkurve verblüffend ähnlich sieht. Übrigens kann man auf diese Art auch einfach die Quantile abrufen:

> quantile(logRT, seq(0,1,0.1))

Das liefert uns die gewünschten Quantile. Was man in diesem Zusammenhang auch nutzen kann sind sogenannte **Boxplots**. Hierfür gibt es den Befehl **boxplot()**.

> boxplot(logRT)

Ein Boxplot liefert uns folgende Informationen:

1. Den Median,
2. das obere und untere Quartil (0.25 und 0.75)
3. und die Extremwerte;

4. außerdem gibt es die Antennen; die bestehen aus dem oberen – unteren 2.5% Quartil. Wir sehen sie in diesem Fall nur nach oben, da sie nach unten derart eng an den Extremwerten der mittleren 95% liegen, dass R auf eine gesonderte Darstellung verzichtet.

Hausaufgabe 5

Bitte bearbeiten bis zum 25.5.2021

Wir haben in der Stunde gesehen, dass hist und density nicht wirklich Funktionen definieren, die Werte ausgeben. Das ist natürlich schade, aber wir knnen hier nachhelfen.

1. Implementieren Sie eine Funktion f, so dass gilt: $f(n)$ liefert die Anzahl der Einträge in exp(verbs$ LengthOfTheme), die den Wert n haben (also wie der Balken im Histogramm).

2. Schreiben Sie ein Programm, der Sie einen beliebigen Vektor mit natürlichen Zahlen geben können, und die Ausgabe ist die entsprechende Funktion wie in 1.

11 Hypothesen prüfen

11.1 Verteilungen und Vertrauensgrenzen in R

Wir kommen nun zur sog. **inferentiellen Statistik**; es geht also darum, aus Datensätzen Inferenzen zu ziehen. R ist eine mächtige Programmiersprache, die für statistische Analysen ausgelegt ist. Dementsprechend sind bereits viele wichtige Funktionen eingebaut und müssen nicht erst mühsam definiert werden. Das umfasst z.B. die Funktion $\binom{n}{k}$, die geschrieben wird mit `choose(n,k)`:

```
> n<- 10
> k<- 6
> choose(n,k)
[1] 210
```

Das erlaubt uns beispielsweise, die Binomialverteilung zu definieren:

```
> bin.vert<- function(k, n, p) {
choose(n,k) * p^ k * (1-p)^ (n-k)
}
```

Das liefert uns z.B.

```
> bin.vert(40,150,0.75)
[1] 2.631372e-35
```

wobei $e-35$ soviel bedeutet wie *mal* 10^{-35}, d.h. wir müssen das Komma um 35 Stellen nach links verschieben, um den richtigen Wert zu bekommen. Die Binomialverteilung ist übrigens auch schon eingebaut in R, wir hätten uns die Arbeit also auch sparen können; sie wird abgerufen als `dbinom(k,n,p)`.

Wir werden jetzt einen einfachen Fall von statistischer Inferenz betrachten. Es folgt aus den grundlegenden Eigenschaften der Binomialverteilung und des Erwartungswertes, dass

$$(101)\quad argmax_{p\in[0,1]}\texttt{dbinom}(\texttt{k},\texttt{n},\texttt{p}) = \frac{k}{n}$$

D.h. für gegebene k, n nimmt die Funktion ihr Maximum in $\frac{k}{n}$. Umgekehrt

gilt natürlich auch folgendes:

$$(102)\ argmax_{0\leq i\leq n}\texttt{dbinom}(\texttt{i},\texttt{n},\frac{\texttt{k}}{\texttt{n}}) = k$$

D.h. für eine Gegebene Wahrscheinlichkeit $\frac{k}{n}$ und gegebene Anzahl von Iterierungen n nimmt die Funktion ihr Maximum für $i = k$ (erster Parameter). Nun kann man aber folgendes beobachten: je größer ich n, k wähle (bei gleichbleibendem $\frac{n}{k}$), desto kleiner wird dieses Maximum:

```
dbinom(4,10,(4/10))
[1] 0.2508227

dbinom(40,100,(4/10))
[1] 0.08121914

dbinom(400,1000,(4/10))
[1] 0.02574482
```

Der Grund hierfür ist ganz einfach: wir haben eine diskrete Funktion (nur endlich viele Werte > 0), die sich insgesamt auf 1 summieren, und je größer wir n, k wählen, desto mehr Werte sind > 0, während ihre Gesamtsumme gleich bleibt, d.h.

$$(103)\ \sum_{1\leq k\leq n}\texttt{dbinom(k,n,p)} = 1).$$

Also müssen die Werte kleiner werden (man sagt: die Wahrscheinlichkeitsmasse wird aufgeteilt unter diesen Werten). Das bedeutet aber auch: je öfter wir ein Experiment iterieren, desto unwahrscheinlicher wird das wahrscheinlichste Ergebnis, und je öfter wir ein Bernoulli Experiment wiederholen (mit Anzahl n), desto unwahrscheinlicher wird es, dass wir tatsächlich den "wahren" Parameter $\frac{n}{k}$ treffen, d.h. k-mal Ergebniss 1 haben. Das widerspricht zunächst unserer Intuition, da wir denken: je öfter wir ein Experiment iterieren, desto mehr nähern sich die Ergebnisse der "wahren" Wahrscheinlichkeit an.

Dieses Problem ist kompliziert und löst sich im "Gesetz der großen Zahlen" auf. Wir umgehen das erstmal ganz praktisch, indem wir anstelle einzelner Werte die sog. *Vertrauensgrenzen* oder *Konfidenzintervalle* benutzen. Intervalle werden in R mittels Vektoren gehandhabt:

```
> 0:5
[1] 0 1 2 3 4 5
> x<- 0:6
> x[3:5]
[1] 2 3 4
> sum(x)
[1] 21
```

Die letzte Zeile ist die Summe $1 + 2 + \ldots + 6$. Wir definieren jetzt eine Wahrscheinlichkeitsfunktion, die Intervalle berechnet:

```
> p<- 1/2
> n<- 40
>int<- dbinom(0:n,n,p)
```

Diese Funktion berechnet eine Liste `dbinom(0,40,1/2),dbinom(1,40,1/2), dbinom(2,40,1/2)` etc. Hierbei gibt es zu beachten dass `dbinom(0,40,1/2)=int[1]`, `dbinom(40,40,1/2)=int[41]`! Das wahrscheinlichste Ergebnis für k ist – nach allem was wir wissen –

```
> int[21]
[1] 0.1253707
```

Das ist relativ niedrig. Was wir aber jetzt machen können ist auf Intervalle von Ergebnissen zugreifen:

```
> int[19:23]
[1] 0.1031187 0.1194007 0.1253707 0.1194007 0.1031187
```

Was wir sehen ist folgendes: 1. die Verteilung ist symmetrisch (denn $p = 0.5$), 2. sie hat ihr Maximum bei $k = 20$ (entspricht `int[21]`!) Es gibt aber noch dritte wichtige Beobachtung:

```
> sum(int[19:23])
[1] 0.5704095
```

D.h.: wenn wir die Werte für die Ergebnisse $k = 18$ bis $k = 22$, also die 5 wahrscheinlichsten Werte addieren, dann entfällt auf diese Werte bereits

die Hälfte der Wahrscheinlichkeitsmasse! Wir werden diese Prozedur jetzt leicht generalisieren. Dazu müssen wir noch wissen, dass für einen Vector wie `vec<- 1:n` wir den k-ten Wert mit `vec[k]<- i` ändern können.

```
> mittel<- 21
> interval<- 1:20
> for (i in 1:20) { indices<- seq(mittel-i, mittel+i) ; interval[i]<-
sum(int[indices]) }
```

Was wir hier bekommen ist folgendes: `interval[4]` ist `sum(int[21-4:21+4])`, also die Summe der 9 wahrscheinlichsten Ergebnisse.

```
>interval[5]
[1] 0.9193095
```

Diese machen also bereits 90% der Wahrscheinlichkeitsmasse aus! Damit wir diese Zahlen etwas anschaulicher machen, setzen wir sie in eine Tabelle.

```
> vertrauen<- data.frame(grenze = rep(1:20), wahrscheinlichkeit =
interval)
> vertrauen[1:6,1:2]
    grenze  wahrscheinlichkeit
 1  1       0.3641720
 2  2       0.5704095
 3  3       0.7318127
 4  4       0.8461401
 5  5       0.9193095
 6  6       0.9615227
```

Hier sehen wir ein fundamentales Prinzip der Statistik, das eigentlich willkürlich ist: man legt normalerweise die **Vertrauensgrenze** bei 95% fest. Das heißt: wenn wir p als Parameter eines Bernoulli-Raumes nicht kennen, nehmen wir erstmal an dass $p = 0.5$ (das ist die sog. uniforme Verteilung, die unseren Mangel an Wissen widerspiegelt). Man nennt das auch die **Nullhypothese**. Wir nehmen nun also diesen Parameter als gegeben an. Dann zeigt uns unsere Funktion `int` dass unser Ergebnis k aus 40 Iterierungen des Experiments mit einer Wahrscheinlichkeit von über 0.95 im Interval [21-6,21+6] liegen muss. Wenn das Ergebnis darin liegt, dann finden wir

die Nullhypothese noch akzeptabel, wenn das Ergebnis außerhalb der Vetrauensgrenzen liegt, dann weisen wir die Nullhypothese zurück: sie ist zu unplausibel. Unsere Vertrauensgrenze liegt also bei einer Abweichung von 6 vom Erwartungswert; wenn unser Ergebnis innerhalb der Grenzen liegt, haben wir nichts gewonnen; wenn es außerhalb liegt, lehnen wir die Nullhypothese ab. Wir stellen das ganze nun grafisch dar:

```
> plot(vertrauen$grenze, vertrauen$wahrscheinlichkeit, type="b",
xlab="Grenze", ylab="Wahrscheinlichkeit")
> segments(0,0.95,5.7,0.95)
> segments(5.7,0,5.7,0.95)
```

Wir sehen also wie mit wachsender Größe des Intervalls die Wahrscheinlichkeitsmasse steil wächst und letztlich langsam gegen 1 konvergiert.

Hier kann man nun auch sehen, wie sich unsere vorige Paradoxie auflöst. Beim jetztigen Beispiel liegen die Vertrauensgrenzen bei einer Abweichung von 6 vom Mittelwert bei einer maximal möglichen Abweichung von 20. Wir rechnen nun dasselbe Beispiel nochmal mit $n = 400$ durch.

```
> n = 400
> sum(int2[(201-60):(201+60)])
[1] 1
```

Wir haben – proportional gesehen, die Grenzen genauso gesetzt wie vorher, diesmal bei 60 von 200. Wir sehen aber, dass der Wert schon so nahe an 1 ist, dass R ihn nicht mehr unterscheidet. Dass heißt bei einer Iterierung von $n = 400$ eine Proportional Abweichung von 3/10 um ein vielfaches unwahrscheinlicher ist! In diesem Sinne gibt uns eine häufigere Iteration ein besseres Abbild der tatsächlichen Wahrscheinlichkeit.

Der Sinn der Vetrauensgrenzen ist eigentlich folgender: wir nehmen eine zugrunde liegende (normale) Verteilung als **Nullhypothese** an; falls unser tatsächlich beobachtetes Ergebnis außerhalb dieser Grenzen liegt, weisen wir die Nullhypothese zurück. Wir haben also ein einfaches Mittel, eine Hypothese zurückzuweisen.

11.2 Sequentielle Überprüfung von Hypothesen 1

Eigentlich ist dieser Abschnitt eher eine Fußnote, er ist aber wichtig um dem Mißbrauch der präsentierten Methoden vorzubeugen, und um ein besseres Verständnis für ihre Natur zu bekommen. Das Problem ist folgendes: nehmen wir an, wir machen ein Experiment (100 Münzwürfe), allerdings ist das Ergebnis nach unserer Auffassung nicht konklusiv – es erlaubt keine definitive Schlussfolgerung darüber, ob die Münze fair ist oder nicht. Also wiederholen wir das Experiment, und prüfen das Ergebnis usw. Irgendwann haben wir dann ein zufriedenstellendes Ergebnis erreicht. Diese Herangehensweise ist leider gängige Praxis, stellt aber in den meisten Fällen einen groben Mißbrauch dar.

Betrachten wir die Methode der Vertrauensgrenzen, und nehmen wir das Procedere sieht in der Praxis wie folgt aus:

1. Wir machen das Experiment (100 Würfe), schauen ob das Ergebnis innerhalb unser Vertrauensgrenzen liegt – und die Antwort ist positiv.

2. Aus irgendeinem Grund – vielleicht liegt es am Rande, vielleicht haben wir einen Verdacht – befriedigt uns das nicht.

3. Wir wiederholen das Experiment (100 Würfe), und schauen ob das Gesamtergebnis ($n \cdot 100$ Würfe) innerhalb der Vertrauensgrenzen liegt.

4. Nach n Durchgängen der vorigen Punkte 2. und 3. (z.B. $n = 5$) liegt das Ergebnis außerhalb der Vertrauensgrenzen. "Aha", sagen wir, "wußt ich es doch. Zum Glück habe ich nicht aufgegeben!"

Wo liegt der Fehler in dieser Vorgehensweise?

Schauen wir uns zwei Ereignisse an:

> E_{500}: Das Ergebnis liegt nach 500 Würfen innerhalb der Vertrauensgrenzen.

Dem gegenüber steht ein anderes Ereignis:

> E^5_{100}: Das Ergebnis liegt sowohl nach 100,200,300,400 also auch nach 500 Würfen innerhalb der Vetrauensgrenzen.

Haben wir

$$P(E_{500}) = P(E_{100}^5)?$$

Diese Frage ist leicht zu beantworten, denn unsere Ereignisse konsitutieren sich als Mengen von Ergebnissen (Folgen in $\{0,1\}^{500}$). Nun ist es leicht zu sehen, dass

$$E_{100}^5 \subseteq E_{500}$$

nach der Definition der beiden Ereignisse. Nun nehmen wir aber folgendes Ergebnis:

$e := \langle 0^{250}, 1^{250} \rangle$, d.h. erst 250mal Kopf, dann 250mal Zahl.

Wir haben natürlich $e \in E_{500}$ – denn das Ergebnis liegt genau am Erwartungswert. Wir haben aber $e \notin E_{100}^5$, denn nach den ersten 200 Würfen (alle Kopf!) liegt unser Ergebnis mit Sicherheit außerhalb jeglicher Vertrauensgrenze. Daraus folgt:

$$E_{100}^5 \subsetneq E_{500}$$

und folgereichtig, da $P(e) > 0$,

$$P(E_{100}^5) < P(E_{500}).$$

Wie stark ist dieser Effekt? Unser Beispiel e ist derart unwahrscheinlich, dass wir es vernachlässigen können. Das Problem ist aber, dass die Vertrauensgrenzen relativ mit wachsender Zahl von Ergebnissen relativ immer enger werden, es werden also immer mehr Ergebnisse ausgeschlossen!

Betrachten wir einmal genauer passiert: die Eregnisse

$$E_{100},\ E_{100}^2,\ E_{200}$$

sind analog zu den obigen definiert. Für E_{100}^2 (2 Iterationen) können wir die folgende Rechnung aufmachen:

$$P(E_{100}^2) = P(E_{100} \cap E_{200}) = P(E_{200}|E_{100})P(E_{100}) \tag{104}$$

Wir können diese Rechnung leicht verallgemeinern; es handelt sich nämlich um eine sogenannte **Markov-Kette**: in einem Satz bedeutet das:

$$P(E_{300}|E_{200}, E_{100}) = P(E_{300}|E_{200})$$

etc., also zählt immer nur das letzte Ergebnis. Also gilt:

(105)
$$P(E^5_{100}) = P(E_{500}|E_{400})P(E_{400}|E_{300})P(E_{300}|E_{200})P(E_{200}|E_{100})P(E_{100})$$

oder etwas allgemeiner ausgedrückt:

(106) $P(E^{n+1}_{100}) = P(E_{100(n+1)}|E_{100n})P(E^n_{100})$

Damit ist also klar sichtbar, dass wir Wahrscheinlichkeit mit sinkendem n immer kleiner wird. Im Gegensatz dazu vergleiche das per Definition gilt:

(107) $P(E_{500}) \approx P(E_{100}) \approx P(E_m) \approx c$,

wobei $m \in \mathbb{N}$ beliebig und c unsere Vertrauenskonstante ist, z.B. 0.95. Hier müssen wir also schon sehen, dass das vorgehen der iterierten Tests äußerst problematisch ist – mir messen etwas ganz anderes als was wir vorgeben! Es kommt aber noch besser: mit etwas Überlegung und etwas komplizierterer Mathematik ist es nicht sonderlich schwer zu sehen dass

(108) $lim_{n\to\infty}P(E^n_{100}) = 0$

anders gesagt: egal wie weit/eng unsere Vertrauensgrenzen sind, wenn wir sie Methode oben nur oft genug iterieren, werden wir mit mathematischer Sicherheit irgendwann ein Resultat finden, dass außerhalb unserer Grenzen liegt! Wenn sie also diese Methode als legitim erachten, können wir mit mathematischer Sicherheit jede Nullhypothese "widerlegen".

Wohlgemerkt : Sie fragen sich warum sollte jemand eine Münze so oft werfen? Stellen Sie sich folgendes vor: es wird ein Medikament getestet; die Nullhypothese ist, dass es keine Wirkung hat (das ist etwas komplizierter, am im Prinzip ähnlich). Sie haben einige Jahre hart gearbeitet, Tiere gequält, und sind völlig überzeugt von der Wirksamkeit des Medikaments. Sie machen nun eine Testreihe an Menschen; wenn die Testreihe gut läuft, dann verdient die Firma viel Geld, Sie steigen in der Karriereleiter auf. Wenn die Tests ergebnislos verlaufen, dann haben Sie viel Zeit, die Firma viel Geld in den Sand gesetzt, Ihr Chef ist sauer, Ihre Frau enttäuscht etc.

Sie machen die Testreihe mit 100 Teilnehmern, und das Ergebnis liegt gerade am Rand der Vertrauensgrenzen (aber innerhalb!). Ihr Chef sagt: "Dann probieren Sie halt in Gottes Namen den Test mit noch 100 Teilnehmern!"

11.3 SÜH 2 – Unabhängig

Wir können das Problem auch anders angehen: anstatt dass wir unsere Fallzahlen aufaddieren, wiederholen wir einfach das Experiment von 0 an, und lassen das alte Experiment z.B. einfach in der Schublade verschwinden, tun also so, als hätte es nie stattgefunden. Beim 5 Durchlauf haben wir endlich das gewünschte Ergebnis. Ist das in Ordnung? Wir haben allen Grund mißtrauisch sein: wenn wir unsere Vertrauensgrenze bei $c := 0.95$ festsetzen, dann gibt es immerhin eine Wahrscheinlichkeit von $\frac{1}{20}$, dass wir rein zufällig außerhalb landen. Wie ist also die Wahrscheinlichkeit, dass wir mit 5 Experimenten 1 Ergebnis erzielen, dass außerhalb der Vertrauensgrenzen liegt? Das ist nun einfach, denn die Experimente sind nach unserer Annahme unabhängig. D.h. die Wahrscheinlichkeit, dass wir bei 5 Durchgängen immer ein Ergebnis innerhalb der Vetrauensgrenzen finden, liegt unter Annahme von H_0 bei

(109) $0.95^5 = 0.7737809375$

D.h. die Wahrscheinlichkeit unter dieser Methode ein Ergebnis zu finden, bei dem wir H_0 zurückweisen, ist

(110) $1 - 0.7737809375 \approx 0.23$

also bereits bei fast $\frac{1}{4}$! Hier steigt die Wahrscheinlichkeit also rapide, und es ist offensichtlich, dass die Wahrscheinlichkeit, bei n Experimenten immer innerhalb der Vertrauensgrenzen zu landen, gegen 0 geht.

Die Methode, die wir hier betrachtet haben, würde kein Forscher, der kein bewußter Betrüger ist, anwenden (im Gegensatz zu der obigen!). Aber dennoch ist sie fast noch gefährlicher: nehmen wir an, es gibt eine Hypothese H (wie die Unfairness des Würfels), die aber die Eigenschaft hat, dass sie

1. relativ nahe liegt/populär ist/zu den Dingen gehört die wir alle gerne hören; und

2. wer sie statistisch belegen kann, der kann sich eines sehr positiven Echos sicher sein.

Dementsprechend gibt es viele Forscher, die ähnliche Experimente machen (sagen wir 5). 4 von ihnen haben keine guten Ergebnisse (sie liegen innerhalb der Vertrauensgrenze von H_0). Das will niemand hören, und verschwindet

im Schreibtisch. Der fünfte aber hat "gute" Ergebnisse (beim ersten Experiment!), und macht sie natürlich publik (mit bestem Gewissen!). Wir sehen aber natürlich sofort: die Situation ist genau wie oben, denn wer das Experiment ausführt, ist dem Zufall egal!

Wir haben also eine sehr kritische Situation, da wir nur den Teil der Experimente sehen, die einen wünschenswerten Ausgang haben! Das hat (vermutlich) dazu geführt, dass sich viele wichtige experimentelle Ergebnisse der Psychologie in den letzten Jahre also falsch bzw. artifiziell herausgestellt haben. Das entscheidende ist daher, dass Ergebnisse **replizierbar** sind, also wir bei wiederholten Experimenten immer das gleiche Ergebnis haben.

11.4 p-Hacking

Schokolade Übrigens hat diese ganze Art von vorgehen einen Namen: **p-hacking** (man versucht den p-Wert zu hacken). Ein berühmtes Beispiel ist die **Schokoladenstudie**, die die Korrelation von Schokoladenkonsum und Haarverlust (bei Männern) belegen soll.

Idee: es gibt viele Arten von Schoko-Süssigkeiten. Probiere solange alle Kombinationen aus, bis klar ist ($p < 0.01$), dass Männer die M& Ms, kein Snickers aber dafür Bounty essen, eher an Haarverlust leiden. Es ist relativ klar dass wir eine solche Gruppe finden: nimm an, wir betrachten 10 Schokoladenhaltige Süssigkeiten. Das ergibt 10 binäre Merkmale, also $2^{10} = 1024$ mögliche Kombinationen. Auch wenn wir eine Stichprobe von 10,000 Männern betrachten, ist relativ klar dass wir irgendwie eine Gruppe herausfiltern können, die weniger Haare hat.

Man nennt dieses Vorgehen auch **explorativ**: wir betrachten einfach Daten ohne jedwede Hypothese, und wenn wir eine Auffälligkeit finden, dann formulieren wir die entsprechende Hypothese ("Männer die M& Ms, kein Snickers aber dafür Bounty essen, leiden eher an Haarverlust") und testen sie. Das ist ebenfalls klarer Betrug. Man kann explorative Studien machen, um Hypothesen zu finden/formulieren. Wichtig ist aber:

- Wenn man einen Datensatz genutzt hat, um eine Hypothese zu finden, dann ist er damit "verbrannt".
- Wenn man die Hypothese nun prüfen will, braucht man einen neuen, komplett unabhängigen Datensatz!

Impfungen Gewisse Impfungen (z.B. Masern) werden normalerweise in einem Alter verabreicht, in dem auch die ersten Symptome von Autismus-Störungen bemerkbar werden (die Störungen selbst sind aber wahrscheinlich angeboren). Wir erwarten also, dass manche Kinder relativ kurz nach der Impfung gewisse Autismus-Symptome zeigen. Die betroffenen Eltern machen sich natürlich trotzdem darüber Gedanken.

Wenn sich dann betroffene Eltern/Leute aus ihrem Umkreis in Gruppen organisieren, dann bekommen sie natürlich den Eindruck, es gäbe überwältigende Evidenz dafür dass Impfungen zu autistischen Störungen führen. Aber natürlich ist auch dass nur eine Form des p-hacking: die Gruppe hat sich ja aus demselben Grund gebildet, den sie als Evidenz sieht – so als würden wir alle Würfel versammeln, die bei ihren ersten drei Würfen auf 6 gelandet sind, und das dann als Evidenz verkaufen (wofür auch immer).

Unabhängige, groß angelegte Studien finden übrigens keine Evidenz für einen Zusammenhang von autistischen Störungen und Impfungen. Man sieht wie wichtig es ist, die Datenauswahl unabhängig von der Hypothese zu machen!

Das hat übrigens auch mit zwei wichtigen Begriffen zu tun:

1. *Survivorship bias* Wir hören auf den herausragenden Fall, beachten aber nicht die anderen Fälle. Z.B. ein Fussballstar beschreibt die Methode, mit der er zum Erfolg gekommen ist. Dafür gab es 100 andere Talente, die mit derselben Methode mit 17 kaputte Knie hatten. Das sagt also nicht über Qualität der Methode aus – der Überlebende kann auch aus Glück überlebt haben! (Aber nicht Reinhold Messner, der hat zu oft überlebt)

2. *Anekdotische Evidenz* "Der Frisör von meiner Tante kennt einen, der hat von Ibuprofen sein Bein verloren" – es gibt viele solcher Geschichten, die meisten brauchen keinen Kommentar. Es gibt aber Dinge, für die gibt es haufenweise anekdotische Evidenz: von zu viel Kälte dann bekommt man einen Schnupfen, man muss Fleisch essen wenn man stark werden will, usw. Das Problem ist dass bei anekdotischer Evidenz alle Faktoren von oben reinspielen. Die einzelne anekdotische Evidenz hat einen Wert von 0, die haufenweise summierte anekdotische Evidenz also ebenfalls. Man kann anekdotische Evidenz aber benutzen, um Hypothesen zu formulieren und dann kontrolliert zu testen.

Der Dunning Kruger Effekt

Hier ein berühmtes Ergebnis, das mittlerweile widerlegt wurde: unterdurchschnittlich begabte Personen (in Logik, Humor, Sprachverständnis) *überschätzen* ihre Fähigkeiten, überdurchschnittlich begabte Personen unterschätzen sie.

Das stimmt aber gar nicht, ist ein reines Artefakt der sog. **Regression zur Mitte**!

Das zugrundeliegende Problem der orthodoxen, Signifikanz-basierten Statistik ist: wir arbeiten immer nur darauf hin, die "Nullhypothese" "zurückzuweisen". Aber wir erlauben es niemals, dass sie bestätigt wird. Das ist so, als spielen wir mit einem Gegner solange, bis wir gewinnen – aber wann immer wir hintenliegen, geht das Spiel weiter. Klar das wir irgendwann gewinnen (bei einem Spiel mit Zufallseffekten).

p-Werte sind **fundamental asymmetrisch** und daher – spieltheoretisch ausgedrückt – **unfair** gegenüber der Nullhypothese.

12 RX4a Übung

Berechnen Sie, bei welchen Ergebnissen die Vertrauensgrenzen für die folgenden Binomialverteilungen liegen:

a. $n = 50, p = 0.5$, Vertrauenskonstante $c = 0.95$

b. $n = 200, p = 0.5$, Vertrauenskonstante $c = 0.99$

c. Schreiben Sie ein Programm, dass n und c als Argumente nimmt, und Ihnen die Abweichung vom Erwartungswert, die ausserhalb der Vertrauensgrenze liegt, als Ausgabe liefert.

d. Nehmen wir an, $p = 0.6$. Wie würden Sie nun die Vertrauensgrenzen ausrechnen, wo liegt das Problem?

e. Berechnen Sie die Vertrauensgrenzen von außen für $p = 0.6$, $c = 0.95$. Das ganze geht sehr einfach mit der Funktion *qbinom*.

RX4b

Implementieren Sie ein Zufallsexperiment: eine Funktion $f(n, m)$, wobei n die Anzahl der fairen "Münzwürfe" ist, $f(n, m)$ liefert

1. Relative Häufigkeit von 1

2. ist die Antwort auf die Frage: ist das Ergebnis signifikant (also positive) ($p < 0.05$), würden Sie also die Nullhypthese zurückweisen.

m zuletzt ist ein Parameter, die die Anzahl der Iterationen des Vetrauenstests angibt: $n = 100, m = 5$ wäre E^5_{100}. $f(n, m)$ ist in diesem Fall signifikant, wenn wir dabei *einmal* ein signifikantes Ergebnis hatten.

Übung

Abgabe bis zum *vor dem Seminar*.

1. Bearbeiten Sie Teil a. und b. der obigen Übung.

2. Nehmen wir folgendes an: Sie führen ein Experiment aus (Einfachheit halber Munzwürfe), sie wollen die Münze 100mal werfen, und schauen ob

das Ergebnis innerhalb der Vertrauensgrenze ($c = 0.95$) der Nullhypothese liegt (um zu bestimmen, ob die Münze fair ist). Nach 50 Würfen haben Sie den starken Verdacht, dass die Münze unfair ist, da sie bis dahin ein sehr unausgewogenes Ergebnis haben. Also sagen Sie: "Ich spare mir die 50 restlichen Würfe, prüfe das Ergebnis jetzt an dieser Stelle. Ist ja auch Wurst ob ich ursprünglich 100 oder 1000 oder 50mal werfen wollte." Tatsächlich liegt das Ergebnis außerhalb der Vertrauensgrenzen; Sie weisen also H_0 zurück. Nun die Frage: ist Ihr Vorgehen legitim?

13 Sequentielle Bayesianische Hypothesenprüfung

13.1 Der Bayesianische Ansatz

Nehmen wir an, wir haben eine Münze. Wir werfen sie 10 mal, und wir erhalten

5-mal Kopf, 5-mal Zahl; nennen wir dieses Ereignis $\mathcal{E}$.

Was ist die Wahrscheinlichkeit, dass das passiert? Vorsicht: wir können das natürlich nicht wir; wir kennen ja gar nicht die Wahrscheinlichkeit, mit der die Münze Kopf/Zahl gibt; insbesondere wissen wir gar nicht, ob die Münze fair ist oder nicht. Diese Situation ist im "wirklichen Leben" wesentlich häufiger als die dass wir die Wahrscheinlichkeitsverteilung kennen. Was wir in dieser Situation meistens wollen ist folgendes:

> Wir würden gerne wissen wie wahrscheinlich eine gewisse *Wahrscheinlichkeitsverteilung* ist, gegeben das Ergebnis dass wir beobachtet haben.

Also in unserem Fall: gegeben dass wir aus 10 Würfen 5-mal Kopf haben, wie wahrscheinlich ist es, dass die Münze fair ist?

Das ist nicht ganz einfach, und ohne weitere Annahmen sogar unmöglich. Wir nähern uns dem Problem zunächst wie folgt. Nimm an wir haben zwei Verteilungen p_1, p_2, die eine gegeben durch

$$p_1(K) = p_1(Z) = 0.5$$

, die andere durch

$$p_2(K) = 0.4,\ p_2(Z) = (0.6)$$

(K ist das Ereignis Kopf, Z ist Zahl). Nenne die erste Verteilung F (wie fair), die zweite U (wie unfair). (Genauer: das Ereignis, dass diese Verteilung die "wahre" ist) Wir können nun sehr einfach die Wahrscheinlichkeit von $\mathcal{E}$ gegeben die Verteilung F ausrechnen:

$$P(\mathcal{E}|F) = \binom{10}{5} 0.5^5 \cdot 0.5^5 \approx 0.246 \tag{111}$$

Ebenso leicht lässt sich die Wahrscheinlichkeit von $\mathcal{E}$ gegeben die Verteilung U ausrechnen:

$$P(\mathcal{E}|U) = \binom{10}{5} 0.6^5 \cdot 0.4^5 \approx 0.201 \tag{112}$$

Beachten Sie dass an diesem Punkt Wahrscheinlichkeitsverteilungen selbst Ereignisse geworden sind! Was wir möchten sind nun die Wahrscheinlichkeiten

$$P(F|\mathcal{E})$$

und

$$P(U|\mathcal{E}),$$

also die Wahrscheinlichkeiten der Wahrscheinlichkeitsverteilungen *gegeben* unser Würfelergebnis. Uns allen ist klar, dass wir hier mit dem Satz von Bayes arbeiten müssen.

Nun kommt allerdings der Punkt wo wir einige Annahmen machen müssen, die etwas willkürlich, aber in der einen oder anderen Form unvermeidbar sind. Die erste Annahme ist folgende:

A1 Wir nehmen an, dass entweder F oder U der Fall ist, d.h. $P(U) + P(F) = 1$.

Das ist natürlich willkürlich, denn es gibt noch (unendlich) viele andere denkbare Wahrscheinlichkeitsverteilungen für unsere Münze. Allerdings müssen wir die Möglichkeiten in irgendeiner Form einschränken, um an dieser Stelle weiter zu kommen. (Es geht aber auch ohne, ist aber komplizierter!) Die zweite Annahme die wir machen müssen ist:

A2 Wir müssen $P(F)$ und $P(U)$ bestimmte Werte zuweisen, das sogenannte *apriori*.

Der Grund ist folgender. Wir haben

$$P(\mathcal{E}|F) = 0.246, \qquad P(\mathcal{E}|U) = 0.201 \tag{113}$$

Wir suchen jetzt $P(F|\mathcal{E})$. Nach Bayes Theorem gilt

$$P(F|\mathcal{E}) = P(\mathcal{E}|F) \cdot \frac{P(F)}{P(\mathcal{E})} = 0.246 \cdot \frac{P(F)}{P(\mathcal{E})} \tag{114}$$

Wir sehen jetzt: wir kommen nicht weiter ohne $P(F)$. Nehmen wir also einfach an, dass

$$P(F) = P(U) = 0.5$$

Wir brauchen aber noch die Wahrscheinlichkeit $P(\mathcal{E})$; allerdings kennen wir nur $P(\mathcal{E}|F)$ und $P(\mathcal{E}|U)$! Hier rettet uns die Annahme, dass $P(F \cup U) = 1$ (es gibt keine dritte mögliche Wahrscheinlichkeitsverteilung), und die Tatsache dass $F \cap U = \emptyset$; das bedeutet $\{F, U\}$ ist eine **Partition** der Ergebnismenge! Also gilt:

$$\begin{aligned} P(\mathcal{E}) &= P((\mathcal{E} \cap F) \cup (\mathcal{E} \cap U)) \\ &= P(\mathcal{E} \cap F) + P(\mathcal{E} \cap U) \\ &= P(\mathcal{E}|F)P(F) + P(\mathcal{E}|U)P(U) \\ &= 0.246 \cdot (0.5) + 0.201 \cdot (0.5) \\ &= 0.2235 \end{aligned}$$

Da wir nun die Wahrscheinlichkeit $P(\mathcal{E})$ haben, können wir auch $P(F|\mathcal{E})$ und $P(U|\mathcal{E})$ ausrechnen:

$$P(F|\mathcal{E}) = P(\mathcal{E}|F) \cdot \frac{P(F)}{P(\mathcal{E})} = 0.246 \cdot \frac{0.5}{0.2235} \approx 0.55 \tag{115}$$

Daraus folgt dass

$$P(U|\mathcal{E}) \approx 1 - 0.55 = 0.45$$

(wir können das natürlich auch einfach nachprüfen, indem wir in der letzte Gleichung U statt F verwenden.)

Das ist ein einfaches Beispiel von sogenannter **Bayesianischer Statistik**. Bayesianische Statistik ist sehr elegant und liefert uns genau die Informationen die wir suchen. Es gibt allerdings einige Probleme: das größte sind die beiden Annahmen, die wir auf dem Weg machen mussten (NB: in komplexeren Beispielen ist es noch viel schwieriger, plausible Annahmen, die sogenannten *priors*, die *a priori* Wahrscheinlichkeiten zu finden; und selbst in unserem sehr einfachen Beispiel wird man kaum sagen können dass unsere Annahmen sehr plausibel waren). Wir haben z.B. angenommen dass

$$P(F) = P(U) = 0.5.$$

Problem 1 ist: wenn wir etwas anderes angenommen hätten, z.B.

$$P(F) = 0.8,\ P(U) = 0.2,$$

dann hätten sich auch unsere *a posteriori* Wahrscheinlichkeiten für $P(F|\mathcal{E})$ und $P(U|\mathcal{E})$ geändert!

Problem 2 Wenn wir statt der zwei Wahrscheinlichkeitsverteilungen noch eine dritte zugelassen hätten, etwa

$$p_3(K) = 0.3,\ p_3(Z) = 0.7,$$

dann hätte auch das unser Ergebnis radikal verändert (das können Sie selbst nachprüfen); die Rechnung bleibt essentiell dieselbe, nur einige der Faktoren ändern sich. Daneben gibt es noch eine Reihe technischer Probleme, die in kompizierteren Beispielen entstehen (insbesondere bei stetigen Wahrscheinlichkeitsverteilungen).

Noch einige Anmerkungen sollte ich machen:

Anmerkung 1 Trotz dieser Probleme ist Bayesianische Statistik wesentlich informativer als alle klassische Statistik: denn wir haben die Wahrscheinlichkeit von Hypothesen (d.h. Wahrscheinlichkeitsverteilungen) gegeben eine Menge von Daten, und das ist mehr als uns die klassische Statistik liefern wird.

Anmerkung 2 Für den Bayesianer sind Wahrscheinlichkeiten keine Grenzwerte von relativen Häufigkeiten (der sog. Frequentismus), sondern sie quantifizieren Glauben. D.h. eine Wahrscheinlichkeit von 1 heißt: ich bin vollkommen überzeugt, nichts wird mich von meinem Glauben abbringen; eine Wahrscheinlichkeit von 0 bedeutet: nichts wird mich davon überzeugen das etwas wahr ist. Man sieht das auch am obigen Beispiel: wenn ich die *a priori* Wahrscheinlichkeit auf $P(U) = 0$ setze, dann wird die *a posteriori* Wahrscheinlichkeit $P(U|\mathcal{E})$ ebenfalls immer 0 sein.

Anmerkung 3 Es gibt hier eine interessante Beobachtung: Schizophrene und Menschen, die zu Verschörungstheorien neigen, haben die gemeinsame Eigenschaft, dass sie sich auf Hypothesen festlegen *bevor* sie ausreichende Evidenz haben. Das wurde experimentell belegt durch den sog. Teichtest: es gibt zwei Teiche, eine mit 40/60 blauen/orangen Fischen, einer mit 60/40

blauen/orangen Fischen. Die Teilnehmer sollten entscheiden, an welchem Teich sie sitzen nachdem sie (ihrer Meinung nach) ausreichende Evidenz durch ihren Fang hatten. Hier die Studie:
`https://www.frontiersin.org/articles/10.3389/fpsyt.2020.568942/full`

RX5a Übung

Erster Teil Wir nehmen an, wir haben die beiden Verteilungen F, U, wobei $P(Z|F) = 0.5$, $P(Z|U) = 0.6$. *Apriori*-Wahrscheinlichkeit von beiden ist 0.5. Berechnen Sie die aposteriori Wahrscheinlichkeit

1. $P(F|\mathcal{E}_1)$, wobei $\mathcal{E}_1$ ist: 45 mal Kopf aus 100 Würfen
2. $P(F|\mathcal{E}_2)$, wobei $\mathcal{E}_2$ ist: 55 mal Kopf aus 100 Würfen

Zweiter Teil Jetzt nehmen Sie einmal an, es gibt noch eine dritte mögliche Verteilung: U_2 hat folgende Parameter, nämlich $P(K|U_2) = 0.7$, $P(Z|U_2) = 0.3$. Wir passen auch die apriori Verteilungen an:

$$P(F) = 0.45,\ P(U) = 0.45,\ P(U_2) = 0.1$$

($P(Z|U), P(Z|F)$ wie oben.) Nehmen Sie

- $\mathcal{E}_3$: 6 mal Kopf aus 10 Würfen
- $\mathcal{E}_4$: 55 mal Kopf aus 100 Würfen ($= \mathcal{E}_2$)
- $\mathcal{E}_5$: 550 mal Kopf aus 1000 Würfen

Berechnen Sie nun

3. $P(F|\mathcal{E}_i)$, $P(U|\mathcal{E}_i)$, $P(U_2|\mathcal{E}_i)$ für $i \in \{3, 4, 5\}$.

Dritter Teil Abstrahieren Sie: schreiben Sie eine Funktion f mit Eingabe (k, n): n die Anzahl der Würfe, k die Anzahl von Kopf. Ausgabe sind die *aposteriori* Wahrscheinlichkeiten $P(F|\mathcal{E})$, $P(U|\mathcal{E})$, $P(U_2|\mathcal{E})$ ($\mathcal{E}$ das Eingabeereignis).

Nachtrag zu Datenmengen Eine wichtige Notiz ist folgende: wir können in diesem – und jedem streng wahrscheinlichkeitstheoretischen – Ansatz keine falschen Schlüsse aufgrund "zuwenig Daten" ziehen. Die Anzahl der Daten ist als Evidenz eingepreist in die aposteriori Wahrscheinlichkeit. Zuwenig Daten bedeutet: wir haben kein eindeutiges Bild welche Hypothese besser ist. Wenn wir ein eindeutiges Bild haben bei wenig Daten, dann liegt das daran, dass die Daten extrem eindeutig sind!

Das ist was ich meine: wer auf dem Pfad der reinen Lehre wandelt, der wird nicht ins Dunkel gehen!

13.2 Sequentielle Hypothesenprüfung – Bayesianisch

Das grosse Problem, das der orthodoxen Statistiken zugrunde liegt, ist das wir eine **Asymmetrie** haben: wir nutzen unsere Experimente nur, um eine Hypothese (üblicherweise die Nullhypothese) zu *falsifizieren*; wir verifizieren aber grundsätzlich nichts. Das ist anders im Bayesianischen Ansatz: im obigen Beispiel hatten wir zwei Hypothesen

$$U, F,$$

und deren jeweilige Wahrscheinlichkeiten *apriori* und *aposteriori*. Dieses Szenario ist symmetrisch zwischen den beiden Hypothesen, und es ist tatsächlich so, dass in diesem Fall die sequentielle Heransgehensweise durchaus legitim ist. Das sieht in diesem Fall wie folgt aus: wir haben unsere bekannten *apriori* Wahrscheinlichkeiten

$$P(F) = P(U) = 0.5$$

Zur Erinnerung:

$$P(K|U) = 0.4 \qquad P(K|F) = 0.5$$

Nun werfen wir Kopf (K). Daraus bekommen wir unsere neuen Wahrscheinlichkeiten

$$P(F|K);\ P(U|K)$$

Das lässt sich nach obigem Muster leicht ausrechnen; eine kurze Überlegung liefert uns:

$$P(F|K) > P(U|K)$$

denn $P(K|F) > P(K|U)$. Nun können wir das iterieren:

> Berechne $P(F|K_1K_2)$; $P(U|K_1K_2)$
> Berechne $P(F|K_1K_2Z_3)$; $P(U|K_1K_2Z_3)$
> ...

Es ist klar dass jedes K die Wahrscheinlichkeit von F erhöht, jedes Z die Wahrscheinlichkeit von U. Weiterhin legen wir eine Gewissheits-Konstante $c \in [0, 1]$ fest, so dass für uns gilt: falls wir eine Folge $\vec{W} \in \{K, Z\}^*$ beobachten, so dass

$$(\#) \text{ entweder } P(F|\vec{W}) > c \text{ oder } P(U|\vec{W}) > c,$$

dann hören wir auf und akzeptieren die Hypothese, deren Wahrscheinlichkeit größer c ist. Das macht natürlich nur Sinn, falls c nahe bei 1 liegt, z.B.

$$c = 0.99$$

Dieses Vorgehen einwandfrei – warum? Weil wir beide Hypothesen gleichermaßen berücksichtigen!

Dagegen: Unsere vorige Herangehensweise (Vertrauensgrenze) wäre vergleichbar mit: wir legen eine Konstante $c \in [0, 1]$ fest, so dass falls wir eine Folge $\vec{W} \in \{K, Z\}^*$ beobachten, dass

$$(\#') \ P(U|\vec{W}) > c$$

dann hören wir auf und akzeptieren U. Nun ist vollkommen klar dass auf diese Art und Weise die Hypothese U unfair bevorzugt wird, denn alle beide können an einer gewissen Stelle sehr wahrscheinlich sein.

Hausaufgabe 7

Bitte bearbeiten bis zum 7.1.21.

Schreiben Sie eine Funktion f in R, die als eine Eingabe eine beliebige Folge (entscheiden Sie welches Datenformat) $\vec{W}$ von K, Z gibt (oder einfacher: 0,1), und Ihnen als Ausgabe die *aposteriori*-Wahrscheinlichkeit $P(F|\vec{W})$ liefert (mit Verteilungen F, U und *apriori*-Wahrscheinlichkeit so wie oben).

Ansatz: nutzen Sie einfach die üblichen Formeln, um $P(\vec{W}|F)$ und $P(\vec{W}|U)$ zu berechnen; dann müssen Sie nur noch den Satz von Bayes applizieren!

13.3 RX5b Die effektive Berechnung mit dynamischer Programmierung

Das rechnerische Problem bei der sequentiellen Bayesianischen Hypothesenprüfung ist, dass wir erstmal keine Berechnungen "recyclen" können, d.h. wir müssen immer wieder neu von vorne anfangen. Zunächst schreiben wir etwas allgemeiner:

$$P(F|X_1...X_i),\ P(U|X_1...X_i),$$

wobei $X_1, ..., X_i$ jeweils Variablen sind für ein beliebiges Ereignis $K_1/Z_1, ..., K_i/Z_i$. Nun können wir etwas mathematischer sagen: $P(F|X_1...X_{i+1})$ lässt sich erstmal nicht auffassen als Funktion

$$P(F|X_1...X_{i+1}) = f_{X_{i+1}}(P(F|X_1...X_i))$$

für ein einfaches $f_{X_{i+1}}$, was steht für entweder f_K oder f_Z, je nach Ergebnis.

Dem entspricht ein theoretisches Problem: wir möchten Hypothesen prüfen, indem wir der Reihe nach

$P(F|K_1K_2);\ P(U|K_1K_2)$
$P(F|K_1K_2Z_3);\ P(U|K_1K_2Z_3)$
...

errechnen, und warten dass eine der beiden Wahrscheinlichkeiten einen (vorher festgelegten) Grenzwert überschreitet). Allerdings: nimm an wir in jedem Schritt weiterhin mit den *apriori* Wahrscheinlichkeiten $P(F), P(U)$, obwohl wir es eigentlich besser wissen, also eigentlich bereits

$$P(F|K_1),\ P(F|K_1K_2) \text{ etc.}$$

kennen. Das verstößt gegen einen zentralen Grundsatz der Wahrscheinlichkeitstheorie und Statistik:

> **wir dürfen niemals relevante Information, die uns bekannt ist, außer Acht lassen**.

Denn sonst wäre ja der Willkür Tür und Tor geöffnet, Informationen zu ignorieren, die relevant ist, aber nicht das von uns gewünschte Ergebnis unterstützt. Im Grunde liegt also auch hier ein Missbrauch vor!

Das Problem ist also das folgende: die Berechnung von $P(F|K_1)$ hängt ab von der *apriori*-Wahrscheinlichkeit $P(F)$. Wenn nun das Ereignis K_1

gegeben ist, ändert sich unsere Einschätzung der Wahrscheinlichkeit von F und U. Das wiederum führt dazu, dass z.B.

(116) $P(K_2|K_1) \neq P(K_2)$,

etwas allgemeiner: lassen wir $X_1, X_2, ...$ als Variablen stehen, so dass X_i den Wert K_i oder Z_i annehmen können. Dann gilt:

(117) $P(X_2|X_1) \neq P(X_2); \quad P(X_3|X_2X_1) \neq P(X_3)$ etc.

Warum ist das so? Wir haben oben festgestellt das z.B.

(118) $P(X_2) = P(X_2|F)P(F) + P(X_2|U)P(U)$

Wir nutzen nun eine allgemeinere Formulierungen unserer Regeln (wir zeigen das exemplarisch am Beispiel $X = K$):

(119) $P(K_2|K_1) = P(K_2|FK_1)P(F|K_1) + P(K_2|UK_1)P(U|K_1)$

Überlegungen des gesunden Menschenverstandes sagen uns, dass

(120) $$\begin{array}{l} P(K_2|FK_1) = P(K_2|F) \\ P(K_2|UK_1) = P(K_2|U) \end{array}$$

Das Problem ist:

(121) $$\begin{array}{l} P(F|K_1) \neq P(F) \\ P(U|K_1) \neq P(U) \end{array}$$

Wir können das jetzt noch etwas genauer formulieren, denn wir wissen:

(122) $$\begin{array}{l} P(F|K_1) > P(F) \\ P(U|K_1) < P(U) \end{array}$$

Dasselbe gilt natürlich in die andere Richtung für $P(F|Z_1)$ etc. Das bedeutet wenn wir wirklich sequentiell die Hypothese prüfen, dann müssten wir eigentlich auch jedesmal die Wahrscheinlichkeit von F und U *updaten* (d.h. neu errechnen), und mittels dieser Wahrscheinlichkeit die Wahrscheinlichhkeit der Ergebnisse K, Z neu berechnen.

Wir definieren nun eine neue Variable Y, die die Werte F, U annehmen kann, also

$$Y \in \{F, U\}$$

Damit lassen sich die folgenden Überlegungen allgemeiner formulieren. Man beachte dass

$$P(Y|X_1...X_iX_{i+1}) = P(X_{i+1}|Y, X_1...X_i)\frac{P(Y|X_1...X_i)}{P(X_{i+1}|X_1...X_i)} \quad (123)$$

$$P(X_{i+1}|Y, X_1...X_i) = P(X_{i+1}|Y) \quad (124)$$

Damit wird nun die Arbeit leichter. Wir definieren nun die Wahrscheinlichkeitsfunktion P_{seq} der sequentiellen Prüfung wie folgt:

1. $P_{seq}(Y|X_1) = P(Y|X_1)$
2. $P_{seq}(Y|X_1...X_{i+1}) = P(X_{i+1}|Y)\frac{P_{seq}(Y|X_1...X_i)}{P_{seq}(X_{i+1}|X_1...X_i)}$

Das sieht schon wesentlich besser aus: wir benutzen hier sämtliche Information die uns zur Verfügung steht. Auch die Berechnung wird wesentlich einfacher:

$$P(X_{i+1}|Y)$$

(das eigentlich nur eine Kurzschreibweise für $P(X_{i+1}|YX_1...X_i)$ ist) ist eine konstante c_Y;

$$P_{seq}(Y|X_1...X_i)$$

mussten wir (nach Annahme) bereits ohnehin ausrechnen; bleibt noch der Term

$$P_{seq}(X_{i+1}|X_1...X_i);$$

das lässt sich wie folgt berechnen:

$$P_{seq}(X_{i+1}|X_1...X_i) = P(X_{i+1}|F)P_{seq}(F|X_1...X_i)+P(X_{i+1}|U)P_{seq}(U|X_1...X_i) \quad (125)$$

Das vereinfacht sich zu

$$P_{seq}(X_{i+1}|X_1...X_i) = c_F P_{seq}(F|X_1...X_i) + c_U P_{seq}(U|X_1...X_i) \quad (126)$$

Am Ende bekommen wir also:

$$(127)\quad P_{seq}(Y|X_1...X_{i+1}) = P_{seq}(Y|X_1...X_i)\frac{P(X_{i+1}|Y)}{P_{seq}(X_{i+1}|X_1...X_i)}$$

Das ist ein relativ zufriedenstellendes Ergebnis (das man mit etwas komplexeren Methoden noch besser ausgestalten kann). Wir haben nun unser Ziel erreicht:

$$(128)\quad P(Y|X_1...X_{i+1}) = f_{X_{i+1}}(P(Y|X_1...X_i))$$

wobei f eine relativ einfache Funktion ist. Außerdem benutzen wir in jedem Schritt alle relevanten Informationen. Aber aus diesem Ergebnis lassen sich noch mehr interessante Folgerungen ableiten. Auf den ersten Blick lässt sich folgendes sagen: Wir haben

$$\begin{array}{rcl} P_{seq}(Y|X_1...X_{i+1}) & > & P_{seq}(Y|X_1...X_i) \\ & \Leftrightarrow & \\ \frac{P(X_{i+1}|Y)}{P_{seq}(X_{i+1}|X_1...X_i)} & > & 1 \\ & \Leftrightarrow & \\ P_{seq}(X_{i+1}|X_1...X_i) & < & P(X_{i+1}|Y) \end{array}$$

Nun ist in unserem Fall leicht zu sehen dass immer gilt:

$$(129)\quad P_{seq}(K_{i+1}|X_1...X_i) < P(K|F)$$

und

$$(130)\quad P_{seq}(Z_{i+1}|X_1...X_i) > P(Z|F)$$

d.h. wir haben die Bestätigung dafür, dass jeder Wurf von Kopf die Wahrscheinlichkeit von F erhöht, umgekehrt jeder Wurf von Zahl die Wahrscheinlichkeit von U.

Eine wichtige Frage, die wir hier offengelassen haben, ist was es *bedeutet*, also wie sich P_{seq} von P unterscheidet. Das lässt sich wie folgt beantworten: es gilt:

$$\begin{array}{rcl} P(X_{i+1}|F) & = & P_{seq}(X_{i+1}|X_1...X_i) \\ & \Leftrightarrow & \\ P(U|X_1...X_i) & = & 0 \\ & \Leftrightarrow & \\ P(F|X_1...X_i) & = & 1 \end{array}$$

Dass wir allerdings tatsächlich

$$P(F|X_1...X_i) = 1$$

haben ist theoretisch ausgeschlossen, denn es gibt kein Ergebnis, das mit $P(U)$ wirklich inkompatibel wäre. Dennoch bedeutet dass:

$$\underset{P(F|X_1...X_i)}{argmin} \frac{P(X_{i+1}|F)}{P(X_{i+1}|X_1...X_i)} = 1 \tag{131}$$

(es soll uns nicht stören dass $P(F|X_1...X_i)$ nicht explizit vorkommt in dem Term; wir wissen ja es ist implizit vorhanden) Umgekehrt sieht man aus demselben Grund dass

$$\underset{P(F|X_1...X_i))}{argmax} \frac{P(X_{i+1}|F)}{P(X_{i+1}|X_1...X_i)} = 0 \tag{132}$$

Das bedeutet in Worten:

> Je unwahrscheinlicher F (bzw. U) ist gegeben unsere bisherige Beobachtungen, desto stärkere relative Evidenz liefert eine Beobachtung von K (bzw. Z) für F (bzw. U).

Dasselbe gilt natürlich auch andersrum:

> Je wahrscheinlicher F (bzw. U) ist gegeben unsere bisherige Beobachtungen, desto schwächere relative Evidenz liefert eine Beobachtung von K (bzw. Z) für F (bzw. U).

Das kann man sich intuitiv wie folgt klar machen: wenn wir F bereits für sehr wahrscheinlich halten, dann liefert uns eine Evidenz für F nur geringe neue Information, eine Evidenz für U aber deutlich mehr. Wir haben hier übrigens eine typische *invers exponentielle Sättigungskurve*; in der logarithmischen Transformation wird das also zu einer einfachen Addition (siehe Jaynes: Probability Theory, Kapitel 4)!

Hausaufgabe 8

Bearbeiten bis zum 14.6.21 vor dem Seminar!

1. Nehmen Sie an, Sie werfen die Folge ZKZKZK... etc. Wie viele Würfe wird es dauern bis Sie, für ihr Ergebnis ω_n, $P(F|\omega_n) > 0.95$ haben (also das kleinste n)? Implementieren Sie ein Programm, für das $c = 0.95$ eine Variable ist, und für jedes c das korrekte Ergebnis auswirft.

2. Was wäre die kürzeste Folge von Würfen ω, mittels derer Sie $P(F|\omega) > 0.95$ bekommen? Modifizieren Sie Ihr obiges Programm, um das Ergebnis zu berechnen.

3. Nehmen Sie an, es gibt 3 Hypothesen, $P(F) = P(U) = 0.49$, $P(I) = 0.02$, wobei $P(K|I) = 0.99$. Wie ändert sich das Szenario in Bezug auf die vorigen Aufgaben? Nehmen Sie die (gegebene) Funktion bayesSeqII, und ändern Sie sie, so dass Sie die aposteriori Wahrscheinlichkeit von F (für beliebige Eingabesequenz ω) unter diesen apriori Annahmen berechnet.

Lösungsansatz zu 1: Wir nehmen die Gleichungen, die P_{seq} definieren. Wir definieren:

$$P_0^F := 0.5 \tag{133}$$

$$P_0^U := 0.5 \tag{134}$$

$$P_1^F := P_0^F \frac{P(Z|F)}{P(Z|F)P_0^F + P(Z|U)P_0^U} \tag{135}$$

$$P_1^U := P_0^U \frac{P(Z|U)}{P(Z|F)P_0^F + P(Z|U)P_0^U} = 1 - P_1^F \tag{136}$$

Als nächstes berechnen wir:

$$P_2^F := P_1^F \frac{P(K|F)}{P(K|F)P_1^F + P(K|U)P_1^U} \tag{137}$$

$$P_2^U := P_1^U \frac{P(K|U)}{P(K|F)P_1^F + P(K|U)P_1^U} = 1 - P_2^F \tag{138}$$

Und so weiter, Kopf und Zahl alternierend:

$$P_3^F := P_2^F \frac{P(Z|F)}{P(Z|F)P_2^F + P(Z|U)P_2^U} \tag{139}$$

$$P_3^U := P_2^U \frac{P(Z|U)}{P(Z|F)P_2^F + P(Z|U)P_2^U} = 1 - P_3^F \tag{140}$$

Die Aufgabe besteht nun darin, dass kleinste n zu finden, so dass $P_n(F) > 0.95$ ist. Zu diesem Zweck macht es Sinn, einen Vektor zu erstellen

$$\text{vec}=(P_1, ..., P_n, ...)$$

und zu prüfen, an welcher Stelle er größer wird als 0.95. Dafür müssen wir die obige Rechnung automatisieren. Wir überlegen uns zuerst, welche Datenstruktur wir brauchen, anders gesagt: welche Information wir weitergeben müssen. Das sind folgende Dinge:

1. P_i^F (damit ist P_i^U bereits gegeben, also redundant).
2. ob i gerade oder ungerade ist, d.h. ob wir Kopf oder Zahl haben als nächstes.

Das wars auch schon! Wir haben also Paare der Form (r, X), wobei

1. $r \in [0, 1]$ und
2. $X \in \{Z, K\}$

Wir definieren jetzt eine Funktion f wie folgt:

$$f(r, Z) = (r \cdot \frac{0.5}{0.5r + 0.6(1-r)}, K) \tag{141}$$

$$f(r, K) = (r \cdot \frac{0.5}{0.5r + 0.4(1-r)}, Z) \tag{142}$$

Wir setzen nun $C_0 = (0.5, Z)$, und $C_{i+1} = f(C_i)$. An C_i können wir jeweils P_i^F ablesen.

Implementieren müssen Sie aber schon selber!

(Lösung auskommentiert)

Aufgabe 8

Abgabe bis zum 12.6.18 *vor dem Seminar*, egal ob digital/analog und auf welchem Weg.

Nehmen Sie an, Sie befinden sich im Urlaub einer großen Stadt. Sie haben (am Abreisetag) Ihre Koffer im Hotel deponiert, sind noch in der Stadt unterwegs. Sie müssen also an einem gewissen Punkt erst ins Hotel, dann zum Bahnhof, um Ihren Zug zu bekommen. Dazu müssen Sie *5mal* umsteigen; die Zeiten für Fahrten, den Weg von einem Gleis zum andern etc. sind Ihnen bekannt; die einzige unbekannte sind die genauen Fahrtzeiten: sie wissen nur dass die Bahnen einheitlich alle *10 Minuten* fahren, so dass Sie also schlimmstenfalls 50min reine Wartezeit für diese Reise zu erwarten haben. Natürlich möchten Sie die Abfahrt so lang als möglich hinauszögern. Aber wieviel Zeit veranschlagen Sie als reine Wartezeit auf Anschlusszüge für Ihre Reise *jetziger Aufenthalt*$\rightarrow$*Hotel*$\rightarrow$*Bahnhof*?

1. 50min sind natürlich das Maximum, aber es ist sehr unwahrscheinlich dass Sie so lange warten müssen. Stattdessen beschließen Sie folgendes: Sie möchten mit einer Wahrscheinlichkeit von 0.95 Ihren Zug erwischen. Wieviel Zeit veranschlagen Sie unter dieser Prämisse für das Warten auf Anschlußzüge? Der Einfachheit halber nehmen wir hier immer an, dass Zeit diskret im Minutentakt abläuft.

2. Nehmen Sie an, statt 5mal Umsteigen mit einer Bahn alle 10min müssen Sie *10mal* Umsteigen, aber die Bahnen fahren alle *5min*. Mit ansonsten denselben Prämissen wie in 1., wie verändert sich Ihre Einschätzung, veranschlagen Sie mehr oder weniger Zeit für eine Sicherheit von 0.95? Begründen Sie! Sie können das natürlich explizit ausrechnen; es gibt aber auch eine einfache gute Begründung, wenn Sie unsere bisherigen Erkenntnisse zu Vertrauensgrenzen, Varianz etc. nutzen.

14 Einseitige Tests

14.1 Ein zweites Beispiel – Fehlerquoten

Im vorigen Beispiel ging es um *Sicherheit* – wir wollen sicher sein (bis zu einem gewissen Punkt), dass wir den Zug erwischen. Einen ähnlichen Fall haben wir im folgenden Beispiel, auch wenn die Dinge etwas anders gelagert sind: nehmen wir an, wir produzieren einen Gegenstand, z.B. Achsen für ein Auto. Wir möchten sicher sein, dass es eine Fehlerquote ϵ gibt, so dass gilt:

$$\frac{\text{Anzahl der fehlerhaft produzierten Achsen}}{\text{Anzahl der insgesamt produzierten Achsen}} < \epsilon \tag{143}$$

Den linken Term nennen wir die **Fehlerquote** f. Das ist eine wichtige Kenngröße, denn fehlerhafte Teile können immer produziert werden; was entscheidend ist, ist Wissen um deren Quote. Die genaue Quote können wir nicht kennen, da Materialtests aufwändig sind und teilweise das Material zerstören. Also suchen wir folgendes:

> Wir legen 2 Konstanten q, ϵ fest, und möchten verifizieren mit einer Sicherheit $s > q$ die Fehlerquote $f < \epsilon$ ist.

Das bedeutet: die Wahrscheinlichkeit, dass $f \geq \epsilon$ ist, soll $< 1 - q := p$ sein. Wir haben hier also ein klassisches Problem von p-Werten und Vertrauensgrenzen. Dem ganzen liegt in diesem Fall auch eine einfache, wenn auch asymmetrische Binomialverteilung zugrunde, von daher haben wir eine einfache arithmetische Formel für unsere Rechnungen. Z.B. legen wir fest:

$$\epsilon = 0.001,\ q = 0.95,\ \text{also } p = 0.05$$

Nun machen wir eine **Stichprobe** $\mathfrak{S}$ von sagen wir 10,000 Objekten, und davon sind 3 defekt. Wir machen nun folgendes: wir nehmen als H_0 an, dass

$$f = 1/1,000$$

(wir werden das später hinterfragen). Da unser Erwartungswert (Binomialverteilung!) für eine Probe der Größe 10,000 nun bei 10 liegt, ist das erstmal ein gutes Ergebnis. Aber ist es gut genug?

Wir berechnen (wie immer bei p-Werten) die Wahrscheinlichkeit, dass unser Ergebnis wie gehabt *oder noch extremer* ausfällt, also für uns: wie

ist die Wahrscheinlichkeit, 3 oder noch weniger defekte Teile auf 10,000 zu finden? Dafür haben wir eine einfache Formel:

$$(144)\ \sum_{i=0}^{3} \binom{10,000}{i} (\frac{999}{1,000})^{10,000-i} \cdot (\frac{1}{1,000})^{i}$$

In R lässt sich das einfach berechnen:

```
> int=0:3
> for(i in 0:3){int[i+1]<- choose(10000,i)*
(999/1000)^(10000-i)*(1/1000)^i}
> int
[1] 4.517335e-05 4.521856e-04 2.262965e-03 7.549258e-03
> sum(int)
[1] 0.01030958
```

Das bedeutet also: unser Ergebnis ist (nach unseren Annahmen) signifikant; die Wahrscheinlichkeit unserer Stichprobe (oder einer noch extremeren Probe) liegt bei 0.01. Das heißt: das Ergebnis liegt außerhalb der Vertrauensgrenzen, wir weisen H_0 zurück.

Aber: was ist das für ein Nullhypothese? Sie ist ja in keinem besonderen Sinne neutral, und wir können ja nicht beliebig Hypothesen annehmen, um sie dann zurückzuweisen. Die Rechtfertigung für diese Methode ist die folgende: gegeben unsere Annahmen war H_0 diejenige Hypothese, die die Daten *am wahrscheinlichsten* macht; denn wir wollten sicher gehen dass

$$f < 1/1,000,$$

und das Ergebnis unserer Stichprobe war jenseits des Erwartungswertes in diesem Fall.

- Für jede andere Wahrscheinlichkeit H_0', die besagt dass $P(f) > 1/1,000$, wäre unsere Stichprobe noch unwahrscheinlicher gewesen, unser Ergebnis also noch signifikanter ausgefallen!

Wir haben also H_0 angenommen als diejenige Hypothese, die die Daten am wahrscheinlichsten macht, und wenn wir diese Hypothese zurückweisen, dann weisen wir auch jede andere Hypothese zurück dass die Fehlerquote $> 1/1,000$ ist. Wir weisen also H_0 stellvetretend für alle Hypothesen zurück,

nach denen die Fehlerquote zu hoch ist. Dementsprechend haben wir gezeigt, was wir zeigen wollen.

NB: das funktioniert natürlich nur, wenn in unserer Stichprobe $\mathfrak{S}$ die Fehlerquote *niedriger* ist als der Erwartungswert unter H_0 – sonst wäre das ganze offensichtlich Unsinn. In diesem Fall kann man Fall kann man p-Werte auch bei asymmetrischen Verteilungen nutzen, man muss aber vorsichtig sein, dass das Vorgehen sinnvoll bleibt.

Die Vertrauensgrenze liegt übrigens bei 4: wenn wir mehr als vier fehlerhafte Teile finden, dann können wir H_0 nicht mit Signifikanz p zurückweisen (für 4 gilt $p = 0.02919595$), für 5 haben wir $p = 0.06699137$). Das lässt sich leicht mit obiger Formel errechnen, es reicht i entsprechend einzusetzen.

Übung Oben haben wir die Annahme, dass wir die Stichprobe von 10000 aus einer unendlichen Population ziehen, also die Wahrscheinlichkeit von (nicht) fehlerhaften Achsen immer gleich bleibt. Wir machen nun eine andere Annahme: wir haben eine Population von 100000 Achsen; wir machen eine Stichprobe von 1000. Wir setzen $\epsilon = 1/100$, $q = 0.99$. Wo liegt die Vertrauensgrenze? (Tipp: Sie brauchen die hypergeometrische Verteilung!)

15 RX6 Wahrscheinlichkeitsdichte, Wahrscheinlichkeitsmasse, Quantile

Wir haben bislang in erster Linie die Wahrscheinlichkeitsdichte betrachtet. Beispiele waren die **Binomialverteilung**:

$$(145)\quad f_X(x) = \begin{cases} \binom{n}{x} p^x q^{n-x}, \text{ falls } x \in \{0, 1, ..., n\} \\ 0 \text{ andernfalls} \end{cases}$$

die **hypergeometrische Verteilung**:

$$(146)\quad P(X = x) = \frac{\binom{M}{k}\binom{N-M}{n-k}}{\binom{N}{n}}$$

die **Normalverteilung**:

$$(147)\quad f(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2\right)$$

und die **geometrische Verteilung**:

$$(148)\quad P_r(n) = (1-r)r^{n-1}$$

Die Wahrscheinlichkeitsdichte liefert uns die Wahrscheinlichkeit für jeden Punkt. Jede Wahrscheinlichkeitsdichte hat eine zugeordnete **Wahrscheinlichkeitsmassefunktion**. Die ist sehr einfach definiert, man muss aber zwei Fälle unterscheiden, nämlich

1. Die zugrundeliegende Dichte ist diskret (Binomial, Hypergeometrisch)
2. Die zugrundeliegende Dichte ist stetig (normal, geometrisch)

Nimm an, wir haben eine Funktion f_1 die eine diskrete Verteilung ist; dann definieren wir

$$(149)\quad pmf(f_1)(x) = \sum_{y:y\leq x} f(y)$$

Falls f_2 eine stetige Verteilung ist, dann gilt:

$$(150)\quad pmf(f)(x) = \int_{-\infty}^{x} f(x)$$

Wir bekommen auf diese Form eine Funktion, die monoton steigt, und falls die zugrundeliegende Dichte niemals 0 ist, steigt sie streng monoton.

Wahrscheinlichkeitsdichtefunktionen sind in R fertig erhältlich mit dem Präfix p; d.h. wir haben pbinom(), phyper(), pnorm() etc., anstelle des bisher benutzten d. Die Dichtefunktionen sind für uns ziemlich nützlich wenn wir beispielsweise die linke Grenze der Verteilung funden müssen, wo die Summe der Werte links der Grenze größer ist als p, wie bei den einseitigen Vertrauensgrenzen. Also wird die obige Aufgabe zu:

> Suche größte n, so dass pbinom(n,...)<0.025, und das kleinste n' so dass pbinom(n',...)>0.975.

Das kann man mit einer while-Schleife ermitteln. Man kann das aber auch direkter machen, nämlich mit der **Quantilfunktion**. Nimm an, pf ist eine steige Wahrscheinlichkeitsmassefunktion von f, die streng monoton ist, so wie alle unsere Beispiele. Das bedeutet insbesondere: pf ist eine **Bijektion**, also eine 1 zu 1 Abbildung. Das bedeutet es gibt eine eindeutige Umkehrfunktion

(151) $pf^{-1}(x) = y$ gdw. $pf(y) = x$

Diese Funktion ist die Quantilfunktion von f. Sie liefert uns, gegeben eine Zahl $x \in [0, 1]$, dasjenige y so dass $pf(y) = x$, also die genaue einseitige Vertrauensgrenze von f. Wir nennen diese Funktion $qf()$.

Im diskreten Fall ist das etwas schwieriger, denn hier ist die Umkehrfunktion nicht eindeutig definiert. Was man macht ist: man rundet auf. Sei also f_1 eine diskrete Verteilung. Dann haben wir:

(152) $qf_1(y) = min\{x : f(x) > y\}$

Quantilfunktionen gibt es in R mit dem Präfix q, also qbinom(),qhyper(),qnorm(). Was wir oben als Aufgabe gestellt haben ist also sehr einfach lösbar mit n =qbinom(0.025,1000,0.6).

16 Mehrere Variablen und Tests für Unabhängigkeit

16.1 Vorspiel: Parameter schätzen

Sei also $\mathfrak{T}$ ein Text der Form

$$w_1 w_2 w_3 w_4 w_5 w_6 ...$$

wobei das Subskript nur etwas über die Position des Wortes sagt, nicht seine Identität. Der Test liefert uns viele Information, z.B. für alle vorkommenende *types* die Anzahl der *token*. Wir listen die Types mit

$$L = \{l_1, l_2, l_3, ...\}$$

mit l wie Lexikon. NB:

$$S : \mathfrak{T} \to L \times \mathbb{N}$$

wobei jedem Wort seine Häufigkeit zugewiesen wird, ist nach unserer Definition eine Statistik (aber es ist nicht gesagt, dass sie ausreichend ist!).

Wir nehmen folgende Konventionen:

- $|\mathfrak{T}|$ ist die Länge des Textes (Gesamtzahl der token).
- $|\mathfrak{T}|_w$ die Haüfigkeit des Wortes w im Text (Anzahl der token dieses types).
- Wir schreiben auf $f(w)$ anstatt $|\mathfrak{T}|_w$, und
- $f(w|v)$ für $|\mathfrak{T}|_{vw}$. Das zählt also: wie oft folgt w auf v.

Mit diesen Zahlen können wir Wahrscheinlichkeiten als relative Häufigkeiten schätzen.

Unabhängige Wahrscheinlichkeit:

$$\hat{P}_0(w) = \frac{|\mathfrak{T}|_w}{|\mathfrak{T}|} \tag{153}$$

Abhängige Wahrscheinlichkeit:

$$\hat{P}_1(w|v) = \frac{|\mathfrak{T}|_{vw}}{|\mathfrak{T}|_v} \tag{154}$$

Das sind die geschätzten Wahrscheinlichkeiten in unserem Text.

H_0 wäre: $\hat{P}_0$ (oder etwas in dieser Art) ist die zugrundeliegende Verteilung, also:

(155) $P_0(w_1w_2w_3...) = \hat{P}_0(w_1) \cdot \hat{P}_0(w_2) \cdot \hat{P}_0(w_3) \cdot ....$

Hingegen ist H_1 die Hypothese, dass wir Markov-Abhängigkeiten erster Stufe haben, also:

(156) $P_1(w_1w_2w_3...) = \hat{P}_1(w_1|\#) \cdot \hat{P}_1(w_2|w_1) \cdot \hat{P}_1(w_3|w_2) \cdot ....$

Es ist natürlich klar dass normalerweise

(157) $P_0(\mathfrak{T}) < P_1(\mathfrak{T})$

denn P_1 ist spezifischer.

16.2 RX7 Der Schwellentest

Eine typische Situation in der Statistik ist die folgende: wir haben einen gewissen Datensatz; nehmen wir z.B. an, wir haben einen gewissen Text (Datensatz) D.

> MDNGHRKENGNSKRNSHREHWEJFVBNBFJSKEWRNDSJXYHD
> NWIDHEJKDNXHWKJDHJAJDWREHFKVJCJFHRNENDKXMDJ
> EYUHWNDJFD...

Das kann für etwas beliebiges stehen; wir können auch annehmen, dass es sich um einen sprachlichen Text handelt (mit Wörtern), wobei z.B. B für das Leerzeichen steht. Gleichzeitig haben wir zwei **Sprachmodelle** M_0, M_1. Beide weisen dem Text T eine gewisse Wahrscheinlichkeit zu:

$$M_0(D),\ M_1(D)$$

Wir sollen nun entscheiden, welches Modell besser ist. Die einfache Lösung wäre folgende: wir nehmen einfach

$$max(M_0(D), M_1(D)),$$

und wählen die Hypothese entsprechend. Wenn wir beliebige Hypothesen zulassen, dann enden wir mit einer Art maximum-likelihood Schätzung. Das ist aber in vielen Fällen inadädquat, da durch diese Herangehensweise die Hypothesen *zu stark* durch die konkreten Daten bestimmt werden. Das gilt besonders dann, wenn gewisse Hypothesen stark unabängig motiviert sind, und wir nicht beliebige Hypothesen zur Auswahl haben. In unserem Zusammenhang können wir etwa folgendes Beispiel benutzen:

- M_0 ist ein Modell, in dem alle Wortwahrscheinlichkeiten unabhängig voneinander sind;
- M_1 ist ein Markov-Modell, in dem Wortwahrscheinlichkeiten sich wechselseitig beeinflussen (über einen beschränkten Raum hinweg).

Was uns also interessiert: welche Art der Modellierung ist plausibler?

Bevor wir diese Frage beantworten, müssen wir uns über eine grundlegende Asymmetrie zwischen M_0 und M_1 klarwerden. Es ist erstens klar, dass M_1 bessere Ergebnisse erzielt, wenn wir die Parameter so anpassen, dass sie genau auf D passen; aber das ist soz. trivial und nicht empirisch:

wir möchten eine allgemeine Aussage treffen, von der wir davon ausgehen dass sie auch für andere Texte ihre Gültigkeit hat; wir würden also gerne mit Parametern arbeiten, die allgemein und unabhängig von D geschätzt sind. Wir sollten also davon ausgehen dass die Parameter von M_0, M_1 beide auf unabhängigen Texten geschätzt wurden. Die obige Tatsache aber nur ein Sympton einer tieferliegenden Asymmetrie:

> M_1 ist *spezifischer* als M_0, es spezifiziert mehr Parameter, oder anders gesagt: die Ereignisse sind stärker voneinander abhängig als in M_0.

Aus dieser Tatsache wird – wissenschaftlichen Grundsätzen wie Ockhams Rasiermesser folgend (*entia non sunt multiplicanda*) – das bis auf weiteres M_0 gegenüber M_1 vorzuziehen ist. Wenn wir dennoch sagen, dass M_1 besser ist als M_0, dann brauchen wir dafür gute Gründe. Hier haben wir nun alle Zutaten eines klassischen statistischen Problems beisammen: wir haben zwei voll ausgearbeitete Hypothesen, die sich einteilen lassen in

1. eine **Nullhypothese** M_0 – üblicherweise H_0 geschrieben, und
2. eine alternative Hypothese M_1, üblicherweise H_1 geschrieben.

H_0 ist also die Hypothese, dass Worte im Text unabhänging voneinander sind, H_1 die Hypothese dass sie sich wie Markov-Ketten verhalten. Wir haben nun einen Datensatz D, und möchten uns **entscheiden**, gegeben D, welche der beiden Hypothesen im Allgemeinen vorzuziehen ist. Eine solche Entscheidungsfunktion nennt man einen **Test**.

Hierbei gibt es natürlich folgendes zu beachten: uns interessiert eine zugrunde liegende Wahrscheinlichkeitsverteilung, die erstmal nichts ausschließt. Das bedeutet wir können nicht mit Sicherheit die richtige Antwort finden; es kann immer sein dass unsere Daten D *zufällig* so aussehen, als ob sie von einer Markov-Kette generiert wurden (oder umgekehrt). Wir können das nie ausschliessen; der Trick ist aber: wir können das so unwahrscheinlich wir möglich machen. Zunächst müssen wir folgende Definitionen und Unterscheidungen machen.

Definition 6 *Sei Ω eine Menge von Datensätzen, $P_0, P_1 : \Omega \to [0,1]$ zwei Wahrscheinlichkeitsfunktionen. Sei $H_i : i \in \{0,1\}$ die Annahme, dass P_i die zugrundeliegende Wahrscheinlichkeitsverteilung ist, die D erzeugt hat. Ein* ***Test*** *ist eine Funktion $t : \Omega \to \{H_0, H_1\}$; $t^{-1}(H_1)$ ist der sog.* ***kritische Bereich*** *von t.*

Nehmen wir weiterhin an, wir haben guten Grund H_0 als Nullhypothese zu bezeichnen (im obigen Sinne; es ist etwas kompliziert dieses Konzept formal auszudrücken).

Definition 7 *Ein Test T macht einen* ***Typ I*** *Fehler, falls er H_1 wählt, obwohl H_0 korrekt ist; er macht einen* ***Typ II*** *Fehler, falls er H_0 wählt, obwohl H_1 korrekt ist.*

Im allgemeinen möchte man Typ I Fehler eher vermeiden als Typ II Fehler; das bedeutet, wir möchten eher konservativ sein. Das spiegelt die Tatsache wieder dass die Nullhypothese aus methodologischen Gründen vorzuziehen ist. Praktisch bedeutet dass: wir möchten eher, dass ein Medikament nicht zugelassen wird, da seine Wirkung nicht ausreichend belegt ist (aber womöglich vorhanden), als dass es zugelassen, aber womöglich unwirksam ist.

Wir wissen natürlich nie, ob wirklich ein Fehler vorliegt. Wir können allerdings über die Wahrscheinlichkeit sprechen, mit der ein bestimmter Test bestimmte Fehler macht. Sei T ein Test.

- Die Wahrscheinlichkeit das T *keinen* Typ I Fehler macht, ist seine **Signifikanz**;
- die Wahrscheinlichkeit dass er *keinen* Typ II Fehler macht, ist seine **Mächtigkeit**.

Das bedeutet: je signifikanter ein Test, desto sicherer sind wir die Nullhypothese nicht zu unrecht zu verlassen; je mächtiger er ist, desto sicherer sind wir, nicht zu unrecht bei der Nullhypothese zu bleiben. Ein Test T ist **maximal signifikant**, falls jeder andere Test T', der signifikanter ist als T, echt weniger mächtig ist; T ist **maximal mächtig**, falls jeder Test T' der mächtiger ist, echt weniger signifikant ist.

Sei $p = P(H_0)$ die a priori Wahrscheinlichkeit von H_0; wir nehmen an dass $P(H_1) = 1 - p$, es also keine weitere Hypothesen gibt. Dann haben wir, für den Fall dass wir H_1 wählen,

(158) Wahrscheinlichkeit eines Typ I Fehlers: $F1 := \dfrac{1}{1 + \frac{1-p}{p}\frac{P(D|H_1)}{P(D|H_0)}}$

und für den Fall dass wir H_0 wählen,

(159) Wahrscheinlichkeit eines Typ II Fehlers: $F2 := \dfrac{1}{1 + \frac{p}{1-p}\frac{P(D|H_0)}{P(D|H_1)}}$

Die Ableitung dieser Ergebnisse soll uns hier nicht bekümmern; wir stellen nur folgende Grenzfälle fest:

- für $\frac{1-p}{p}\frac{P(D|H_1)}{P(D|H_0)} \mapsto 1$ haben wir $F1 \mapsto 1/2$ – das ist einleuchtend, da wir in diesem Fall keine Evidenz für beide Hypothesen haben.
- für $(1-p)P(D|H_1) \mapsto 0$ haben wir $F1 \mapsto 1$ – also wie H_1 unplausibel wird, wird die Wahrscheinlichkeit eines Fehlers sicherer.
- umgekehrt gilt für $(1-p)P(D|H_1) \mapsto 0$ dass $F2 \mapsto 0$ – die Wahl von H_0 wird immer wahrscheinlicherer korrekt.

Bevor wir wirkliche Tests einführen können, brauchen wir **Statistiken**.

Definition 8 *Sei Ω ein Wahrscheinlichkeitsraum, $n \in \mathbb{N}$. Eine Statistik S ist eine Funktion auf Ω^n, dem n-fachen Produktraum.*

Statistiken sind also ein sehr allgemeiner Begriff.

Definition 9 *Sei $\mathbb{P} = \{P_\theta : \theta \in \Theta\}$ eine Menge von Wahrscheinlichkeitsfunktionen auf Ω^n. Eine Statistik S ist **ausreichend** für $\mathbb{P}$, falls für alle $\theta, \theta' \in \Theta$, $\omega \in \Omega^n$ gilt:*

$$P_\theta(\omega | S = s\}) = P_{\theta'}(\omega | S = s\})$$

wobei $s = S(\omega)$.

Eine Statistik ist ausreichend, falls sie alle Informationen enthält, die wir brauchen um Wahrscheinlichkeiten zu bestimmen. Wir definieren nun folgende Statistik, gegeben zwei Hypothesen H_0, H_1:

$$(160)\quad R(\omega) := \frac{P(\omega|H_0)}{P(\omega|H_1)} \qquad \text{(Sonderfall für } P(\omega|H_1) = 0)$$

Diese Statistik heißt das likelihood-Verhältnis. Sie ist ausreichend, d.h. enthält alle Information die wir brauchen; sie ist darüber hinaus auch minimal im Sinne dass sie keine nicht-relevante Information enthält. Im Hinblick auf Definition 9 müssten wir schreiben: $\Theta = \{0, 1\}$, und

$$(161)\quad R(\omega) := \frac{P_0(\omega)}{P_1(\omega)}$$

Das ist nur eine andere Schreibweise; das Ergebnis ist trotzdem nicht offensichtlich; wir können es hier nicht zeigen.

Ein **Schwellentest** S_t mit Wert t ist ein Test, der sich für H_0 entscheidet falls $R(\omega) > t$, und für H_1 andernfalls, also:

$$(162)\quad S_t(\omega) = \begin{cases} H_0, \text{ falls } R(\omega) > t \\ H_1 \text{ andernfalls.} \end{cases}$$

Folgender Satz ist von fundamentaler Wichtigkeit:

Satz 10 *Für jeden Wert t ist der Schwellentest S_t maximal signifikant und maximal mächtig.*

Das bedeutet, in gewissem Sinn ist jeder Wert optimal. Da wir aber die Nullhypothese *a priori* bevorzugen, wählt man üblicherweise einen Wert wie $t = 0.05$, mit hoher Signifikanz und geringerer Mächtigkeit.

Übung

Im Schwellentest müssen wir H_0, H_1 komplett ausbuchstabieren. Hierzu müssen wir Parameter schätzen (wie oben), und bilden das likelihood Verhältnis

$$R(\mathfrak{T}) = \frac{P_0(\mathfrak{T})}{P_1(\mathfrak{T}} \tag{163}$$

Als nächstes können wir einfach unseren Schwellentest anwenden:

$$S_{0.05}(\mathfrak{T}) = \begin{cases} H_0, \text{ falls } R(\mathfrak{T}) \leq 0.05 \\ H_1 \text{ andernfalls.} \end{cases} \tag{164}$$

Beispiele durchrechen:

1. $\mathfrak{T}_1 = ababababaabab$, Markov-Ketter erster Ordnung.
2. $\mathfrak{T}_2 = abbaabbaabbaabbaabbaabbabb$, Markov-Ketter erster Ordnung.
3. $\mathfrak{T}_2$, Markov-Ketter zweiter Ordnung.

16.3 p-Werte und statistische Unabhängigkeit

Der p-Wert ist etwas anders als die übrigen Testverfahren: normalerweise interessiert uns, wie Wahrscheinlich die Daten sind gegeben die Nullhypothese. Das ist aber oftmals nicht sehr informativ: gerade wenn wir eine realwertige Verteilung haben, dann interessiert uns nicht ein Punkt, sondern ein Integral. Hier hilft uns der p-Wert:

> Der p-Wert gibt die Wahrscheinlichkeit, dass die Daten so sind wie sie sind *oder noch extremer*, gegeben dass die Nullhypothese wahr ist.

Das ist also wieder das Prinzip der Vertrauensgrenzen. Dieses "*oder noch extremer*" ist aber im Allgemeinen ein Problem: was genau soll das heißen? Hier braucht man Statistiken: wir müssen unsere Ergebnisse so transformieren, dass diese Aussage Sinn macht!

Mann kann sich das sehr einfach mit dem Würfelbeispiel klarmachen.

Wir würfeln 100 mal, und haben 63 Zahl.

Nennen wir dieses Ergebnis ω. H_0 ist, dass der Würfel fair ist. Natürlich können wir sehr einfach $P(\omega|H_0)$ ausrechnen, aber das ist natürlich nicht wirklich informativ: bei vielen Würfen wird jedes Ergebnis sehr unwahrscheinlich. Andererseits, wenn wir H_0 unter ω zurückweisen, dann weisen wir es auch unter jedem ω' zurück, bei dem wir noch öfter Zahl geworfen haben. Außerdem sollten wir, da wir ein rein symmetrisches Experiment haben, H_0 ebenfalls zurückweisen unter ω', falls ω' in 63 oder mehr Würfen von Kopf besteht. Was wir also machen möchten ist: wir fassen alle diese Ergebnisse zu *einem* Ereignis zusammen, und schauen wie wahrscheinlich dieses Ereignis unter H_0 ist. Das können wir natürlich einfach ausrechnen nach den üblichen Regeln; etwas formaler *transformieren wir unseren Wahrscheinlichkeitsraum mit Hilfe von Zufallsvariablen.* Das geht so: zunächst nehmen wir

$$X(\text{Zahl}) = 1,\ X(\text{Kopf}) = 0.$$

Als nächstes nehmen wir die übliche Additionsvariable:

$$Y(\langle x_1, ..., x_i\rangle) = \sum_{j=1}^{i} x_j.$$

Damit sind unsere Ergebnisse Zahlen zwischen 1 und 100, und nicht mehr Laplace-verteilt. Dann nehmen wir eine dritte Variable:

$$Z(x) = |50 - x|.$$

Warum diese Variable? Nun, 50 ist der Mittelwert und Erwartungswert unserer Variable Y. Daher liefert uns $Z(x)$ den Wert der Abweichung von dem Erwartungswert. Was uns interessiert ist jetzt:

(165) $P(Z(x) \geq 13|H_0)$

Das ist der p-Wert von H_0 gegeben ω; normalerweise sagt man: falls $p < 0.05$, dann wird H_0 zurückgewiesen (alternativ: 0.01, 0.001). Wie ist das in unserem Fall? Wir haben

$$(166)\quad P(Z(x) \geq 13|H_0) = 2\sum_{i=63}^{100} \binom{100}{i} (\frac{1}{2})^{100}$$

Falls dieser Wert unterhalb der (vorher!) festgelegten Schwelle liegt, weisen wir H_0 zurück; wir sagen dann, das Ergebnis war **signifikant**. Aber Vorsicht: ein p-Wert $< x$ bedeutet

1. *nicht*, dass die Wahrscheinlichkeit von H_0 gegeben die Daten $< x$ ist!
2. *nicht*, dass H_0 falsch ist!
3. *nicht*, dass irgendein H_1 richtiger ist!

Insbesondere ist zu beachten: wenn mein Schwellenwert 0.05 ist, dann sage ich damit: ich möchte die Wahrscheinlichkeit eines Typ I Fehlers unter 1/20 drücken. Das bedeutet aber umgekehrt: in einem von 20 Experimenten habe ich durchschnittlich einen Typ I Fehler. Und was noch drastischer ist: angenommen, ich mache zwanzig Experimente, eines davon mit signifikantem Ergebnis – dann heißt das überhaupt nichts! Das Problem ist nun: wenn ein Wissenschaftler/Konzern eine Studie mit signifikantem Ergebnis veröffentlicht, woher weiß ich, wie viele andere Studien ohne signifikantes Ergebnis in der Schublade liegen? Die Aussagekraft des p-Wertes hängt sehr stark von Faktoren ab, die außerhalb der Studie selbst liegen. Aus genau diesem Grund gibt es momentan eine starke Bewegung von Statistikern, die gegen die Verwendung von p-Werten argumentiert.

Wir können nun zurück zu unserem Beispieltext $\mathfrak{T}$ in einer unbekannten Sprache. Nehmen wir einmal an, H_0 ist richtig: die Verteilung der Wörter

ist zufällig, d.h. die Wahrscheinlichkeit eines Wortes hängt nicht von seinen Nachbarworten ab.

In diesem Fall können wir folgendes machen: wir nehmen an, $\mathfrak{T}$ in seiner Gesamtheit stellt eine **Population** dar, aus der wir eine **Stichprobe** entnehmen. Mit Population meinen wir, dass sie die "richtigen Verhältnisse" hat, also die zugrundeliegende Wahrscheinlichkeitsverteilung widerspiegelt. Diese Wahrscheinlichkeitsverteilung nennen wir wie oben

$$\hat{P}_0$$

Dass dies die korrekte Verteilung ist kann man natürlich nie wissen, sondern nur annehmen, um daraus gewisse Schlussfolgerungen zu ziehen. Aus der Population können wir nun eine Stichprobe ziehen. Bedingung ist, dass sie zufällig ausgewählt ist; und unter dieser Bedingung sollte die Stichprobe die Wahrscheinlichkeitsverteilung der Population widerspiegeln. Wir nehmen nun als Stichprobe die Menge aller Worte, die auf das Wort w folgen. Da unter H_0 w keinen Einfluss hat auf seinen Nachfolger, ist das qua Annahme eine legitime Auswahl. Wir müssen nur sicherstellen, dass w häufig genug auftritt, sonst haben unsere nachfolgenden Untersuchungen geringe Aussagekraft. Wir benennen

$$\mathfrak{T}_w = \text{Stichprobe der Nachfolger von } w$$

Nachdem wir w gewählt haben, können wir uns die Frage stellen:

$$\text{Hat } \mathfrak{T}_w \text{ dieselbe Verteilung wie } \mathfrak{T}?$$

Vermutlich nicht! Weiterhin können wir fragen: wie wahrscheinlich ist $\mathfrak{T}_w$ unter $\hat{P}_0$? Aber das ist natürlich wieder eine unbefriedigende Fragestellung, denn wenn $\mathfrak{T}_w$ groß genug ist, wird $\hat{P}_0(\mathfrak{T}_w)$ immer sehr klein sein. Was uns also wieder interessiert ist die Frage:

> Wie groß ist, gegeben H_0, die Wahrscheinlichkeit von $\mathfrak{T}_w$ oder eines (gleichgroßen Datensatzes, der *noch unwahrscheinlicher* ist unter $\hat{P}_0$?

Und die nächste Frage ist: *wie implementieren wir dieses Konzept formal*? Die Antwort ist dieses mal etwas abstrakter als mit den Würfeln: wir generieren alle *möglichen* $\mathfrak{T}'$ (derselben Größe), so dass gilt: $\hat{P}_0(\mathfrak{T}') \leq \hat{P}_0(\mathfrak{T}_w)$.

Dass sind natürlich nur endlich viele, wir können also folgende Summe theoretisch ausrechnen:

$$(167) \quad \sum_{\mathfrak{T}':\hat{P}_0(\mathfrak{T}')\leq\hat{P}_0(\mathfrak{T}_w)} \hat{P}_0(\mathfrak{T}')$$

Das ist ein Ereignis und liefert eine Wahrscheinlichkeit, und das ist der p-Wert für H_0 gegeben $\mathfrak{T}_w$. Falls dieser Wert kleiner als unser Schwellenwert ist (z.B. 0.05), dann weisen wir die Nullhypothese zurück. Diese Hypothese war: die Wahrscheinlichkeiten von aufeinanderfolgenden Wörtern sind unabhängig.

Wir haben also eine hübsche, allgemeine Form, die der Computer relativ gut berechnen kann (bzw. approximieren). Wir als Menschen haben allerdings kaum eine Chance diese Formel in eine allgemeine Form zu bringen, die wir effektiv berechnen können. Die Berechenbarkeit von 417 beruht wiederum auf der Annahme von H_0: wir können einfach lokal jeweils Buchstaben durch weniger wahrscheinlichere Buchstaben ersetzen, und wir bekommen einen unwahrscheinlicheren Datensatz. Wir können also einfach Buchstaben nacheinander lokal ersetzen. Zusammen mit den Distributivgesetzen in der arithmetischen Entsprechung lässt sich das ganz gut ausrechnen.

16.4 Die χ^2-Verteilung und der χ^2-Test

Die χ^2 Funktion ist eigentlich eine Familie von Funktionen mit Parameter n; sie ist wie folgt definiert. Sei $n(x|\mu,\sigma)$ die Normalverteilung mit Mittelwert μ und Standardabweichung σ. Man definiert normalerweise:

$$(168) \quad \chi^2_n \sim n(x_1|0,1)^2 + ... + n(x_n|0,1)^2$$

wobei wir hier eine Korrespondenz $x \cong (x_1, ..., x_n)$ haben. Das ist eine etwas kryptische Definition; wenn man das auflöst bekommen man:

$$(169) \quad \chi^2_n(x) = P(x_1^2 + ... + x_n^2 = x),$$

wobei gilt: für alle x_i,

$$(170) \quad P(X_i = x_i) = n(x_i|0,1)^2$$

Das Quadrat bewirkt, dass alle Variablen positiv sind; da $n(x|0,1)$ den Erwartungswert und Maximum bei 0 haben, bekommen wir für $n = 1$ eine

Funktion die von 0 relativ steil abfällt (Exponentialverteilung!). Wie n wächst, verschiebt sich das Maximum aber nach rechts, aus den bekannten kobinatorischen Gründen.

Die χ^2-Verteilung einer Variable X besagt soviel: es gibt eine Menge von n Variablen $Y_1, ..., Y_n$, so dass

1. $Y_1, ..., Y_n$ normal verteilt sind mit Mittelwert 0, Standardabweichung 1,
2. es gibt eine Korrespondenz $x \cong (y_1, ..., y_n)$, so dass $P(X = x) = P(Y_1 = y_1) + ... + P(Y_n = y_n)$ für alle $x \in X$.

Das bedeutet: die Verteilung von X ergibt sich aus einer Summe von "normalen" Normalverteilungen.

Die χ^2-Verteilung ist eindeutig bestimmt durch n, die Anzahl der summierten Variablen; man nennt diese Zahl auch den **Freiheitsgrad** der Verteilung.

Diese Verteilung ist insbesondere interessant in Hinblick auf den sog. χ^2-**Test**. Dieser Test wird dafür benutzt, zu prüfen ob zwei Variablen voneinander unabhängig sind. Nehmen wir an, unsere Beobachtungen D sind Teilmengen von $M_1 \times M_2$, also

$$D \subseteq M_1 \times M_2$$

D.h. alle Beobachtungen haben 2 Merkmale (natürlich viele andere, die aber nicht interessieren). Nehmen wir an,

- wir haben n Beobachtungen gemacht;
- Merkmal M_1 hat n_1 Realisierungen, man schreibt $|M_1| = n_1$, da hier Merkmale nur die Menge ihrer Realisierungen sind),
- $|M_2| = n_2$

Uns interessiert nun: sind diese Merkmale voneinander unabhängig? Was wichtig ist ist folgendes: wir brauchen

$$n \gg n_1 \times n_2,$$

d.h. wir benötigen deutlich mehr Beobachtungen als mögliche Ergebnisse der Beobachtungen, sonst bleiben unsere Ergebnisse unbedeutend.

Wir können nun unsere Beobachtungen als Zahlen in eine

$$n_1 \times n_2\text{-Matrix}$$

eintragen; wir nennen den Eintrag mit Zeile i und Spalte j n_{ij}, wie das üblich ist bei Matrizen. Außerdem haben wir

$$\begin{array}{l}\text{die Zeilensummen } n_{1*} = \sum_{i=1}^{n_2} n_{1i} \text{ etc. und} \\ \text{die Spaltensummen } n_{*1} = \sum_{i=1}^{n_1} n_{i1},\end{array}$$

die uns die Häufigkeit der einzelnen Merkmale liefern. Alle diese Zahlen lassen sich natürlich transformieren in relative Häufigkeiten, indem wir mit n dividieren. Wir wissen, dass Wahrscheinlichkeitstheoretisch gilt: M_1 und M_2 sind **unabhängig**, genau dann wenn gilt:

$$P(M_1 = m_1, M_2 = m_2) = P(M_1 = m_1) \cdot P(M_2 = m_2) \tag{171}$$

(das ist nur eine Instanz unserer Definition der Unabhängigkeit $P(A \cap B) = P(A) \cdot P(B)$). In Bezug auf unsere Beobachtungen bedeutet das:

$$n_{ij} \approx \frac{n_{i*} \cdot n_{*j}}{n} \tag{172}$$

Denn wir müssten haben (für relative Häufigkeiten)

$$\frac{n_{ij}}{n} \approx \frac{n_{i*}}{n} \cdot \frac{n_{*j}}{n} \tag{173}$$

Bei tatsächlicher Unabhängigkeit der Verteilung müsste für alle i, j, die Differenz der beiden Terme eher klein sein. Das Problem ist – wie bei allen Tests – das es eben keine Definition gibt von "eher klein"; wir müssen das eben willkürlich festmachen. Hier hilft uns aber die χ^2-Verteilung. Wir setzen den Erwartungswert für n_{ij} auf $\frac{n_{i*} \cdot n_{*j}}{n}$; also

$$\mathcal{E}(n_{ij}) = \frac{n_{i*} \cdot n_{*j}}{n} \tag{174}$$

und wir nehmen jetzt das Quadrat der Abweichung – also die Varianz. Das liefert uns eine neue Verteilung:

$$\mathtt{V}(n_{ij}) = (n_{ij} - \mathcal{E}(n_{ij}))^2 \tag{175}$$

Diese Varianz – wir arbeiten ja nach wie vor mit absoluten Häufigkeiten – muss wiederum normalisiert werden mit dem Erwartungswert (denn sie wächst natürlich mit den absoluten Zahlen – je mehr Beobachtungen, desto

mehr und größere Abweichungen erwarten wir. Wir bekommen also den normalisierten Term

$$\frac{\mathtt{V}(n_{ij})}{\mathcal{E}(n_{ij})} = \frac{(n_{ij} - \mathcal{E}(n_{ij}))^2}{\mathcal{E}(n_{ij})} \tag{176}$$

Das interessante hieran ist: aus der Annahme, dass die Variablen unabhängig sind, folgt dass die **Abweichung** $n_{ij} - \mathcal{E}(n_{ij})$ **normalverteilt** ist um die 0; wir summieren nun diesen Term über alle i und j; das liefert uns dann

$$P(D) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \frac{(n_{ij} - \mathcal{E}(n_{ij})^2}{\mathcal{E}(n_{ij})} \tag{177}$$

Nun kommt der entscheidende Punkt:

- Wenn unsere Annahme der Unabhängigkeit stimmt, dann ist jeder Summand eine quadrierte Normalverteilung (grob gesagt);
- also ist die Summe eine χ^2-Verteilung mit $(n_1 - 1) \cdot (n_2 - 1)$ Freiheitsgraden (das ist aber nicht offensichtlich oder trivial!);
- nach Annahme müsste unser Prüfwert $P(D)$ also von einer solchen Verteilung generiert worden sein!

Nun können wir einfach einen klassischen, einseitigen Vertrauenstest anwenden: wir legen eine Vertrauensgrenze fest (z.B. $c = 0.95$). Dann schauen wir, ob die Prüfgröße $P(D)$ im (linken) 0.95-Quantil der $\chi^2_{(n_1-1)\cdot(n_2-1)}$-Verteilung mit $(n_1-1)\cdot(n_2-1)$ Freiheitsgraden liegt. Falls ja, müssen wir bei H_0 bleiben: die beiden Variablen sind unabhängig; falls nein, weisen wir H_0 zurück.

16.5 χ^2 in R

Wir können natürlich den χ^2-test in R implementieren: Mittelwerte bzw. Erwartungswerte verschiedener Variablen sind schnell berechnet, ebenso die einfachen arithmetischen Funktionen die uns die Prüfgröße $P(D)$ liefern. Was man natürlich ebenso implementieren kann ist die Verteilung χ^2_n; das ist aber relativ kompliziert (wie auch die Definition kompliziert ist!) und trägt der konzeptuellen Einfachheit keine Rechnung. Die Verteilung gibt es fertig in *R* mit dem Namen **dchisq()**, wobei wir die üblichen Varianten **pchisq(),qchisq()** haben; die Funktion hat zwei Argumente: das eigentliche Argument und die *Freiheitsgrade*.

> dchisq(0.5,df=1)
> chi21 ¡- function(x)dchisq(x,df=3)
> plot(chi21,0,10)

Ebenso ist natürlich auch der χ^2-Test eingebaut R. Dieser Test nimmt als Eingabe eine sog. **contingency table**. Diese haben wir bereits betrachtet und kennen die Befehle:

> table1 ¡- xtabs(~ RealizationOfRec + AnimacyOfRec, data = verbs)

Nun können wir hierauf einen Test applizieren:

> chisq.test(table1)

Eine nette Übung ist es, diesen Test nun selbst zu durchzuführen!

16.6 t-test – ein Beispiel

Der Studentsche t-Test beruht darauf, dass wir Mittelwerte miteinander vergleichen, und zwar für

- Eine normalverteilte Gesamtheit und eine nach Annahme zufällig ausgewählte, normalverteilte Stichprobe daraus
- oder zwei nach Annahme zufällig ausgewählten normalverteilten Stichproben aus einer Gesamtheit.

In diesem Fall ist unser Datensatz ziemlich schwierig auf diese Art und Weise zu behandeln; es wäre besser einen Datensatz von reellen Zahlen zu haben, also:

$$\mathfrak{T}_2 = x_1 x_2 x_3 ... x_n, \text{ wobei } x_i \in \mathbb{R}$$

Wir können nun den Mittelwert dieses Satzes nehmen:

$$\mu(\mathfrak{T}_2) = \sum_{i=1}^{n} x_n \cdot \frac{1}{n} \tag{178}$$

also die Summe aller Werte geteilt durch die Anzahl der Werte. Nun nehmen wir eine zufällige Stichprobe; da nach Annahme von H_0 die Werte voneinander unabängig sind, ist

$$\mathfrak{S} = y_1 y_2 ... y_j : y_i = x_{i'}, x_{i'-1} \in [a, b]$$

Wir nehmen uns also die Folge aller Werte aus $\mathfrak{T}_2$, für die gilt: der Vorgängerwert in $\mathfrak{T}_2$ liegt im Intervall $[a, b]$, was ein beliebig gewähltes Intervall ist. Wir nehmen hier eher ein Intervall als einen Wert, sonst bekommen wir zuwenig Datenpunkte. Nun gilt: $\mathfrak{S}$ ist zufällig gewählt unter Annahme von H_0. Das bedeutet aber auch, wir sollten haben:

$$\mu(\mathfrak{S}) \approx \mu(\mathfrak{T}_2) \tag{179}$$

Wenn die beiden nun sehr unterschiedlich sind, dann spricht das gegen die Annahme, dass H_0 korrekt ist – wir haben also Evidenz, H_0 zurückzuweisen. Andersrum, wenn 180 erfüllt ist, bedeutet dass, das zumindest für $[a, b]$ nichts gegen H_0 spricht.

Allerdings gibt es eine wichtige Sache zu beachten: das setzt voraus dass die Daten in $\mathfrak{T}_2$ einigermaßen *normalverteilt* sind. Um das zu sehen, bedenke

man folgendes: nimm an, $\mathfrak{T}_2$ liegt eine Zipf-Verteilung zugrunde, d.h. es gibt sehr wenige Punkte, die sehr viel Masse auf sich vereinen. Nun kann es gut sein, dass rein zufällig $\mathfrak{S}$ diese (wenigen) Punkte nicht enthält, und daraus folgt natürlich:

(180) $\mu(\mathfrak{S}) \ll \mu(\mathfrak{T}_2)$

($\ll$ bedeutet: deutlich kleiner). Dasselbe kann auch bei beliebigen Stichproben gelten. Das bedeutet: einem t-Test sollte eine Normalverteilung zugrunde liegen.

Kehren wir zurück zu unserem Datensatz D. Das Problem in unserem Beispiel war folgendes: H_1, sobald sie ausbuchstabiert ist, ist viel zu spezifisch. Wir möchten eher eine allgemeinere Hypothese H_1, nämlich dass die Wahrscheinlichkeiten von einzelnen Worten abhängig voneinander sind. Hier können wir den sog. **t-test** benutzen, der

- für einen gegebenen Datensatz, und
- zwei Stichproben daraus

bestimmt, ob zwei Faktoren *unabhängig* voneinander sind. Bevor wir damit anfangen können, müssen wir zunächst unsere Daten etwas präparieren.

Zur Erinnerung: unsere beiden Hypothesen waren:

> H_0, alle Worte als Ereignisse sind unabhängig voneinander;
>
> H_1, die Wahrscheinlichkeit eines Wortauftretens ist abhängig vom vorhergehenden Wort.

Wir brauchen also, für jedes Wort v in unserem Text, eine ganze Reihe Parameter:

1. seine absolute Häufigkeit, geteilt durch die Anzahl der Worte (*token*) im Text (also relative Häufigkeit)
2. die Häufigkeit des Vorkommens von v nach einem gegebenen Wort w, geteilt durch die Häufigkeit von w, für jedes Wort w das im Text auftritt.

Wir kriegen also zu jedem Wort (eine Zeile in einer Tabelle) eine Reihe von Zahlen (Spalten in einer Tabelle).

- Wir nennen die Spalte mit den allgemeinen relativen Häufigkeiten S_1,
- die Spalte mit den relativen Häufigkeiten für Vorgänger w nennen wir S_w.

Es handelt sich also um Vektoren (Listen) von Zahlen.

Was wir als nächstes brauchen ist das Konzept des Mittelwertes: gegeben eine endliche Menge (oder Liste) von Zahlen

$$\mathbf{X} = \{x_1, ...x_n\} \subseteq \mathbb{R}$$

bezeichnen wir den **Mittelwert** von $\mathbf{X}$ mit

$$\mu(\mathbf{X}) = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Wir können nun beispielsweise, indem wir etwas liberal im Umgang mit Listen und Mengen sind, einfach

$$\mu(S_1),\ \mu(S_w) \text{ etc.}$$

berechnen. Wie machen wir das? Wir summieren alle Zahlen der Spalte auf, und dividieren durch die Anzahl der Zeilen darin. NB: die Anzahl der Zeilen ist die Anzahl der *types* in unserem Text. Wir dividieren also zunächst absolute Häufigkeiten durch Anzahl der token, addieren die Ergebnisse und dividieren durch Anzahl der types.

Was bedeutet dieser Wert? Er sagt uns, wie oft ein beliebiges Wort geteilt durch die Anzahl der Worte (*token*) im Text durchschnittlich auftritt. Wegen der arithmetischen Distributivgesetze können wir diese Transformation umkehren: nehmen wir an,

- unser Text $\mathfrak{T}$ enthält k *token* , und sei
- $\mu(S_1) = x$.
- Dann kommt ein beliebiges Wort durchschnittlich kx im Text vor.

Nehmen wir an, jedes Wort kommt nur einmal vor. k ist die Anzahl der *token*; sei $|L|$ die Anzahl der *types* im Text. Wir bekommen dann also als Wert

$$\mu(S_1) = \frac{1}{k},$$

da in diesem Fall $l = k$ ist. Im allgemeinen Fall lässt sich leicht zeigen, dass das Ergebnis immer

$$\mu(S_1) = x = \frac{1}{l} \tag{181}$$

lautet, also die Anzahl der token. Die Frage ist nur: Wie sind die Häufigkeiten verteilt? Auch hierfür haben wir eine gute Antwort: die Verteilung wird normalerweise eine Zipf-Verteilung sein.

Dasselbe können wir nun auch für die anderen Spalten machen; bsp. für das Wort w_1. Während für S_1 das Ergebnis in gewissem Sinne trivial war, ist es nun alles andere als trivial. Was wir dann bekommen ist

$$\mu(S_{w_1}),$$

d.i. die durchschnittliche Häufigkeit eines beliebigen Wortes nach w_1, geteilt durch die Anzahl der Vorkommen von w_1. Hier sehen wir, warum wir die Division machen:

- dadurch werden etwa $\mu(S_1)$ und $\mu(S_{w_1})$ *vergleichbar*.

Für $\mu(S_{w_1})$ gibt es allerdings etwas sehr wichtiges zu beachten: wir dürfen nicht durch die Gesamtlänge der Spalte dividieren, sondern nur durch die Anzahl von Kästchen, die einen positiven Eintrag haben. Warum? Weil wenn wir Dinge anfangen, Dinge zu berücksichtigen, die gar nicht vorkommen, dann müssten wir ja Äpfel und Birnen etc. berücksichtigen.

Was erwarten wir uns also, wenn wir diesen Wert berechnen? Nun, nehmen wir beispielsweise an, in $\mathfrak{T}$ folgt auf das Wort w_1 *immer* dass Wort w_2. Was wäre dann $\mu(S_{w_1})$? Wir bekommen dann tatsächlich den Wert

$$\mu(S_{w_1}) = 1 \tag{182}$$

denn wir haben als Summe aller Zahlen in 1, und da wir in S_{w_1} nur in einem Kästchen einen positiven Eintrag finden, dividieren wir durch 1.

Nehmen wir an wir machen diese Beobachtung. Das ist natürlich Evidenz gegen H_0. Aber ist diese Evidenz stark? Das hängt natürlich davon ab, wie häufig w_1 ist! Denn wenn es nur einmal vorkommt, dann war unsere Beobachtung trivial. Auf der anderen Seite, wenn w_1 sehr häufig ist, dann sollten wir erwarten, dass $\mu(S_{w_1})$ dem Wert $\mu(S_1)$ eher ähnlich ist. Wenn wir uns aber ein bestimmtes, häufiges Wort aussuchen, dann laufen wir natürlich Gefahr, durch diese Auswahl wiederum die Gültigkeit unserer Beobachtung

einzuschränken. Außerdem lassen wir den größten Teil unserer Information ungenutzt. Was können wir also tun?

Wir können uns helfen, indem wir folgendes machen: wir betrachten nicht nur $\mu(S_{w_1})$, sondern wir **generalisieren** die ganze Prozedur, so dass wir den Mittelwert über *alle* Mittelwerte bilden; wir berechnen also

(183) $\mu(\{\mu(S_w) : w \in \mathfrak{T}\})$

Wir betrachten also wiederum alle diese Werte, und mitteln über sie. Was sagt uns das? Das hängt wiederum davon ab, wie die Häufigkeiten verteilt sind: wenn jedes Wort nur einmal vorkommt im Text, dann wird auch dieser Wert sehr uninformativ sein. Wenn aber alle Worte sehr haüfig sind, dann würden wir folgendes erwarten:

(184) $\mu(\{\mu(S_w) : w \in \mathfrak{T}\} \approx \mu(S_1) = \frac{1}{l}$

– unter Annahme der Nullhypothese!

Das Problem ist nun folgendes: wir haben gesagt dass wir wahrscheinlich eine Zipf-Verteilung haben, das bedeutet: die überwiegende Mehrheit der Worte taucht nur einmal auf. Es wird also auf jeden Fall einen verzerrten Wert geben, denn die Mehrzahl der Worte wird eben einen sehr hohen Wert haben. Wie kommen wir um dieses Problem herum? Der Trick ist:

> wir nehmen eine **Stichprobe** $\mathfrak{S}$ aus unseren Daten, die **normalverteilt** ist; d.h. mit wenig sehr seltenen und wenig sehr häufigen und vielen mittelhäufigen Wörtern.

Dann berechnen wir wiederum daraus den Mittelwert, und nun können wir die beiden Mittelwerte vergleichen.

Nun nehmen wir an, H_0 ist wahr. In diesem Fall sollten die beiden Mittelwerte in etwa gleich sein, also

(185) $\mu(\{\mu(S_w) : w \in \mathfrak{S}\} \approx \mu(S_1)$

Das bedeutet, die Sicherheit, mit der wir wissen dass ein Nachfolger nach einem gewissen Vorgänger kommt, ist nicht wesentlich größer als die Sicherheit, das er überhaupt auftritt (sie wird natürlich immer größer sein, da wir immer gewisse Artefakte haben.

Das bedeutet also: je *weniger* der Wert

$$\mu(\{\mu(S_w) : w \in \mathfrak{T}\})$$

höher sein wird als

$$\mu(S_1),$$

desto *weniger* Evidenz haben wir gegen die Nullhypothese; umgekehrt, je mehr der Wert nach oben abweicht, desto eher können wir die Nullhypothese ablehnen.

Wir wissen natürlich immer noch nicht, ab wann wir H_0 ablehnen können; allerdings haben wir unsere Daten nun so zurecht gelegt, dass sie nur in *eine* Richtung abweichen, fallst H_0 falsch ist; alles was wir brauchen ist ein Schwellenwert, ab dem wir H_0 ablehnen. Um diesen Wert zu finden, brauchen wir natürlich zusätzliche Erwägungen.

Irgendwie bleibt das aber alles unbefriedigend, denn wir haben kein Maß dafür, inwieweit ein Wort für ein nachfolgendes Wort **informativ** ist. Das werden wir als nächstes betrachten.

17 Markov-Ketten

17.1 Vorgeplänkel

Markov-Ketten sind stochastische Prozesse, bei denen die Wahrscheinlichkeiten eines Ergebnisses von einer begrenzten Reihe von vorherigen Ergebnissen abhängen. Man spricht auch von Markov-Prozessen, wobei dieser Begriff eher für kontinuierliche Prozesse verwendet wird, der Begriff Markov-Kette eher für diskrete Prozesse. Wir werden uns hier ausschließlich mit **diskreten Prozessen** beschäftigen. Markov-Prozesse sind diskret, wenn sie über eine diskrete Kette von Ereignissen definiert sind. Eine Kette ist eine lineare Ordnung $(M, <)$ mit $<\subseteq M \times M$, wie etwas die natürlichen Zahlen, eine beliebige Teilmenge davon, die rationalen, reellen Zahlen etc., geordnet nach "ist größer als".

Definition 11 *Eine Kette $(M, <)$ ist **diskret**, falls es für jedes $m \in M$ ein m' gibt, so dass für alle $n \in M$ gilt: falls $n < m$. dann $n \leq m'$. m' ist dann der unmittelbare Vorgänger von m.*

$(\mathbb{N}, <)$ ist diskret, wobei der unmittelbare Vorgänger von m, für $m \geq 2$, $m-1$ ist. Jede endliche Kette ist diskret. Die Ketten $(\mathbb{Q}, <)$ und $(\mathbb{R}, <)$ sind nicht diskret: denn was ist die größte rationale (reelle) Zahl, die echt kleiner als 2 ist?

Nehmen wir wieder einmal die Münze. Wie ist die Wahrscheinlichkeit dafür, mit 10 Würfen mindestens dreimal Zahl zu werfen? Diese Wahrscheinlichkeit kennen wir (unter der Annahme dass die Münze fair ist). Wir werden jetzt aber noch zusätzliche Informationen berücksichtigen: wie ist diese Wahrscheinlichkeit, gegeben dass wir mit den ersten 9 Würfen nur *einmal* Zahl geworfen haben? Offensichtlich 0, ganz unabhängig von der Münze! Umgekehrt, wie ist die Wahrscheinlichkeit, gegeben dass wir mit den ersten 9 Würfen bereits 6mal Zahl geworfen haben? Offensichtlich 1, ebenfalls unabhängig von den zugrundeliegenden Wahrscheinlichkeiten. Ein dritter Fall ist die Wahrscheinlichkeit unter der Annahme dass wir mit 9 Würfen 2mal Zahl geworfen haben - hier hängt die Wahrscheinlichkeit von der Münze selber ab, und ist 0.5 im Fall einer fairen Münze.

Wir verallgemeinern das Beispiel. Sei $\mathcal{P}$ ein Bernoulli-Raum mit $p = P(1)$, X_n eine (Reihe von) Zufallsvariable(n) mit

$$X_n(\langle \omega_1, ..., \omega_n \rangle) = \sum_{i=1}^{n} \omega_i,$$

für beliebige $n \in \mathbb{N}$. Wir bezeichnen mit S_n ein Ereignis von n Würfen (mit irgendwelchen Ergebnissen), S_{n-1} das Ereignis von n_1 Würfen etc. Uns interessiert die Wahrscheinlichkeit von $P(X_n = r)$, also r-mal Zahl von n Würfen. Diese Wahrscheinlichkeit von $X_n(S_n) = r$ hängt nun offensichtlich ab von $X_{n-1}(S_{n-1})$, wie wir oben gesehen haben; falls $X_{n-1}(S_{n-1}) = r - 1$, dann ist $P(X_n = r) = p$, falls $X_{n-1}(S_{n-1}) = r$, dann ist $P(X_n = r) = 1 - p$, und in allen anderen Fällen ist $P(X_n = r) = 0$.

NB: was hier wichtig ist $X_{n-1}(S_{n-1})$; alle vorigen Ergebnisse, also $X_{n-j}(S_{n-j})$ für $1 < j < n$ sind vollkommen unerheblich. Hier handelt es sich um ein typisches Beispiel von einer Markov Kette *erster Ordnung* - die Verteilung für S_n hängt ausschließlich an S_{n-1} und p; die Zukunft und die fernere Vergangenheit spielen überhaupt keine Rolle.

17.2 Markov-Ketten: Definition

Wir haben nun das wichtigste Merkmal von Markov-Ketten beschrieben: Ereignisse beeinflussen die Wahrscheinlichkeiten von Nachfolgeereignissen, aber nur einem begrenzten Abstand. Wir werden nun eine formale Definition liefern.

Definition 12 *Sei $S_n : n \in \mathbb{N}$ eine (endliche oder unendliche) Reihe von Ergebnissen, $X_n : n \in \mathbb{N}$ Reihe von Zufallsvariablen auf S_n. $X_n : n \in \mathbb{N}$ ist eine Markov-Kette $m-ter$ Ordnung, falls*

$$\begin{aligned} & P(X_{t+1} = x_{t+1} | X_t = x_t, X_{t-1} = x_{t-1}, ..., X_1 = x_1) \\ = \quad & P(X_{t+1} = x_{t+1} | X_t = x_t, X_{t-1} = x_{t-1}, ..., X_{t-m} = x_{(t-m)+1}) \end{aligned}$$

Das bedeutet soviel wie: nur die letzten m Ergebnisse beeinflussen die Wahrscheinlichkeit von $P(X_t = x_t)$, alle anderen sind irrelevant. Beachten Sie, dass Ketten von Ereignissen der Form: der n-te Wurf ist Zahl, der $n+1$-te Wurf ist Kopf etc., wo alle Ereignisse unabhängig sind, Markov Ketten 0-ter Ordnung sind.

Unser obiges Beispiel war in gewissem Sinne irreführend für die Anwendung von Markov-Prozessen, denn in diesem Beispiel sind alle Ereignisse voneinander unabhängig. Markov-Ketten werden hingegen gerade dann benutzt, wenn diese Prämisse *nicht* gegeben ist, d.h. wenn wir nicht annehmen können dass die vorherigen Ergebnisse die nachfolgenden nicht beeinflussen. Ein besseres Beispiel wäre ein Spiel wie "Schiffe versenken": hier können Sie die von der Wahrscheinlichkeit sprechen, dass ein Spieler ein gewisses Feld wählt; aber natürlich nicht unabhängig von seinen bieherigen Entscheidungen: denn er wird gewisse strategisch interessante Felder wählen.

Auf der anderen Seite, wenn Sie einen Zustand betrachten als die Spielsituation nach einem bestimmten Zug, dann ist es allein der letzte Zustand, der die Wahrscheinlichkeit beeinflusst (d.h. die Spielsitation nach dem letzten Zug). Die Reihe der vorherigen Informationen hingegen wird völlig irrelevant, denn alle relevante Information steckt ja bereits im letzten Zustand (hier lassen wir natürlich Faktoren wie eine bestimmte Strategie des Spielers oder Gewohnheit außen vor).

Wir haben von Zuständen geredet, wie es bei Markov-Ketten üblich ist. Als Zustand betrachten wir Objekte der Form $X_n(S_n)$, also Bilder der Zufallsvariablen. Beachten Sie dass wir hier Zufallsvariablen in einem weiteren Sinne benutzen als gewöhnlich; insbesondere sind, in unserem letzten

Beispiel, die Zustände *nicht* die Züge, sondern die Spielsituationen, die daraus resultieren! Andernfalls haben wir sicher keine Markov-Kette erster Ordnung, und im allgemeineren Fall nicht einmal eine Markov-Kette! Hieraus wird hoffentlich deutlich, warum wir von Zuständen sprechen.

17.3 (Teile der) Sprache als Markov-Prozess

Wir haben bereits in einigen Beispielen Texte behandelt, in denen gewisse Buchstaben eine gewisse Wahrscheinlichkeit des Auftretens haben. Wenn wir natürliche Sprachen wie Deutsch betrachten, dann macht es wenig Sinn mit solchen Wahrscheinlichkeiten zu rechnen: denn die entscheidende Voraussetzung für die Methode, mit der wir Wahrscheinlichkeiten von Worten berechnet haben, ist das Buchstaben in einem (deutschen) Text zufällig verteilt sind. Diese Annahme ist natürlich abwegig, wie wir bereits auf der Ebene der sog. Phonotaktik feststellen: `kle` ist eine mögliche Buchstabenfolge, während eine Folge wie `klp` nicht möglich ist als deutsche Buchstabenfolge. Diese einfache Regelmässigkeit ist eine von vielen, und wir können sie ganz einfach wie folgt erfassen:

(186) $P(\texttt{p}|\texttt{kl}) = 0$

Hier ist `x` eine Kurzschreibweise für das Ereigniss: "der n-te Buchstabe im Text ist x; wir können das notieren als $n = \texttt{x}$; und

(187) $P(\texttt{p}|\texttt{kl})$

ist eine Kurzform für:

(188) $P(n = \texttt{p}|n-1 = \texttt{l}, n-2 = \texttt{k})$

Wir können also phonotaktische Regeln als Markov-Kette kodieren.

Die große Frage ist jedoch: ist die Verteilung von Buchstaben in einem Text tatsächlich ein Markov Prozess? Diese Frage lässt sich natürlich, wie alle empirischen Fragen über Wahrscheinlichkeiten, nur näherungsweise betrachten. Wenn wir zunächst phonotaktische Beschränkungen betrachten, dann stellen wir fest dass es solche Beschränkungen nur im Rahmen einer Silbe gibt. (Das gilt wohlgemerkt nicht für alle Sprachen; es gibt phonotaktische Phänomene wie Vokalharmonie die über die Silbengrenze hinweg gelten.) Die mögliche Größe von Silben ist beschränkt: die (meines Wissens) Längste deutsche Silbe ist das Wort `springst` mit 8 Phonemen bzw. `schlingst` mit neun Buchstaben. Und eigentlich können wir diese Zahl noch weiter verringern, denn die Buchstaben in Silbenauftakt (*onset*) und Coda haben keinen Einfluss aufeinander; aber darauf soll es uns nicht ankommen.

Wir können also aus diesem Grund behaupten, dass die Verteilung von Buchstaben in deutschen Texten mit einer Markov-Kette modelliert werden kann; und zumindest wird uns jeder Recht geben, dass dieses Modell

besser ist als das krude Zufallsmodell. Wenn wir allerdings annehmen, dass auch Faktoren Syntax und Semantik eine Rolle spielen für die Verteilung von Buchstaben, dann ist unser Modell natürlich völlig inadequat.

17.4 Likelihood und Parameter-Schätzung bei für Markov-Ketten

Wir haben gesehen, wie wir effektiv Parameter aus Daten schätzen können mit der Maximum-Likelihood Schätzung. Wenn wir also annehmen, dass Buchstaben in Texten zufällig verteilt sind, dann können wir, gegeben einen Text der groß genug ist um einigermaßen zuverlässig zu sein, effektiv schätzen was die zugrundeliegenden Parameter sind. Können wir diese Methode effektiv erweitern für Markov-Ketten?

- Sei $\mathfrak{T}$ ein Text.
- Wir bezeichnen mit $\mathtt{a}(\mathfrak{T})$ die Anzahl von $\mathtt{a}$s in $\mathfrak{T}$ etc.,
- mit $|\mathfrak{T}|$ bezeichnen wir die Anzahl der Zeichen in $\mathfrak{T}$.

Wir haben gesehen dass wir die Maximum Likelihood für $P(\mathtt{a})$ effktiv schätzen können mit

$$\frac{\mathtt{a}(\mathfrak{T})}{|\mathfrak{T}|} \tag{189}$$

Wir erweitern unsere Schätzung nun für Markov-Ketten. Einfachheit halber nehmen wir zunächst eine Markov-Kette erster Ordnung als Modell, obwohl das natürlich inadäquat ist, wie wir gesehen haben. Zunächst machen wir folgende Annahme: wir erweitern unser Alphabet Σ, das bereits das Leerzeichen enthält, um die Zeichen $\ltimes, \rtimes \notin \Sigma$. Wir nehmen an, dass

- $\ltimes$ (nur) am Anfang jedes Textes steht,
- $\rtimes$ nur am Ende.

Uns interessiert natürlich nicht die Wahrscheinlichkeit von # selbst, sondern die Wahrscheinlichkeit, dass ein Buchstabe am Anfang eines Textes steht. Diesen Fall müssen wir natürlich gesondert betrachten, denn in diesem Fall haben wir keine Vorgängerzustände, auf die wir uns berufen können. Wir müssen dabei beachten, dass wir für verlässliche Schätzungen für $P(\mathtt{a}|\#_e)$ eine Vielzahl von Texten betrachten müssen, da wir pro Text nur eine solche Folge haben.

Wir möchten zunächst die Wahrscheinlichkeit von $P(\mathtt{a}|\mathtt{x})$ für alle $\mathtt{x} \in \Sigma$ berechnen. Wir tun das auf eine denkbar einfache Art und Weise: wir erweitern unsere Notation $\mathtt{a}(\mathfrak{T})$ auf Worte, so dass $\mathtt{abc...}(\mathfrak{T})$ die Anzahl aller Vorkommen von $\mathtt{abc...}$ in $\mathfrak{T}$ ist. Wir sagen nun:

$$P(\mathtt{a}|\mathtt{b}) := P(X_{i+1} = a|X_i = b) \tag{190}$$

Nach ML-Schätzung bekommen wir nun:

$$\hat{P}(\mathtt{a}|\mathtt{b}) := \frac{\mathtt{ba}(\mathfrak{T})}{\mathtt{b}(\mathfrak{T})}, \tag{191}$$

wobei $\hat{P}$ die von uns geschätzte Wahrscheinlichkeit bezeichnet. Diese Schätzung erlaubt es uns, für alle $\mathtt{a},\mathtt{b} \in \Sigma$ die Wahrscheinlichkeit $\hat{P}(\mathtt{a}|\mathtt{b})$ zu schätzen.

Diese Methode lässt sich leicht auf beliebige Markov-Ketten n-ter Ordnung verallgemeinern: sei $\vec{\mathtt{w}}$ ein Wort mit $|\vec{\mathtt{w}}| = n$; dann ist

$$\hat{P}(\mathtt{a}|\vec{\mathtt{w}}) := \frac{\vec{\mathtt{w}}\mathtt{a}(\mathfrak{T})}{\vec{\mathtt{w}}(\mathfrak{T})}. \tag{192}$$

Mit diesen bedingten Wahrscheinlichkeiten können wir nicht ohne weiteres zu den unbedingten Wahrscheinlichkeiten zurückkommen: wir haben zwar die bekannten Regeln zur bedingten Wahrscheinlichkeit und Partitionen, und bekommen:

$$\hat{P}(\mathtt{a}) := \sum_{|\vec{\mathtt{w}}|=n} \hat{P}(\mathtt{a}|\vec{\mathtt{w}})\hat{P}(\vec{\mathtt{w}}), \tag{193}$$

Allgemeiner ausgedrückt, für eine Markov-Kette n-ter Ordnung, $|\vec{\mathtt{w}}| \leq n$, haben wir

$$\hat{P}(\mathtt{a}|\vec{\mathtt{w}}) := \sum_{|\mathtt{x}\vec{\mathtt{w}}|=n} \hat{P}(\mathtt{a}|\mathtt{x}\vec{\mathtt{w}})\hat{P}(\vec{\mathtt{x}}), \tag{194}$$

Aber um Wahrscheinlichkeiten zu berechnen, brauchen wir Wahrscheinlichkeit von Wörtern $\hat{P}(\vec{\mathtt{w}})$! Wahrscheinlichkeit von Wörtern berechnet sich wie folgt: sei $\vec{\mathtt{w}} = \mathtt{a_1 a_2 ... a_i}$. Dann ist

$$\begin{aligned} \hat{P}(\mathtt{a_1 a_2 a_3 ... a_i}) &= \hat{P}(\mathtt{a_1})\hat{P}(\mathtt{a_2}|\mathtt{a_1})\hat{P}(\mathtt{a_3}|\mathtt{a_1 a_2})...\hat{P}(\mathtt{a_n}|\mathtt{a_1 ... a_{n-1}}) \\ &= \prod_{i=1}^{n} \hat{P}(\mathtt{a_i}|\mathtt{a_1 ... a_{i-1}}) \end{aligned} \tag{195}$$

Wir müssen also, für eine Markov-Kette n-ter Ordnung, alle Wahrscheinlichkeiten $\hat{P}(\mathtt{a}|\vec{\mathtt{w}}) : 0 \leq |\vec{\mathtt{w}}| \leq n$ schätzen. Mit diesem Wissen und einiger Mühe lässt sich natürlich zeigen:

$$\hat{P}(\mathtt{a}) := \sum_{|\vec{\mathtt{w}}|=n} \hat{P}(\mathtt{a}|\vec{\mathtt{w}})\hat{P}(\vec{\mathtt{w}}) = \frac{\mathtt{a}(\mathfrak{T})}{|\mathfrak{T}|}, \tag{196}$$

wie wir das erwarten.

Übung

Wir nehmen ein Markov Modell über $\Sigma = \{a, b\}$, das wie folgt spezifiziert ist (wir benutzen hier die eingeführten Kurzformen; $\ltimes$ steht für den Wortanfang, $\rtimes$ für das Ende):

- $P(a|\ltimes) = 0.4$
- $P(b|\ltimes) = 0.6$
- $P(\rtimes|\ltimes) = 0$
- $P(a|a) = 0.8$
- $P(b|a) = 0.1$
- $P(\rtimes|a) = 0.1$
- $P(a|b) = 0.2$
- $P(b|b) = 0.7$
- $P(\rtimes|b) = 0.1$.

Berechnen Sie die Wahrscheinlichkeit der folgenden Ereignisse in der dazugehörigen probabilistischen Sprache:

0. Ein Wort fängt an mit a
1. Der dritte Buchstabe in einem beliebigen Wort ist ein a.
2. Ein Wort hat n Buchstaben (bitte eine Formel über n!)

Übung

Bitte bearbeiten bis 21.6.2020

1. Nehmen Sie die Wahrscheinlichkeiten wie in der letzten Aufgabe. Wie ist die wahrscheinlich ist es, dass ein Wort mindestens ein b hat? Denken Sie an unsere Wahrscheinlichkeitsgesetze, und wie Sie diese Wahrscheinlichkeit clever (d.h. berechenbar) repräsentieren können!
2. Nehmen Sie folgendes Korpus: $\mathfrak{T} = \{aba, ababb, aabba, ababab, abbabab\}$. Schätzen Sie die Wahrscheinlichkeiten nach Maximum Likelihood für eine Markov Kette erster Ordung.

18 Wahrscheinlichkeiten schätzen

18.1 Die Likelihood-Funktion

Es gibt in der Stochastik/Statistik eine Unterscheidung zwischen Wahrscheinlichkeit (*probability*) und Likelihood, die man sich etwas schwierig klarmacht, da im Deutschen (und der englischen Umgangssprache) beide Begriffe zusammenfallen. In gewissem Sinne ist likelihood aber das Gegenteil (oder Gegenstück) zu Wahrscheinlichkeit. Intuitiv gesagt können wir von Wahrscheinlichkeit sprechen, wenn wir die zugrundeliegenden *Parameter* eines Experimentes kennen. Mit Parameter bezeichnet der Statistiker das, was der Stochastiker als Wahrscheinlichkeitsfunktion bezeichnet; die Parameter zu kennen bedeutet also: die zugrundeliegenden Wahrscheinlichkeiten zu kennen. Beispielsweise: wenn ich weiß dass eine Münze fair ist, als die Parameter des Experimentes kenne, kann ich fragen: was ist die *Wahrscheinlichkeit*, dass ich dreimal Zahl werfe? Wahrscheinlichkeit in diesem engeren Sinne bezeichnet also die Wahrscheinlichkeit eines Ereignisses, gegeben einen zugrundeliegenden, *bekannten* (oder als bekannt angenommenen) Wahrscheinlichkeitsraum. Wahrscheinlichkeit in diesem engeren Sinn haben wir ausführlich behandelt. Wenn wir das Ereigniss 3-mal Zahl als ω bezeichen, θ als die Wahrscheinlichkeit von Zahl im einfachen Bernoulli-Raum, P_θ als die Wahrscheinlichkeit im Produktraum, dann ist die Lösung $P_\theta = \theta^3$.

Wenn man das Beispiel verallgemeinert, dann ist die Wahrscheinlichkeit also eine Funktion, die jedem Ergebniss (jeder Beobachtung) einen Wert in $[0, 1]$ zuweist.

Likelihood bezeichnet dagegen die Plausibilität von zugrundeliegenden Wahrscheinlichkeitsräumen (sprich: Parametern), gegeben eine Reihe von Beobachtungen die wir gemacht haben. Beispielsweise: wir werfen eine Münzen 100mal, und werfen immer Zahl (nennen wir diese Beobachtung wieder ω. Was ist die Plausibilität dafür, dass der Würfel fair ist? Allgemeiner: was ist die Plausibilität für beliebige Parameter (sprich: zugrundeliegende Münzwahrscheinlichkeiten) gegeben ω? Wir haben auch ein solches Problem bereits einmal behandelt (siehe die 3. Sitzung). Dort haben wir versucht, zugrundeliegenden Parametern Wahrscheinlichkeiten zuzuweisen, und haben dabei gesehen, dass man das nicht ohne weitere Annahmen lösen kann: wir können zwar den Satz von Bayes benutzen um das Problem anzugehen, aber um es letztendlich zu lösen, brauchen wir einige zusätzliche Annahmen, und wir werden immer nur einen beschränkten Raum von Hypothesen

zulassen.

Eine andere Lösung ist die, dass man eben nicht Wahrscheinlichkeiten von Parametern sucht, sondern sich auf **Likelihood** beschränkt. Was wir nämlich machen können ist folgendes. Sei Θ die Menge aller möglichen Parameter für eine gegebene Beobachtung ω (also alle Wahrscheinlichkeitsräume, die zu dem Experiment passen). Θ ist also eine Menge von Wahrscheinlichkeitsräumen. Wir bekommen eine Funktion

$$L_\omega : \Theta \to [0, 1],$$

wobei

$$L_\omega(\theta) = P_\theta(\omega) \tag{197}$$

L_ω gibt uns also für jeden Parameter θ an, wie wahrscheinlich ω ist unter der Annahme dass der zugrundeliegende Parameter θ ist. L_ω ist die **Likelihoodfunktion**. Hier wird klar, warum wir hier nicht von Wahrscheinlichkeiten sprechen sollten: der Wert $L_\omega(\theta)$ gibt uns *nicht* die Wahrscheinlichkeit von θ; insbesondere gilt im allgemeinen Fall:

$$\int_{\theta \in \mathbb{R}} L_\omega(\theta) \neq 1 \tag{198}$$

(bei stetigen Verteilungen ist nicht die Summe 1, sondern das Integral). (*Aufgabe*: zeigen Sie dass mit einem Beispiel!) Wir können hier also nicht von Wahrscheinlichkeiten sprechen. Was uns $L_\omega(\theta)$ uns gibt ist Wahrscheinlichkeit von ω gegeben θ, also $P_\theta(\omega)$. Das ist eben qua Definition die *Likelihood* von θ gegeben ω, $L_\omega(\theta)$; wir können das mit Plausibilität übersetzen.

18.2 Maximum Likelihood Schätzung I

Warum ist Likelihood interessant, wenn sie uns am Ende nichts sagt, was wir nicht schon aus der Wahrscheinlichkeitsfunktion P erfahren? Der Grund ist folgender:

(199) $L_\omega(\theta_1) = P_{\theta_1}(\omega)$

sagt uns nichts über die Wahrscheinlichkeit von θ_1. Aber: Nehmen wir an, wir haben zwei mögliche zugrundeliegende Parameter θ_1, θ_2. Wir können nun deren Likelihood berechnen. Falls wir nun haben

(200) $L_\omega(\theta_1) \leq L_\omega(\theta_2)$,

dann sagt uns das sehr wohl etwas:

> Das Ergebnis ω unter der Annahme der Parameter in θ_2 wahrscheinlicher ist als unter der Annahme der Parameter θ_1.

Und daraus folgt: gegeben ω ist θ_2 wahrscheinlicher als θ_1 – wenn beide Parameter *apriori* gleich wahrscheinlich sind. Und das ist im Prinzip alles was wir wissen möchten: normalerweise interessiert uns nicht die genaue Wahrscheinlichkeit eines Parameters (im Normalfall: einer wissenschaftlichen Hypothese), uns interessiert was die beste Hypothese ist. Warum können wir das sagen? Die Korrelation von Lkelihood eines Parameters und seiner Wahrscheinlichkeit lässt sich aus dem Satz von Bayes herleiten. Wir schreiben

$$P_\theta(\omega) := P(\omega|\theta) = L_\omega(\theta)$$

Nun sei also $P(\omega|\theta_1) \leq P(\omega|\theta_2)$. Nach Bayes Theorem gilt:

$$\begin{aligned} P(\theta_i|\omega) &= P(\omega|\theta_i) \cdot \frac{P(\theta_i)}{P(\omega)} \\ \Leftrightarrow P(\theta_i|\omega)P(\omega) &= P(\omega|\theta_i)P(\theta_i) \end{aligned} \tag{201}$$

Nachdem $P(\omega)$ immer gleich ist für θ_1, θ_2 etc, spielt das für uns keine Rolle, und wir können es getrost weglassen. Wir haben daher:

(202) $P(\theta_i|\omega) \sim P(\omega|\theta_i)P(\theta_i)$,

wobei wir mit $\sim$ eine lineare Korrelation meinen: je größer der eine Term, desto größer der andere. Jetzt kommen wir vorerst nicht weiter, denn wir müssen immer noch die *a priori* Wahrscheinlichkeit der Parameter $P(\theta_i)$ berücksichtigen. Wir müssen also die Annahme machen, dass alle Parameter *a priori* gleich wahrscheinlich sind, was in vielen, jedoch nicht in allen Kontexten sinnvoll ist. (Z.B. wenn wir eine Münze finden, die absolut normal aussieht werden wir es für viel wahrscheinlicher halten, dass sie fair ist, als das sie eine starke Tendenz hat.) Dann fällt also auch der Term $P(\theta_i)$ weg (da er für alle $i = 1, i = 2...$ gleich ist), und wir haben:

(203) $P(\theta_i|\omega) \sim P(\omega|\theta_i)$.

Das ist genau was wir zeigen wollten: je größer $P(\omega|\theta_i) = L_\omega(\theta)$, desto größer ist $P(\theta_i|\omega)$, die Wahrscheinlichkeit der Parameter gegeben unsere Beobachtungen. Insbesondere gilt also:

(204) $L_\omega(\theta_1) \leq L_\omega(\theta_2)$

daher

(205) $P(\theta_1|\omega) \leq P(\theta_2|\omega)$

– natürlich nur unter der Annahme, dass alle Parameter *a priori* gleich wahrscheinlich sind.

Das führt uns zu der wichtigen Methode der Maximum Likelihood Schätzung. Wenn wir den Hypothesenraum Θ betrachten, dann haben wir natürlich mehr als zwei Hypothesen darin; genauer gesagt, im Normalfall werden wir *kontinuierlich viele* Parameter haben. "Kontinuierlich" bedeutet: "so viele, wie es reelle Zahlen gibt", ebenso wie abzählbar bedeutet: soviele wie die natürlichen Zahlen. Wir können uns also unmöglich hinsetzen und alle möglichen Parameter prüfen. Wir können also mittels Likelihood die Plausibilität von Hypothesen prüfen. Das nächste Problem ist: es gibt viel zu viele Hypothesen, als das wir sie alle prüfen könnten

Um das nächste Problem zu lösen brauchen wir zunächst etwas Notation. Sei

$$f : M \to \mathbb{R}$$

eine Funktion von einer beliebigen Menge in die reellen Zahlen (eigentlich reicht es schon, wenn die Menge linear geordnet ist, aber für uns spielt das keine Rolle). Dann definieren wir

(206) $argmax_{m \in M} f := \{m : \forall m' \in M,\ f(m') \leq f(m)\}$

D.h. argmax(f) liefert uns die $m \in M$, für die $f(m)$ maximal ist. Z.B.

(207) $argmax_{x \in \mathbb{R}}(-(x^2)) = 0$

da $f(x) = -(x^2)$ für $x = 0$ seinen größten Wert annimmt. $argmax_{x \in \mathbb{R}}(x^2)$ ist nicht definiert, da es für $f(x) = x^2$ keinen maximalen Wert $x \in \mathbb{R}$ gibt. $argmax(f)$ ist also nur definiert, wenn f *nach oben beschränkt* ist.

Die Maximum Likelihood Schätzung ist nun einfach die Methode, für ω und Θ den Parameter

(208) $argmax_{\theta \in \Theta} L_\omega(\theta)$

zu finden. Wie löst man dieses Problem? Nehmen wir an, unsere Beobachtungen sind binär, d.h. wir haben zwei mögliche Ergebnisse, und unsere Beobachtung ist eine Sequenz dieser Ergebnisse; nach unserer Konvention schreiben wir $\omega \in \{0, 1\}^n$. In diesem Fall ist Θ, unsere möglichen Parameter, eine Menge von Bernoulli-Räumen; und weil jeder Bernoulli-Raum ein möglicher Parameter für unsere Beobachtung ist, ist Θ (bis auf Isomorphie) die Menge *aller* Bernoulli-Räume (bis auf Isomorphie bedeutet: wir haben alle, wenn wir erlauben die beiden Elemente in $\Omega = \{0, 1\}$ beliebig anders zu benennen). Jeder Bernoulli-Raum ist (wieder bis auf Isomorphie) eindeutig charakterisiert durch $p := P(1)$; sobald dieser Wert gegeben ist, stehen alle anderen Dinge fest.

Das wiederum bedeutet: wir können jedem $\theta \in \Theta$ eine Zahl $p_\theta \in [0, 1]$ zuweisen. In diesem Fall können wir also Likelihood-Funktion

$$L_\omega : \Theta \to \mathbb{R}$$

auffassen als eine Funktion

$$L_\omega : \mathbb{R} \to \mathbb{R}.$$

Es lässt sich zeigen, dass diese Funktion stetig und differenzierbar ist (im Sinne der Analysis). Daraus wiederum folgt, dass wir die Maxima mit den klassischen Mitteln der Analysis bestimmen können (erste Ableitung gleich 0 setzen, prüfen ob es ein Extremwert ist). In diesem Fall lässt sich das Problem also lösen.

Was ist, wenn unsere Beobachtungen nicht einem Bernoulli-Raum entsprechen? Wenn wir beispielsweise einen Würfel 10mal werfen? Um in diesem Fall eine Maximum Likelihood Schätzung vornehmen zu können, müssen wir diesen Raum vereinfachen: für jedes der 6 möglichen Ergebnisse in Ω partitionieren wir die Menge der Ergebnisse in den zwei Ereignisse: für ω ein Ergebnis nehmen wir die Partition $\{\{\omega\}, \Omega - \omega\}$. So haben wir wiederum einen Bernoulli-Raum, wobei ein nicht-Bernoulli Experiment in eine Reihe von Bernoulli-Experimenten aufgeteilt wird.

18.3 Ein Beispiel

Wir werden nun ein einfaches Beispiel aus der statistischen Sprachverarbeitung betrachten. Nehmen wir an, wir betrachten ein Korpus mit 1.000.000 Wörtern, und finden darin 60mal das Wort `Hase`. Was uns interessiert ist die Wahrscheinlichkeit, mit der das Wort `Hase` in einem beliebigen Text auftritt. Wir möchten nun die MLS-Methode dafür anwenden. Wie machen wir das? Wir benennen

$$p = P(\texttt{Hase})$$

die Wahrscheinlichkeit des Wortes `Hase`; wir haben $q = 1 - p$, also einen Bernoulli-Raum. ω ist die Beobachtung die wir gemacht haben: dass nämlich in einem Text von 1.000.000 Wörtern 60 mal `Hase` vorkommt.

Was ist unsere Likelihood-funktion? Hier können wir unser wissen über Binomialverteilungen nutzen, und bekommen

$$L_\omega(\theta) = P_\theta(\omega) = L_\omega(p_\theta) = \binom{1.000.000}{60} p_\theta^{60} \cdot (1 - p_\theta)^{1.000.000-60} \tag{209}$$

Wenn uns nur das Maximum interessiert, können wir den Term $\binom{1.000.000}{60}$ außer Betracht lassen; wir suchen also, etwas allgemeiner ausgedrückt,

$$argmax_{p\in[0,1]}(p^n \cdot (1-p)^{m-n}), \text{ for } m \geq n \tag{210}$$

Das Ergebnis ist – wenig überraschend:

$$argmax_{p\in[0,1]}(p^n \cdot (1-p)^{m-n}) = \frac{n}{m} \tag{211}$$

In diesem einfachen Beispiel sagt uns also die MLS, dass die Wahrscheinlichkeitstheorie mit unseren Intuitionen über die Korrelation von Frequenz Wahrscheinlichkeit übereinstimmt. Das heißt natürlich nicht, dass $\frac{60}{1.000.000}$ die beste Schätzung der Wahrscheinlichkeit von `Hase` in einem beliebigen Text ist; aber es sagt uns dass es die plausibelste Schätzung ist gegeben die Beobachtung die wir gemacht haben.

18.4 Definitionen

Wir haben also folgende Definitionen:

Definition 13 *Sei Ω ein Bernoulli-Raum. Eine Schätzung ist eine Funktion $S_n : \Omega_n \to \Theta$, wobei Θ die Menge der möglichen Parameter ist.*

Sei $\Omega = \{0, 1\}$. Wir bezeichen, für $\vec{\omega} = \langle \omega_1, ..., \omega_n \rangle$, $f_1(\vec{\omega}) = \sum_{i=1}^{n} \omega_i$, und $f_0(\vec{\omega}) = n - \sum_{i=1}^{n} \omega_i$.

Definition 14 *Die Maximum-Likelihood Schätzung für $P(1)$ gegeben Ω^n ist die Funktion*

$$MLS_n(\omega) := \frac{f_1(\omega)}{n}$$

Der entscheidende Punkt ist der folgende, den wir bereits oben angedeutet, wenn auch nicht wirklich bewiesen haben:

Satz 15 *Für jeden Bernoulli-Raum Ω und alle zugrundeliegenden Wahrscheinlichkeiten θ gilt: für $\omega \in \Omega^n$, $MLS_n(\omega) = argmax_{\theta \in \Theta} P(\omega|\theta)$; anders gesagt, unter der Annahme, dass alle Parameter $\theta \in \Theta$ gleich wahrscheinlich sind, ist $P(MLS_n(\omega)|\omega)$ die wahrscheinlichste Hypothese.*

Das ist der Grund warum MLS_n die Maximum-Likelihood Schätzung genannt wird. Neben einer ganzen Reihe positiver Eigenschaften hat sie vor allen Dingen eine: sie ist sehr einfach zu berechnen.

RX10 Implementierung von Markov Ketten

1. Implementieren Sie eine Funktion, die für ein array von Worten (Eingabe-Variable!) die Markov-Wahrscheinlichkeiten (erster Ordnung) für diese Sprache nach Maximum-Likelihood schätzt.

2. Implementieren Sie die Funktion *score*, die die Wahrscheinlichkeit eines beliebigen Eingabewortes nach diesen Wahrscheinlichkeiten berechnet.

3. Im Schwellentest müssen wir H_0, H_1 komplett ausbuchstabieren. Wir sagen: H_0 ist: die Wahrscheinlichkeiten der einzelnen Buchstaben sind unabhängig, H_1 lautet: sie bilden eine Markov Kette erster Ordnung. Dann berechnen wir das likelihood-Verhältnis

   $$R(\mathfrak{T}) = \frac{P_0(\mathfrak{T})}{P_1(\mathfrak{T})} \tag{212}$$

   Als nächstes können wir einfach unseren Schwellentest anwenden:

   $$S_{0.05}(\mathfrak{T}) = \begin{cases} H_0, \text{ falls } R(\mathfrak{T}) \geq 0.05 \\ H_1 \text{ andernfalls.} \end{cases} \tag{213}$$

   Berechnen Sie das Ergebnis dieses Tests für folgende Beispiele:

   (a) $\mathfrak{T}_1 = ababababaabab$

   (b) $\mathfrak{T}_2 = abbaabbaabbaabbaabbaabab$

Hilfestellung

Wir brauchen erstmal eine Reihe von Befehlen. Die wichtigsten Befehle zur Manipulation von strings sind die folgenden:

> nchar("Testwort")

> substr("Testwort", start=2,stop=4)

Wichtig ist dass strings mit Anfürungszeichen markiert werden. Es gibt noch viele weitere Befehle für strings; die gibt es im Paket **stringr**:

> install.packages("stringr")

> library(stringr)

Hier gibt es eine Unmenge von Operationen auf strings. Für uns relevant ist erstmal nur **Konkatenation**; hierfür gibt es die Operation **str_c**:

> str_c("Test", "wort")

Diese Operation kann übrigens beliebig viele Argumente nehmen. Das reicht erstmal. Was noch ganz nett (aber nicht unbedingt nötig) ist, sind **Mengen**. Mengen gehören nicht zu den normalen Strukturen von R, daher sind sie erstmal nicht gegeben. Es gibt aber ein Paket **sets**, das uns alle wichtigen mengentheoretischen Strukturen und Operationen liefert. Das Paket findet sich hier:

https://cran.r-project.org/web/packages/sets/index.html

Wir laden nun sets_1.0-18.tar.gz herunter. Man installiert das mittels

> install.packages("Pfad", repos = NULL, type="source")

Oder einfach:

> install.packages("sets")

Wir möchten nun folgendes: gegeben ein **Wort** möchten wir die Markov-Ketten Wahrscheinlichkeiten schätzen nach Maximum Likelihood. Wie macht man das? Hier gibt es natürlich viele Möglichkeiten, wir nehmen folgende: wir definieren eine Funktion

$$count1(b|D) \mapsto \mathbb{N}$$

Das bedeutet soviel wie: wir nehmen unser gegebenes Wort $\overline{x}$ als ein Parameter der Funktion *count*, die uns sagen soll wie oft a in $\overline{x}$ vorkommt. Dasselbe geht natürlich mit

$$count2(ba|D) \mapsto \mathbb{N}$$

Damit können wir dann ganz einfach die Wahrscheinlichkeiten schätzen also

$$\hat{P}(a|b) = \frac{count2(ba)}{count(b)} \tag{214}$$

Damit haben wir die Wahrscheinlichkeit die wir brauchen. Was wir noch implementieren müssen ist folgende Funktion:

$$score : \Sigma^* \to [0, 1]$$

also die Funktion, die effektiv die Wahrscheinlichkeit der Markov-Kette berechnet. Das berechnet man natürlich einfach nach dem Schema:

(215) $P(a_1...a_n) = P(a_1|\ltimes)P(a_2|a_1)...P(\rtimes|a_n)$

Wir haben also folgenden Ablauf:

1. Gegeben ein Wort/array von Worten, berechne das Alphabet (nicht nötig, aber hilfreich)
2. Implementiere $count1, count2$
3. Schätze $\hat{P}(a|b)$ für $a, b \in$Alphabet
4. Implementiere die score-Funktion.

Wir fangen an mit der Funktion alphabet (nicht wirklich nötig, aber praktisch um gewisse Dinge zu visualisieren):

```
> alphabet <- function(x){
+     S1 = set();
+     for(i in 1:nchar(x)){
+       S1 = cset_union(S1,substr(x,start=i,stop=i));
+     }
+     return(S1);
+ }
```

Wir möchten die Häufigkeiten zählen. Dazu brauchen wir ein Testwort, das wir festlegen.

```
> word = "Testwort"
> uniCount <- function(x){
+     count = 0;
+     for(i in 1:nchar(word)){
+       if(x == substr(word,start =i,stop=i)){count=count+1}
+     }
```

```
+ return(count)
+ }
> uniCount("t")
```

Das schöne hieran ist: wir können nun einfach die Variable word ändern, und bekommen die neuen uniCounts:

```
> word = "Testworttest"
> uniCount(t)
```

Wir definieren jetzt die Menge der uniCounts; auch das ist nur für die Visualisierung und keinesfalls nötig.

```
> uniCountSet = set();
> for(x in alphabet(word)){
+     uniCountSet = cset_union(uniCountSet,c(x,uniCount(x)))}
```

In Mengen werden Vektoren nicht richtig angezeigt.

```
> for(x in uniCountSet){print(x)}
```

Was Sie machen müssen ist also noch:

- Counts für bigrame implementieren. Nicht vergessen $\rtimes, \ltimes$! (das läuft unter preprocessing)
- Wahrscheinlichkeiten schätzen (ML).
- Wahrscheinlichkeiten scoren.
- Likelihood-ratio berechnen. Hierfür brauchen Sie natürlich auch das Markov-Modell 0ter Ordnung, also H_0. Das ist aber sehr einfach zu berechnen mit dem gegebenen Code.

Noch eine letzte Sache: um die korrekten Bigram-Wahrscheinlichkeiten zu bekommen, brauchen wir $\rtimes, \ltimes$. Das mag etwas schwierig anmuten, ist aber ganz einfach: wir brauchen einfach einen sog. *preprocessing-step*: gegeben ein Wort über einem Alphabet, nehmen wir uns zwei Symbole die nicht in diesem Alphabet vorkommen. Eines schreiben wir vor das Wort, eines

dahiner (str_c!). Das schätzt uns die Wahrscheinlichkeit, die wir brauchen. Dasselbe machen wir, bevor wir ein Wort scoren.

18.5 Hausaufgabe 10

Bearbeiten Sie Aufgabe 1-3 oben bis zum 28.6.2021!
(Lösung auskommentiert)

19 Parameter glätten – Smoothing 1 (add one)

Wie wir gesehen haben ist die ML-Schätzung problematisch, wenn unsere Daten sehr dünn sind – was z.B. insbesondere bei Markov-Ketten höherer Ordnung unvermeidlich ist: wenn unser Lexikon Σ 10.000 Worte enthält, dann gibt es

(216) $10.000^5 = (10^4)^5 = 10^{20}$

5-grame – das sind enorm viele, und es ist fast ausgeschlossen dass wir einen repräsentativen Einblick in die Verteilung für seltene 5-grame bekommen. Wenn aber ein Ergebnis nicht beobachtet wurde, bekommt es nach ML-Schätzung die Wahrscheinlichhkeit 0 – ein sehr extremer Wert, den man oft in dieser Form nicht will, denn er absorbiert alle anderen Werte.

Deswegen benutzt man verschiedene Verfahren um Parameter zu **glätten**, d.h. solche extremen Werte zu vermeiden. Das einfachste Verfahren ist das sog. **add-one smoothing**, das darauf basiert dass wir einfach so tun, als hätten wir jedes Ergebnis *mindestens einmal* beobachtet. Die Schätzung (bleiben wir beim Beispiel der Markov-Kette) sieht dann wie folgt aus:

(217) $$\hat{P}_{\textit{add-one}}(a|w) = \frac{wa(\mathfrak{T}) + 1}{w(\mathfrak{T}) + |\Sigma|}$$

Wir nehmen also an dass w noch $|\Sigma|$-oft vorkommt, jedes mal gefolgt von einem anderen $a \in \Sigma$.

Diese Methode ist tatsächlich die einfachste um 0-Schätzungen zu vermeiden; sie ist allerdings oft kritisiert worden, aus folgendem Grund: *add-one-smoothing* sei so wie denen, die wenig haben, etwas wegzunehmen, um es denen zu geben, die gar nichts haben. Das ist natürlich bildlich gesprochen und bedeutet: durch diese Art von smoothing verschiebt sich viel Wahrscheinlichkeitsmasse von den selten gezählten zu den gar nicht gezählten. Nehmen wir beispielsweise an,

$$|D|_{wa} = 1,\ |D|_w = c|\Sigma|,\ |D|_{wb} = 0$$

Dann ist

(218) $$\hat{P}_{\textit{add-one}}(a|w) = \frac{2}{(c+1)|\Sigma|}$$

und

$$\hat{P}_{add\text{-}one}(b|w) = \frac{1}{(c+1)|\Sigma|} \quad (219)$$

Damit ist klar dass je größer c ist, desto kleiner ist die Differenz der beiden. Weiterhin können wir folgendes sehen: falls

$$|D|_{vb} = n,\ |D|_{v} = m|\Sigma|$$

und

$$\frac{n}{m} < \frac{1}{c} \quad (220)$$

dann ist

$$\hat{P}_{add\text{-}one}(b|w) > \hat{P}_{add\text{-}one}(b|v) \quad (221)$$

obwohl ersteres nie beobachtet wurde, letzteres möglicherweise durchaus häufig!

Ein weiteres Problem ist folgendes: nehmen wir wieder das obige Beispiel, wir haben ein Korpus mit 1.000.000 Worten (*token*) und 10.000 *types*, und wir möchten damit 5-grame schätzen. Aus einer Division ergibt sich, dass wir selbst im allerbesten Fall

$$\frac{10^{24}}{10^{6}} = 10^{20} \quad (222)$$

5-grame nicht beobachten können. Jedes dieser 5-grame wird dann eine Wahrscheinlichkeit bekommen, die wir natürlich nicht genau bestimmen können; allerdings zeigt schon eine kurze Überlegung, dass der **allergrößte Teil** der gesamten Wahrscheinlichkeitsmasse auf diese 5-grame entfällt, obwohl wir sie noch nie beobachtet haben. Das ist natürlich ein großes Problem, denn es macht unsere gesamten Schätzungen ziemlich wertlos, da die allermeiste Information völlig uninformiert vergeben wird. Es gibt also eine Reihe Probleme mit dieser Schätzung.

19.1 Add-one smoothing als ein bestimmes lineares Modell

Lineare Modelle, Marginalisierung Nehmen wir einmal an, wir haben zwei oder mehr Wahrscheinlichkeitsverteilungen über eine Menge Ω, und wir denken: alle sind relevant für uns, aber evtl. in verschiedenem Maße. Der einfachste Weg, diese beiden Funktionen zu verknüpfen ist die **gewichtete Summe**:

(223) $P_3(x) = \lambda_1 P_1(x) + \lambda_2 P_2(x)$

wobei $\lambda_1 + \lambda_2 = 1$. Auf diese Art und Weise sind wir sicher, dass

1. $P_3 : \Omega \to [0, 1]$ erfüllt, wobei
2. $\sum_{x \in \Omega} P_3(x) = 1$ (im diskreten Fall) bzw.
3. $\int_{-\infty}^{\infty} P_3(x)dx = 1$

Mit anderen Worten: wir bekommen eine neue Wahrscheinlichkeitsverteilung!

Das kann man beliebig generalisieren:

(224) $P(x) = \lambda_1 P_1(x) + ... + \lambda_i P_i(x)$

ist eine Wahrscheinlichkeitsverteilung, vorausgesetzt dass

1. $\sum_{n=1}^{i} \lambda_i = 1$
2. $P_1, ..., P_i : \Omega \to [0, 1]$ Wahrscheinlichkeitsverteilungen sind.

Man beachte dass das im Prinzip der Marginalisierung entspricht, wenn wir schreiben:

$$\begin{aligned} \lambda_j &\mapsto P(j) \\ P_j(x) &\mapsto P(x|j) \end{aligned}$$

Dann bekommen wir:

(225) $P(x) = P(x|1)P(1) + ... + P(x|i)P(i)$

und die Seitenbedingung entsprechen der Bedingung, dass $\{1, ..., 1\}$ eine Partition von Ω sind.

Man kann das auch auffasssen als ein lineares Modell: eine **lineare Funktion** in i Variablen ist eine Funktion der Form:

(226) $f(x_1, ..., x_i) = a_1 x_1 + ... + a_i x_i + b$

In unserem Fall sind die Verteilungen $P_1, ..., P_i$ die Variablen, bzw. entsprechen den Variablen, und werden linear Verknüpft. Das hat viele Vorteile:

- Falls die Verteilungen $\hat{P}_j$ nur geschätzt sind, dann ist die *Modellkomplexität* ein wichtiger Faktor.
- Lineare Modelle präservieren die Komplexität aller relevanten Modelle.

Aufgabe Wir schätzen Bigramm-Wahrscheinlichkeiten mit Add-One-Smoothing mit

$$\hat{P}_{Add1}(w_i \mid w_{i-1}) = \frac{count(w_{i-1}w_i) + 1}{count(w_{i-1}) + |V|}$$

Zeigen Sie, dass dies ein lineares Modell (gewichtete Summe) aus

1. der Maximum-Likelihood-Schätzung und
2. der Gleichverteilungschätzung (jedes Wort in V is gleich wahrscheinlich)

ist, also dass

(227) $\hat{P}_{Add1}(w_i \mid w_{i-1}) = \lambda_1 \hat{P}_{MLE}(w_i \mid w_{i-1}) + \lambda_2 \frac{1}{|V|}$

mit $\lambda_1 + \lambda_2 = 1$.

Lösung Wir setzen (zur Vereinfachung der Terme) $count(w_{i-1}) = C_1$, $count(w_{i-1}w_i) = C_2$, $|V| = v$. Wir haben nun ein lineares Gleichungssystem mit zwei Gleichungen:

$$\frac{C_2 + 1}{C_1 + v} = \lambda_1 \frac{C_2}{C_1} + \lambda_2 \frac{1}{v} \quad (228)$$

$$\lambda_1 + \lambda_2 = 1 \quad (229)$$

Lösen zunächst die erste Gleichung auf (bzw. vereinfachen):

$$\frac{C_2+1}{C_1+v} = \frac{C_2}{C_1+v} + \frac{1}{C_1+v} \tag{230}$$

Was wir suchen ist also

$$\lambda_1 \frac{C_2}{C_1} = \frac{C_2}{C_1+v} \Longleftrightarrow \lambda_1 = \frac{C_1}{C_1+v} \tag{231}$$

und

$$\lambda_2 \frac{1}{v} = \frac{1}{C_1+v} \Longleftrightarrow \lambda_2 = \frac{v}{C_1+v} \tag{232}$$

Jetzt können wir leicht prüfen dass

$$\lambda_1 + \lambda_2 = \frac{C_1}{C_1+v} + \frac{v}{C_1+v} = \frac{C_1+v}{C_1+v} = 1 \tag{233}$$

Also haben wir hier bereits die Lösung!

20 Parameter glätten – Good-Turing smoothing (vereinfacht)

Good-Turing smoothing wurde von Alan Turings Assistenten Good entwickelt, um den deutschen Enigma-Kode zu entschlüsseln. Die Grundlage dieser Methode ist folgende: anstatt zu fragen:

> Wie ist die Wahrscheinlichkeit, ein Objekt der Art x zu treffen?

Fragen wir nun:

> Wie ist die Wahrscheinlichkeit, ein Objekt der Häufigkeit n zu treffen?

Z.B.: wie wahrscheinlich ist es, ein Objekt zu sehen, das die Häufigkeit 210 hat? Der Sinn dahinter ist folgender: uns interessiert natürlich insbesondere die Frage:

> Wie ist die Wahrscheinlichkeit, ein Objekt zu treffen, das wir noch nie zuvor beobachtet haben?

Denn diese Wahrscheinlichkeit ist genau diejenige, die wir den Objekten zuweisen wollen, die wir in unseren Daten nicht beobachtet haben. Und das schöne ist: das lässt sich sehr einfach schätzen: den jedesmal, wenn wir ein Objekt nur einmal beobachtet haben, haben wir eine neue Beobachtung gemacht; wir nehmen hierfür also einfach die (nach Maximum likelihood) geschätzte Wahrscheinlichkeit, ein Objekt nur einmal zu beobachten. Das Problem ist allerdings, eine konsistente Wahrscheinlichkeitsverteilung daraus zu bekommen.

Nehmen wir also folgende Definitionen:

- $\mathfrak{T}$ ist unser Datensatz.
- $G = |\mathfrak{T}|$ die Gesamtzahl der Beobachtungen.
- L ist die Menge der Beobachtungen, die wir gemacht haben (also die Menge der beobachteten types!).
- $D : L \to \mathbb{N}$ ist eine Menge von Paaren, die jeder Beobachtung ihre Häufigkeit zuweist.

- $N : \mathbb{N} \to \mathbb{N}$ ist eine Funktion ("Häufigkeiten der Häufigkeiten"), die jeder Häufigkeit einer Beobachtung ihre Häufigkeit zuweist; formal und besser verständlich:

$$N(n) = |\{x \in L : D(x) = n\}|$$

Die einfache Good-Turing Schätzfunktion funktioniert nun wie folgt: wir schätzen die Wahrscheinlichkeit eines nicht-beobachteten Ereignisses als

$$\hat{P}(neu) = \frac{N(1)}{G} \quad (234)$$

Dem liegt die Überlegung zugrunde, dass eine neue Beobachtung immer darin resultieren würde, diese Beobachtung einmal gemacht zu haben, also nehmen wir einfach die geschätzte Wahrscheinlichkeit hiervon. Wohlgemerkt: hier handelt es sich nicht um die Wahrscheinlichkeit *eines* unbeobachteten n-grams, sondern um die Wahrscheinlichkeit *aller*, die wir nicht beobachtet haben. Nehmen wir also an, es gibt c n-grame, die wir nicht beobachtet haben. Dann wäre also, für x ein solches n-gram,

$$\hat{P}(x) = \frac{\hat{P}(neu)}{c} = \frac{N(1)}{G \cdot c} \quad (235)$$

Denn in Ermangelung von weiterem Wissen sollten wir alle diese n-grame für gleich wahrscheinlich halten. Nun ist die Frage: wir schätzen wir die übrigen Wahrscheinlichkeiten? Hier lautet die Antwort: wir setzen

$$\hat{P}(neu) := \hat{P}(0) \quad (236)$$

– wir schätzen also die Wahrscheinlichkeiten von Häufigkeiten von Häufigkeiten, und verallgemeinern:

$$\hat{P}(n) = (n+1) \cdot \frac{N(n+1)}{G} \quad (237)$$

Wir schätzen also allgemein die Wahrscheinlichkeit von n mittels der Häufigkeit von $n+1$. Die Motivation hierfür ist folgende: wenn wir eine Beobachtung machen aus der Klasse von Objekten, die wir n mal beobachtet haben, dann haben wir sie $n+1$-Mal beobachtet – also nehmen wir diese Wahrscheinlichkeit. Der Term $n+1$ steht hier für die Tatsache, dass die Zahl $N(n+1)$

mit $n+1$ multipiziert werden muss, um die wahren Häufigkeiten abzubilden, wir brauchen das also für die Konsistenz. Für eine bestimmte Klasse von Beobachtungen $x \in L$ bekommen wir dann:

$$\hat{P}(x) = (n+1) \cdot \frac{N(n+1)}{G \cdot N(n)} \quad (238)$$

denn wir müssen die Wahrscheinlichkeiten wieder gleich verteilen.

Soweit, so gut – diese Schätzfunktion wird uns eine konsistente Wahrscheinlichkeitsfunktion liefern. Es gibt aber ein großes Problem hierbei: die Dünne der Daten (English: *sparseness*. Das ist ein großes Problem, wenn wir z.B. Zipf-verteilte Daten haben, dann kann das wie folgt aussehen:

n	$N(n)$
1	427
2	285
3	157
4	103
...	...
401	1
402	0
403	1

Hier haben wir ein Problem: nehmen wir an, a ist die Klasse so dass $N(a) = 401$). Nun gilt:

$$\hat{P}(a) = 402 \cdot \frac{N(402)}{G \cdot N(401)} = 402 \cdot \frac{0}{G} = 0 \quad (239)$$

Das ist natürlich Unsinn und völlig daneben. Das zugrundeliegende Problem ist: die Wahrscheinlichkeit von a hängt ab von der Wahrscheinlichkeit, mit der Objekte mit Häufigkeit $D(a)+1$ auftreten. Dadurch dass es keine solchen Objekte gibt, bekommen wir 0 als Ergebnis. Und es ist erst an dieser Stelle, dass Good-Turing smoothing kompliziert wird. Es gibt für dieses Problem 2 Lösungen:

1. Anstatt der Funktion N, die auf den Daten basiert und evtl. Lücken hat nutzen wir $S(N)$, was eine (lineare) Approximation von N darstellt.

$S(N)$ ist also eine lineare Funktion (allgemeiner: ein Polynom), das sich ähnlich wie N verhält, aber eben keine Lücken hat. In diesem Fall bekommen wir die Schätzfunktion:

$$\hat{P}(n) = (n+1) \cdot \frac{S(N)(n+1)}{G \cdot NS(N)(n)} \tag{240}$$

Die Funktion N kann man normalerweise gut und effektiv approximieren mittels **linearer Regression**, die wir später behandeln werden.

2. Wir konstruieren eine Mischung aus einfachem Good-Turing und der approximierten Funktion. Das ist besser, aber deutlich komplizierter, denn wir müssen entscheiden, a) wann wir welche Methode wählen, und b) die Wahrscheinlichkeiten addieren nicht auf 1, wir müssen also **normalisieren**.

Das gute ist: es gibt fertige Pakete, die verschiedene Methoden von Good-Turing Schätzung fertig implementiert haben; das ganze von 0 an zu machen ist relativ kompliziert.

21 Parameter schätzen – Bayesianisch

21.1 Uniformes Apriori

Im Allgemeinen gilt, wie wir gesehen haben, für eine Hypothese H bezüglich der zugrunde liegenden Wahrscheinlichkeiten, D eine Reihe von Beobachtungen wir gemacht haben,

$$(241)\quad P(H|D) = P(D|H)\frac{P(H)}{P(D)} \propto P(D|H)P(H)$$

erstmal wegen Bayes, und zweitens weil der Term $P(D)$ unabhängig ist von H, also für die Suche eines Maximums über H (und viele andere Operationen) keine Rolle spielt.

Wie wir gesehen haben, basiert die "orthodoxe" Schätzung auf der Likelihood, die wiederum darauf basiert, die *a priori*-Wahrscheinlichkeit $P(H)$ zu unterdrücken: wenn wir annehmen, dass wir keine Informationen über $P(H)$ haben, können wir den Term auch weglassen:

$$(242)\quad P(D|H)P(H) \propto P(D|H) = L_P(H|D)$$

So kommen wir von der Wahrscheinlichkeit zur Likelihood von H, nämlich $P(D|H)$. In der bayesianischen Auffasseung gibt es aber praktisch immer eine *a priori* Information, die wir O nennen. Wir haben also:

$$(243)\quad P(H|DO) = P(D|HO)\frac{P(H|O)}{P(D|O)}$$

Bayesianische Parameterschätzung basiert nicht auf der

$$\text{Likelihood } P(D|H)$$

sondern auf der

$$\textit{aposteriori}\text{-Wahrscheinlichkeit } P(H|DO).$$

Information geht für uns niemals verloren, auch nicht durch unsere Beobachtung D, dementsprechend müssen wir über O Rechnung ablegen. Information kann zwar irrelevant werden – aber im Allgemeinen gibt es keinen Grund dafür! In unserem Fall besteht O darin, dass wir keine weitere spezielle Information bezüglich der Wahrscheinlichkeit von Ereignissen; die Frage ist,

wie wir das in formale Wahrscheinlichkeitsverteilung transformulieren. Der einfachste Fall ist (wie immer) die **uniforme Verteilung**, auch wenn man in manchen Fällen besser davon abweicht. Wie schätzen wir also die *aposteriori*-Wahrscheinlichkeiten?

Nehmen wir das Beispiel eines Textes mit Worten $a, b, c, ...$ Unser Text D ist die Beobachtung, die wir machen. Wir haben bereits gesehen, dass die Maximum-Likelihood von a errechnet wird durch

$$\frac{|D|_a}{|D|} \tag{244}$$

Wie geht die Schätzung bayesianisch? Zunächst folgende Konvention: wir nennen θ_a das, was wir vorher $\hat{P}(a)$ genannt haben, also

$$\theta_a \triangleq \hat{P}(a)$$

Der Vorteil hiervon ist: wir können nun θ_a als Zufallsvariable auffassen, die Werte in $[0, 1]$ annimmt mit einer unterschiedlichen Wahrscheinlichkeit.

$$P(\theta_a = x|DO) = P(D|\theta_a = x, O)\frac{P(\theta_a = x|O)}{P(D|O)} \tag{245}$$

$P(D|\theta_a = x, O)$ ist für uns die Wahrscheinlichkeit der Daten gegeben unsere Wahrscheinlichkeit von a ist x. Wohlgemerkt:

$$P(\theta_a = x|O) \neq \frac{1}{|\Sigma|} \tag{246}$$

die uniforme Verteilung verlangt vielmehr dass

$$\int_0^1 P(\theta_a = x|O)d(x) = 1 \tag{247}$$

und

$$\text{für alle } x, y \in [0, 1],\ P(\theta_a = x|O) = P(\theta_a = y|O) \tag{248}$$

$P(D|O)$ ist ein Term, der unabhängig ist von θ_a (und allen anderen Parametern). Wir haben also

$$P(\theta_a = x|D, O) = P(D|\theta_a = x, O)\frac{1}{C} \tag{249}$$

wobei C eine Normalisierungs-Konstante ist die unabhängig ist von allen Parametern. Wohlgemerkt ist

$$P(\theta_a|DO)$$

nicht die *aposteriori*-Wahrscheinlichkeit von a, sondern vielmehr eine Wahrscheinlichkeitsverteilung über mögliche Werte von θ_a.

Das heißt die eigentliche Schätzung steht natürlich noch aus, Hier hat man wieder dieselben Möglichkeiten wie vorher, und in unserem Fall läuft auf die ML-Methode hinaus.

21.2 Kein uniformes Apriori

Man kann sich fragen, worin die Bedeutung unseres *a priori* liegt, wenn er darin besteht, dass wir keine relevante Information haben. Die Antwort ist folgende: wie wir bereits gesagt haben, erwarten wir dass unsere Beobachtungen normalerweise *extremer* sind, als die zugrundeliegende Verteilung, insbesondere dort, wo wir wenige Beobachtungen haben. Während wir in der orthodoxen Likelihood eben *ad-hoc* eine Lösung dafür finden müssen, ist die Lösung in der bayesianischen Methode bereits eingebaut, nämlich mittels eine gut gewählten *apriori*-Verteilung.

Unsere *apriori* sagt uns bereits, welche Häufigkeiten wir erwarten, und jede Beobachtung, die davon abweicht, wird dadurch gemildert. Insbesondere wird – wenn die *apriori*-Wahrscheinlicheit > 0 ist, die *aposteriori*-Wahrscheinlichkeit ebenso > 0 sein; somit können wir sehr extreme Ergebnisse ausschließen.

Bisher hatte sich, durch unser uniformes Apriori, nichts geändert an der Schätzung, da das *apriori* nur als Konstante eingeflossen ist. Das ist anders wenn wir eine apriori-Verteilung wählen, die nicht uniform ist, und die beispielsweise berücksichtigt, dass extreme Werte unwahrscheinlicher sind als "mittlere" Werte. Ein Beispiel hierfür ist:

$$P(\theta_a = x|O_1) = C(x \cdot (1-x)) \tag{250}$$

wobei C eine **Normalisierungs-Konstante** ist, und O_1 das entsprechende *apriori* ist. Dabei entspricht O_1 dem Wissen, dass wir "gemäßigte" θ_a bevorzugen. Die Verteilung ist symmetrisch, mit einem Maximum an

$$P(\theta_a = 0.5|O_1) = 0.25 \tag{251}$$

und, was sehr wichtig ist,

$$P(\theta_a = 0|O_1) = P(\theta_a = 1|O_1) = 0 \tag{252}$$

d.h. unser apriori schließt aus, das die Wahrscheinlichkeit θ_a je 0 wird. Wichtig ist: die Regeln der Wahrscheinlichkeitstheorie sind so, dass wenn etwas kategorisch ausgeschlossen ist (wie $\theta_a = 0$) sich dass durch keine Beobachtung ändert! Das kann sinnvoll sein, wenn wir eine Münze werfen oder ähnliches: denn es ist *apriori* viel wahrscheinlicher, das der korrekte Parameter irgendwo in der Mitte liegt, während es an den Rändern immer unplausibler wird den korrekten Parameter zu finden.

Um zu sehen, dass $P(\theta_a|O_1)$ das eine Wahrscheinlichkeitsverteilung ist, müssten wir noch zeigen dass

$$(253)\ \int_0^1 P(\theta_a = x|O_1)\ d(x) = \int_0^1 C \cdot x \cdot (1-x) d(x) = 1$$

(wie ging das nochmal?)

$$(254)\ \int x \cdot (1-x) d(x) = \int -x^2 + x\ d(x) = -\frac{x^3}{3} + \frac{x^2}{2}$$

Dementsprechend:

$$(255)\ \int_0^1 x \cdot (1-x) d(x) = (-\frac{1^3}{3} + \frac{1^2}{2}) - (-\frac{0^3}{3} + \frac{0^2}{2}) = \frac{1}{2} - \frac{1}{3} = \frac{1}{6}$$

Das bedeutet, dass das Integral unserer Wahrscheinlichkeit nur 1/6 beträgt, wir brauchen also

$$(256)\ C = 6$$

Das bedeutet aber:

$$(257)\ P(\theta = 0.5|O_1) = 1.5$$

d.h. es ist keine echte Wahrscheinlichkeit mehr. Das ist aber kein Problem, wir müssen uns nur erinnern: wir haben jetzt kontinuierliche Verteilungen, die Wahrscheinlichkeit an einem **Punkt** ist immer 0; was interessant ist, ist die Masse in einem Integral.

Nun nehmen wir einmal an, wir haben a in unseren Daten D kein einziges Mal beobachtet. Wir haben nun zwei relevante Faktoren:

$$(258)\ lim_{\theta_a \to 0} P(D|\theta_a, O_1) = 1$$

d.h. für $\theta_a \to 0$ geht die Wahrscheinlichkeit von D gegen 1; aber:

$$(259)\ lim_{\theta_a \to 0} P(\theta_a|O_1) = 0$$

Das bedeutet: wenn wir

$$(260)\ P(\theta_a|DO_1)$$

maximieren möchten, dann reicht es nicht, $P(D|\theta_a, O_1)$ zu maximieren! Insbesondere werden wir niemals ein Maximum bei $\theta_a = 0$ haben.

Wo der genaue Wert landet, hängt also von der Interaktion der Verteilungen ab (anders als bei ML): insbesondere hängt es davon ab, wie oft wir das Experiment wiederholen. Das ist intuitiv klar: je mehr Beobachtungen machen, desto unwichtiger wird unser apriori. Gehen wir das an einem ganz konkreten Beispiel durch: Sei $|D_1| = 5$, $|D_1|_a = 0$. Dann haben wir

$$P(\theta_a = x|D_1, O) = ((1-x)^5)(x(1-x))C \tag{261}$$

C ist eine Normalisierungskonstante (ergibt sich aus $P(D|O_1)$) die unabhängig von θ_a gleich bleibt. Wir brauchen also:

$$argmax_{0\leq x\leq 1}((1-x)^5)(x(1-x)) = argmax_{0\leq x\leq 1}((1-x)^6)x \tag{262}$$

Wir haben

$$\frac{d}{d(x)}((1-x)^6)x = (x-1)^5(7x-1) \tag{263}$$

und für

$$(x-1)^5(7x-1) = 0 \tag{264}$$

gibt es die Lösungen $x = 1$ und $x = \frac{1}{7}$, wobei ersteres natürlich kein Maximum sein kann. Also ergibt die ML-Schätzung in diesem Fall

$$\hat{P}_{ML}(\theta_a) = \frac{1}{7} \tag{265}$$

Nun nehmen wir an, wir haben in D_2 20 Beobachtungen gemacht, und a war immer noch nicht darunter. In diesem Fall ist alles wie gehabt, nur ein Parameter ändert sich:

$$P(\theta_a = x|D_2, O_1) = (1-x)^{20}(x(1-x))C \tag{266}$$

Uns interessiert also

$$argmax_{0\leq x\leq 1}(1-x)^{21}x \tag{267}$$

Also:

$$\frac{d}{d(x)}(1-x)^{21})x = (x-1)^{20}(22x-1) \tag{268}$$

dann setzen wir

$$(269)\ \ (x-1)^{20}(22x-1)=0 \Leftrightarrow x=\frac{1}{22}$$

Das Muster ist leicht zu erkennen: für n Beobachtungen, von denen keine a ist, schätzen wir mit ML Schätzung und unserem konvexen apriori

$$(270)\ \ \hat{P}_{ML}(\theta_a)=\frac{1}{n+2}$$

– also bei $n=0$, $\theta_a=\frac{1}{2}$, wie es sein sollte. Wir sehen also, dass unser konvexes Apriori das Smoothing vollkommen unnötig macht.

Umgekehrt, nehmen wir an wir bekommen ein Ergebnis wie

$$(271)\ \ \frac{|D|_a}{|D|}=\frac{1}{3}$$

d.h. die (orthodoxe) ML-Schätzung liegt relativ nahe am apriori-wahrscheinlichsten Wert. Wie ist nun der aposteriori maximale Wert? Das hängt wiederum von $|D_3|$ ab; sagen wir $|D_3|=21$. Dann haben wir

$$(272)\ \ P(\theta_a=x|D_3,O_1)=\binom{21}{7}(1-x)^{14}x^7(1-x)x\frac{1}{C}=\binom{21}{7}(1-\theta_a)^{15}\theta_a^8\frac{1}{C}$$

Uns interessiert also

$$(273)\ \ argmax_{0\leq x\leq 1}\binom{21}{7}(1-x)^{15}x^8$$

Das errechnet sich aus

$$(274)\ \ \frac{d}{d(x)}\binom{21}{7}(1-x)^{15}x^8=0 \Leftrightarrow x=\frac{8}{23}$$

Wir weichen also nur um

$$(275)\ \ \frac{8}{23}-\frac{1}{3}=\frac{24}{69}-\frac{23}{69}=\frac{1}{69}$$

von der “orthodoxen” ML-Schätzung ab. Allgemeiner gesagt

$$\frac{d}{d(x)}\binom{n}{k}(1-x)^{n-k}x^k = 0 \tag{276}$$
$$\Leftrightarrow x = \frac{k+1}{n+2}$$

D.h. unser konvexes Apriori gibt uns eine Schätzung

$$\hat{P}(\theta_a|DO_1) = \frac{|D|_a + 1}{|D| + 2} \tag{277}$$

Also in auch in diesem Fall ist die Berechnung sehr einfach und hat den Vorteil, dass wir keinerlei weitere Methoden brauchen, um sehr extreme Ergebnisse abzumildern.

22 Numerische Parameter und Alternativen zu ML

ML für Erwartungswerte Nehmen wir einmal an, wir schätzen einen stetigen Parameter, also anstellen von θ_a (für $a \in \Sigma$) oder θ_x (für $x \in \{0,1\}$) schätzen wir θ_x (für $x \in [0,10]$). Das macht erstmal nicht so viel Sinn – wir müssten ja unendlich viele Parameter schätzen. Das macht aber durchaus Sinn wenn wir einen Erwartungswert schätzen: nehmen wir an, wir haben eine Zufallsvariable X deren Erwartungswert wir schätzen möchten. Das wäre z.B.: wir treffen Menschen und fragen Sie nach Ihrem Alter. Wir möchten den Altersschnitt schätzen, suchen also den Erwartungswert von X.

Wenn wir die zugrundeliegenden Wahrscheinlichkeiten kennen würden, dann müssten wir einfach nur $\mathcal{E}(X)$ berechnen; allerdings können wir uns nur auf eine Stichprobe berufen. Natürlich können wir einfach folgendes machen: sei D unser Datensatz, der wie folgt aussieht:

Alter	Anzahl
1	3
2	2
3	-
4	4
....	
91	1
92	1

D besteht also aus Paaren (n,m); ausserdem sei G die Gesamtgröße unserer Stichprobe, also

$$G = \sum_{(n,m)\in D} n \tag{278}$$

Der naheliegendste Ansatz wäre also:

$$\langle X\rangle_{ML} = \sum_{(n,m)\in D} \frac{n}{G} m = \frac{1}{G} \sum_{(n,m)\in D} nm \tag{279}$$

Diese Art den Erwartungswert zu berechnen entspricht der ML-Schätzung; das sieht man wie folgt: erinnern wir uns dass

$$P(X = n) = P(X^{-1}(n)) \tag{280}$$

Weiterhin ist

$$(281)\ \ \mathcal{E}(X) = \sum_m m \cdot P(X = n)$$

Wir sehen dass nach ML-Schätzung

$$(282)\ \ \hat{P}_{ML}(X^{-1}(n)) = \frac{m}{G} :\ (n, m) \in D$$

und dementsprechend ist (279) nichts anderes als der Erwartungswert von $\mathcal{E}(X)$ mit der unterliegenden Wahrscheinlichkeit $\hat{P}_{ML}$ nach ML geschätzt.

Least squared error ML ist durchaus sinnvoll im Szenario mit Alter. Allerdings haben wir bereits besprochen das diese Form der Schätzung Schwächen aufweist; insbesondere ist ihr vollkommen gleich, ob der Wert $\langle X \rangle_{ML}$ tatsächlich auftritt (in unserem Fall wahrscheinlich nicht – wir messen Alter in ganzen Zahlen, $\langle X \rangle_{ML}$ wird aber aller Voraussicht nach keine ganze Zahl sein). In unserem Fall lässt sich das durch Rundung beheben; im allgemeinen Fall ist das schwierig aufzulösen. Nehmen wir einmal folgendes an:

> wir sollen $\langle X \rangle_2$ auf eine gewisse Art schätzen, und jedesmal wenn ein neu gemessener Wert von unserer Schätzung abweicht, kostet uns das Geld (und zwar in Form einer Funktion über den Grad der Abweichung).

Wir haben also ein Interesse daran die Abweichung so gering als möglich zu halten. An dieser Stelle kommen wir zurück auf den Begriff der Varianz:

$$(283)\ \ V(X) = \mathcal{E}((X - \mathcal{E}(X))^2)$$

Die Varianz misst, wieviel wir im Quadrat erwarten abzuweichen von unserem Erwartungswert. Die Methode des **kleinsten quadratischen Fehlers** schätzt $\langle X \rangle_{LSE}$ so, dass die quadratische Abweichung der Daten von $\langle X \rangle_{LSE}$ minimal ist:

$$(284)\ \ \langle X \rangle_{LSE} = argmin_{x \in \mathbb{R}} \sum_{(n,m) \in D} ((x - n) \cdot m)^2$$

Das bedeutet: wir möchten die Abweichungen (im Quadrat) von unserem Wert minimieren. Das Quadrat kommt natürlich erstmal daher, dass wir positive Werte möchten. Im Allgemeinen hat die LSE-Schätzung einen bedeutenden Vorteil vor der ML-Schätzung:

- Die LSE-Schätzung ist sensibel für Abweichungen vom geschätzten Wert und versucht sie zu vermeiden;
- der ML-Schätzung sind Abweichungen egal, solange sie sich "ausgleichen".

Das Quadrat hat aber noch eine weitere Auswirkung: extreme Abweichungen werden stärker bestraft als geringe, d.h. "Ausreißer" werden überproportional gewichtet. Nehmen wir einmal an,

(285) $\langle X \rangle_{ML} = 38,$

allerdings gibt es in D einen Ausreißer $(120, 1)$ – wir haben also einen 120-jährigen getroffen! Das würde uns also für die LSE-Schätzung eine Abweichung von

(286) $(38 - 120)^2 = 6724$

liefern – und damit mehr ins Gewicht fallen als 6 70-jährige!

(287) $6 \cdot (38 - 70)^2 = 6 \cdot 1024 < 6724$

Das kann in manchen Fällen gewünscht sein, in anderen ist es das nicht. Wir haben also folgende Nachteile:

- Die LSE-Schätzung ist sehr anfällig gegenüber Ausreißern – sie mißt ihnen großes Gewicht bei;
- und sie ist deutlich komplizierter zu berechnen (auch wenn das heutzutage kein Problem mehr sein sollte.

Mediane Nehmen wir jetzt nun folgendes Szenario an: anstelle des Alters der Personen, die wir treffen, haben wir ihr Einkommen. Was ändert das? Nun, wie wir wissen sind die Einkommen grundsätzlich anders verteilt als die Altersstruktur: insbesondere haben wir eine normalerweise eine **Zipf-Verteilung**, d.h. sehr wenige sehr reiche Leute, und eine große Anzahl wenig wohlhabender Leute. Daraus folgt dass sowohl für ML- als auch für LSE-Schätzung der Wert stark nach oben gezogen wird: weder

$$\langle X \rangle_{ML}$$

noch

$$\langle X \rangle_{LSE}$$

liefern uns einen vernünftigen Wert für das Einkommen einer Person, der wir zufällig auf der Straße begegnen – die extremen Werte werden einfach zu stark berücksichtigt. Hier kann die **Median-Schätzung** hilfreich sein: der Median von D ist folgender Wert (D eine Menge von Paaren (n, m) mit (n =Einkommen,m =Anzahl der Verdiener, $G = |D|$),

$$\begin{aligned} & med(D) = m \Leftrightarrow \\ \text{(288)} \quad & \text{es gibt } \frac{G-1}{2} \text{ Datenpunkte } n' : n' \leq n \& \\ & \frac{G-1}{2} \text{ Datenpunkte } n' : n' \geq m \end{aligned}$$

Das liefert zumindest in diesem Fall eine vernünftige Schätzung: denn das Einkommen einiger weniger Superreichen hat ja tatsächlich keinen Einfluß darauf, welches Einkommen wir einem Menschen, den wir zufällig begegnen, zumessen!

Wenn wir also eine Zipf-Verteilung haben, dann wird uns die Media-Schätzung mit ziemlicher Sicherheit einen Wert am unteren Ende der Verteilung liefern.

23 Parameter für offene Skalen schätzen

23.1 Einleitung

Wenn wir – kontinuierliche oder diskrete – Skalen haben, die nach oben offen sind, gibt es einige besondere Dinge zu beachten. Das sieht man beispielsweise an folgendem Rätsel: Sie sitzen/liegen im Nachtzug, schlafen, wachen irgendwann auf, schauen auf dem Fenster. Sie sehen eine Straße mit Häusern, Sie sind also in einer Stadt, haben aber keine Anhaltspunkte für die Größe der Stadt. Sie sehen auch ein Taxi mit der Nummer 32 (nehmen wir an Taxis einer Stadt sind durchnummeriert). Sie sollen nun schätzen wie viele Taxis es gibt. Was schätzen Sie?

- ML sagt Ihnen: 32, denn das maximiert natürlich die Likelihood ihrer Beobachtung. Aber etwas an dieser Schätzung widerspricht unserer Intuition: ist es nicht unwahrscheinlich dass wir genau das "letzte" Taxi sehen?

- Intuitiv plausibler ist 64. Aber warum? Dazu müssen wir den Erwartungswert berücksichtigen: wenn wir 64 Taxis haben, alle gleich wahrscheinlich zu beobachten, dann liegt der Erwartungswert unserer Beobachtungen bei 32:

$$\sum_{i=1}^{64} i \cdot \frac{1}{64} = 32 \tag{289}$$

Warum haben wir in diesem Fall auf einmal Intuitionen, die so stark gegen ML sprechen? Der Grund liegt in der Natur des Parameter und Beobachtungen: dadurch dass wir wissen, dass es sich um eine Skala handelt, deren Parameter in eine Richtung offen sind, wissen wir auch, dass ML automatisch denjenigen Wert schätzt, der die Skala möglichst klein hält. Das impliziert dass unsere Beobachtung(en) am (in diesem Fall oberen) Rand der Skala liegen. Das ist aber nicht plausibel – viel plausibler ist es, dass sie sich um den Erwartungswert befinden.

23.2 Apriori Verteilungen über diskrete offene Skalen

Wir haben über *apriori*-Verteilungen gesprochen, und darüber, dass für endliche Räume die uniforme Verteilung die maximale Entropie hat. Nun nehmen wir aber das obige Beispiel: wir haben eine abzählbar unendliche Menge von möglichen Parametern: es gibt

$$P(\text{es gibt } n \text{ Taxis}) : n \in \mathbb{N}$$

Wenn wir die Wahrscheinlichkeitsmasse uniform darüber verteilen, dann bekommt jedes n aber eine *apriori* Wahrscheinlichkeit von 0, wir haben also keine diskrete Verteilung mehr! Wenn wir also diskrete Verteilungen über abzählbar unendliche Mengen suchen, steht die uniforme Verteilung nicht mehr zur Verfügung. Was ist also die **neutralste Verteilung** über $\mathbb{N}$? Hier gibt es keine eindeutige Antwort, sondern eine ganze Familie von Funktionen.

Wir nennen Funktionen $P : \mathbb{M} \to [0, 1]$, die die Form haben

$$P_r(n) = (1 - r)r^{n-1} \tag{290}$$

mit $r \in [0, 1)$, **geometrische Verteilungen**. Jede geometrische Verteilung P_r hat den Erwartungswert $r/(1 - r)$, denn es gilt unabhängig von r dass

$$\sum_{i=1}^{\infty}(1 - r)r^{n-1} = r/(1 - r) \tag{291}$$

Ein Spezialfall hiervon ist die Funktion

$$P_{0.5}(n) = \frac{1}{2^n} \tag{292}$$

die wir bereit kennengelernt haben, und die nach der letzten Gleichung also den Erwartungswert 1 besitzt. Die Wichtigkeit der geometrischen Verteilungen wird durch folgendes Ergebnis belegt:

Lemma 16 *Für jeden Wert $r/(1 - r)$ ist die die geometrische Verteilung P_r die eindeutige Wahrscheinlichkeitsverteilung über $\mathbb{N}$ mit 1. diesem Erwartungswert und 2. der maximalen Entropie.*

Wir haben also eine Familie von Funktionen, die für ihren jeweiligen Erwartungswert die maximale Entropie haben. Man beachte auch folgendes: die Funktion

$$f(x) = \frac{x}{1 - x} \tag{293}$$

ist stetig und nimmt für $x \in [0,1)$ jeden Wert in $\mathbb{R}$ an; es gibt also für jeden Erwartungswert $x \in \mathbb{R}$ eine Verteilung P_r mit genau diesem Erwartungswert.

Es gibt jedoch noch das **Problem der Permutation**: für die Entropie spielt die Natur eines Ereignisses keine Rolle, sondern einzig dessen Wahrscheinlichkeit. Dementsprechend ändert sich die Entropie von P nicht unter Permutationen. Eine Permutation π ist eine Abbildung

$$\pi : \mathbb{N} \to \mathbb{N},$$

so dass

$$\pi[\mathbb{N}] = \mathbb{N} = \pi^{-1}[\mathbb{N}],$$

also eine Bijektion, für die außerdem jede Zahl ein Urbild hat ($f(n) = n + 1$ ist z.B. eine Bijektion, aber keine Permutation – die 1 hat kein Urbild). Es ist nun leicht zu sehen, dass für jede Permutation π gilt:

(294) $H(P_r) = H(P_r \circ \pi)$

Denn Addition ist kommutativ und kümmert sich also nicht um die Reihenfolge der Elemente. Allerdings ist

(295) $\mathcal{E}(P_r) < \mathcal{E}(P_r \circ \pi)$

(es folgt aus der Natur der geometrischen Verteilung dass jede Permutation den Erwartungswert nach oben schiebt). Weiterhin gibt es eine geometrische Verteilung $P_{r'}$ so dass

(296) $\mathcal{E}(P_{r'}) = \mathcal{E}(P_r \circ \pi)$

wobei dann natürlich $r' > r$ (je größer r in der geometrischen Verteilung, desto weiter nach rechts verschiebt sich der Erwartungswert. Daraus folgt natürlich wiederum, dass

(297) $H(P_{r'}) > H(P_r \circ \pi) = H(P_r)$,

also: je größer r, desto größer die Entropie. Da aber $r \in [0,1)$ liegt, gibt keinen Maximalwert.

23.3 Schätzen von kontinuierlichen Skalenparametern

(Nach Jaynes, Probability Theory, p190ff.) Nehmen wir einmal an, wir möchten schätzen, wie weit eine reellwertige Skala reicht, wobei wir eine Menge von Beobachtungen $D = \{x_1, ..., x_i\} \subseteq \mathbb{R}$ haben. Wir suchen $\alpha \in \mathbb{R}$, die Obergrenze der Skala, und unser *apriori* Wissen sagt uns, dass

$$(298)\quad P(x|\alpha, I) = \begin{cases} \frac{1}{\alpha}, \text{ if } 0 \leq x \leq \alpha \\ 0 \text{ andernfalls.} \end{cases}$$

Es ist leicht zu sehen, dass wir hier im Prinzip das Taxi-Problem aufgreifen, nur eben mit reellwertigen Parametern und der entsprechenden kontinuierlichen Wahrscheinlichkeitsfunktion. Eine Verteilung wie in (298) nennt man auch **rechteckig**; wem der Grund unklar ist, der zeichne sich den Graphen. Die Wahrscheinlichkeit unserer Daten, gegeben einen Parameter α und $0 \leq x_1, \ldots, x_i \leq \alpha$ lässt sich leicht berechnen als

$$(299)\quad P(D|\alpha, I) = \prod_{n=1}^{i} P(x_n|\alpha, I) = \frac{1}{\alpha^i}$$

Wenn wir die aposteriori-Verteilung möchten, brauchen wir einfach den Satz von Bayes der uns sagt:

$$(300)\quad P(\alpha|D, I) = P(D|\alpha, I)\frac{P(\alpha|I)}{P(D|I)}$$

$P(D|I)$ ist natürlich erstmal uninteressant (aber später wichtig); was jedoch wichtig ist, ist die apriori Wahrscheinlichkeit $P(\alpha|I)$. Wir legen einmal folgendes apriori fest:

$$(301)\quad P(\alpha|I) = \begin{cases} \alpha_1 - \alpha_0, \text{ falls } \alpha_0 \leq \alpha \leq \alpha_1 \\ 0 \text{ andernfalls.} \end{cases}$$

für feststehende α_0, α_1. Das setzt natürlich voraus, dass $x_1, ..., x_i \leq \alpha_1$, ansonsten haben wir eine logische Inkonsistenz.

23.4 Jeffreys Apriori-Verteilung

Harold Jeffreys hat als erster bemerkt, dass eine ebene Verteilung für einen kontinuierlichen, offenen Parameter nicht wirklich optimal ist um völlige Ignoranz zu modellieren. Stattdessen sollte die Verteilung uniform über den Logarithmus des Parameters sein, d.h. es gibt eine konstante c so dass gilt:

(302) $P(log(\alpha)|I)c \propto \frac{1}{\alpha}$

(was heißt das erste, wie kommt eins zum anderen???)

24 Entropie, Kodierung, und Anwendungen

24.1 Definition

Entropie basiert auf dem Konzept der Information: Information ist invers zur Wahrscheinlichkeit, nach dem Prinzip:

> Je geringer die Wahrscheinlichkeit eines Ereignisses, desto mehr Information bekommen wir, wenn wir erfahren dass es stattfindet.

Man nimmt hierfür den inversen Logarithmus:

(303) $I(\omega) = -logP(\omega)$

Das hat zwei Spezialfälle:

1. falls $P(\omega) = 1$, dann ist $I(\omega) = 0$ (wie zu erwarten)
2. falls $P(\omega) = 0$, dann ist $I(\omega) = \infty$. Das kann manchmal für Probleme sorgen (in Termen), wird aber normalerweise rausgefiltert.

Die Entropie erhält man, wenn man Information und Wahrscheinlichkeit multipliziert. Das Konzept der Entropie formalisiert die *Unsicherheit* in einem System. Die Definition der Entropie ist wie folgt: wir haben eine Wahrscheinlichkeitsfunktion P und ein Ereignis ω. Die Entropy von ω (nach P), geschrieben $H_P(\omega)$, ist

(304) $H_P(\omega) := P(\omega) \cdot -log(P(\omega)) = P(\omega) \cdot I(\omega)$

Die Entropie eines einzelnen Ereignisses ist normalerweise weniger interessant als die Entropie einer ganzen Verteilung P (über einen diskreten Raum Ω, geschrieben $H(P)$:

(305) $H(P) := -\sum_{\omega \in \Omega} P(\omega) log(P(\omega))$

Es ist leicht zu sehen dass das einfach die Summe der Entropie der Ergebnisse ist; wir haben nur das minus ausgeklammert (das funktioniert nur im diskreten Fall).

Entropie als erwartete Information Information lässt sich als Zufallsvariable auffassen. Die Entropie (einer Verteilung) ist der Erwartungswert dieser Informationsvariable. Die Entropie einer Verteilung ist also die erwartete Informativität (eines Ergebnisses).

Grundregel zu maximaler Entropie im Endlichen Als allgemeine Regel lässt sich sagen: in einem Raum mit n Ergebnissen ist die Entropie *maximal*, wenn alle Ereignisse die gleiche Wahrscheinlichkeit $1/n$ haben; sie wird minimal (geht gegen 0), falls es ein Ereignis gibt dessen Wahrscheinlichkeit gegen 1 geht.

Maximale Entropie im Unendlichen Das ist ein kompliziertes Kapitel; hier nehmen die Normalverteilung und die geometrische Verteilung eine prominente Stellung ein. Es gibt allerdings nicht *die* Verteilung über $\mathbb{N}, \mathbb{R}$ mit der meisten Entropie (die uniforme kann es nicht sein).

Das deckt sich mit unseren Intuitionen: je größer die Entropie, desto weniger Sicherheit haben wir, wie das Ergebnis sein wird. Z.B.: nehmen wir das Beispiel eines fairen Würfels; wir können die Entropie des zugehörigen Wahrscheinlichkeitsraumes wie folgt ausrechnen:

```
> x = 0 : 5
> for(i in 1 : 6){
+x[i] < −1/6 ∗ log2(1/6)}
> −sum(x)
[1]2.584963
```

(Wir verzichten darauf, die Entropie ins positive zu wenden). Wenn wir hingegen annehmen, 5 Seiten haben die Wahrscheinlichkeiten 1/10 und die 6 hat eine Wahrscheinlichkeit 1/2, dann bekommen wir:

```
> x = 0 : 5
> for(i in 1 : 5){
+x[i] < −1/10 ∗ log2(1/10)}
> x[6] = 1/2 ∗ log2(1/2)
> −sum(x)
```

```
[1]2.160964
```

Andersrum gesagt: je größer die Entropie (einer Wahrscheinlichkeitsverteilung für ein Zufallsexperiment), desto größer der Informationsgewinn, der darin besteht das Ergebnis zu erfahren. Wichtig ist: Entropie ist immer unabhängig von den einzelnen Ergebnissen, es spielt also keine Rolle ob die 1 oder die 6 eine erhöhte Wahrscheinlichkeit hat. Alles was zählt ist eben die Ungewissheit; wir können das mit einem weiteren Versuch nachrechnen:

```
> x = 0 : 5
> for(iin1 : 3){
+x[i] < −1/10 ∗ log2(1/10)}
> x[4] = (1/20) ∗ log2(1/20)
> x[5] = (3/20) ∗ log2(3/20)
> x[6] = 1/2 ∗ log2(1/2)
> −sum(x)
[1]2.12322
```

Die Entropie ist also weiter gesunken, denn wir haben die Wahrscheinlichkeiten weiter ungleich aufgeteilt zwischen 2 Ergebnissen: während also die Entropie für 1,2,3,6 gleich geblieben ist, ist sie für 4,5 lokal gesunken, also ist sie auch global gesunken. Man kann auch umgekehrt sagen: da die uniforme Wahrscheinlichkeitsverteilung für uns den *Mangel* an relevanter Information bezeichnet, gibt es die Korrelation

$$\text{maximale Entropie} \approx \text{maximale Unwissenheit}$$

Darauf basiert eine wichtige Methode der Wahrscheinlichkeitstheorie, die sog. Maximum Entropie Schätzung. Die basiert auf dem Grundsatz:

> In Ermangelung sicherer Information ist es besser, möglichst wenig Sicherheit anzunehmen, als falsche Sicherheit die es nicht gibt (es ist besser zu wissen dass man etwas nicht weiß)

Das bedeutet effektiv: wir sollten die Wahrscheinlichkeitsverteilung annehmen, die

1. mit unserem Wissen kompatibel ist,
2. ansonsten aber die Entropie maximiert.

Man definiert die Entropie auch oft für Zufallsvariablen:

$$H(X) := -\sum_{x \in X} P(X = x) log(P(X = x)) \quad (306)$$

24.2 Kodierungstheorie und Entropie

Der Zusammenhang von Entropie und Kodierung (z.B. im Alphabet $\{0,1\}$) beruht darauf, dass wir ein Symbol, was häufig ist ($\cong$ hohe Wahrscheinlichkeit) möglichst kurz kodieren, während relativ seltene Symbol eher lange Kodes bekommen. Auf diese Art sind unsere Kodes von Texten im Normalfall möglichst kurz.

Seien Σ, T zwei Alphabete (z.B. $T = \{0,1\}$). Ein Kode (von Σ in T) ist Paar

$$(\phi, X),$$

wobei

$$X \subseteq T^*, \text{ und } \phi : \Sigma \to X$$

eine Bijektion ist, so dass die homomorphe Erweiterung von

$$\phi : \Sigma^* \to X^*$$

weiterhin eine Bijektion ist.

Ein Kode ist **präfixfrei**, falls es kein $x, y \in X$ gibt so dass

$$xz = y, \text{ für beliebiges } z \in T^+.$$

Beispiel 1 (z.B.: $\phi(a) = 0110, \phi(b) = 001, \phi(c) = 011, \phi(d) = 1001$) ist nicht präfixfrei. Dekodiere:

$$011001100110011001100110$$

Was man hieran sieht: die Dekodierung muss verschoben werden, bis *ganz ans Ende*: erst wenn wir das Ende haben, können wir mit der Dekodierung anfangen. Sowas ist extrem aufwändig bis unmöglich, normalerweise wollen wir das nicht.

Beispiel 2 Der einfachst Kode ist immer der sog. Block-Kode: setze $\phi : \Sigma \to T^*$ so dass f.a. $a \in \Sigma$,

(307) $|\phi(a)| = n$

(also feste Länge). So funktioniert ASCII. Wenn ϕ injektiv ist, dann ist es auch seine homomorphe Erweiterung, denn wir dekodieren Block für Block.

Das Problem bei solchen Block-Kodes ist: wenn es einen großen Unterschied gibt in der Wahrscheinlichkeit der Buchstaben, sind sie ziemlich ineffizient.

Wir sind in der Informatik meist in Kodes über $\{0,1\}^*$ interessiert, und wir möchten üblicherweise Alphabete kodieren, die mehr als zwei Buchstaben enthalten. Es stellt sich die Frage, wie man das am besten macht. Intuitiv ist unser Ziel: wir möchten, dass jede Kodierung eines Textes möglichst kurz wird. Das ist natürlich trivial, sofern wir nur die Buchstaben Σ haben. Aber nehmen wir an, wir haben eine Wahrscheinlichkeitsverteilung über Σ, und weiterhin, dass die Wahrscheinlichkeiten der Worte unabhänging voneinander sind. Das bedeutet:

- wenn ein Buchstabe sehr wahrscheinlich ist, dann wollen wir ihn kürzer kodieren,
- wenn er unwahrscheinlich ist, dann länger.

Sei $w \in \Sigma^*$. Wir bauen uns eine Zufallsvariable X, so dass

$$X(a) = |\phi(a)| \tag{308}$$

(die Länge des Wortes). Was wir möchten ist: wir möchten den Erwartungswert von X möglichst klein machen. Wir haben

$$\mathcal{E}(X) = \sum_{a \in \Sigma} |\phi(a)| \cdot P(a) \tag{309}$$

Jeder Buchstabe im Ausgangsalphabet Σ hat Länge 1; er wird – nach Erwartungswert – in der Kodierung im Schnitt mit $\mathcal{E}(X)$ Symbolen ersetzt. Deswegen nennen wir die *Inversion*

$$\mathcal{E}(X)^{-1} = \frac{1}{\mathcal{E}(X)} \tag{310}$$

den **Kompressionsfaktor** der Kodierung. Ein zentrales Ergebnis ist nun:

Lemma 17 *$\mathcal{E}(X) \geq H(P)$; die erwartete Wortlänge unter einer beliebigen binären Kodierung ist immer größer als die Entropie der zugrundeliegenden Verteilung. (zur Basis 2!!)*

Das bedeutet wir müssen jedes Symbol im Schnitt mit mindestens $H(P)$ zeichen kodieren. Deswegen Entropie immer zur Basis 2; sonst geht diese Korrelation verloren.

Wir möchten im Allgemeinen den Erwartungswert minimieren, d.h. den Kompressionsfaktor maximieren. Es gibt einen einfachen Algorithmus, den sogenannten **Huffman code**, der folgendes liefert:

- Eingabe: ein beliebiges Alphabet Σ mit einer zugehörigen Wahrscheinlichkeitsfunktion $P : \Sigma \to [0, 1]$
- Ausgabe: eine Kodierung von Σ in $\{0, 1\}^*$ in einem Präfix-freien Kode mit maximalen Kompressionsfaktor (es gibt aber meist mehrere solcher Kodierungen).

Wir zeigen den Algorithmus, der eigentlich sehr einfach ist, mit einem Beispiel.

Ein Beispiel Nehmen wir an, $\Sigma = \{a, b, c, d\}$, mit folgenden Wahrscheinlichkeiten (bzw. Häufigkeiten):

- $P(a) = 0.1$
- $P(b) = 0.2$
- $P(c) = 0.3$
- $P(d) = 0.4$

Wir fangen damit an, das Buchstabenpaar zu nehmen, das am seltensten vorkommt. Das ist natürlich

$$\{a, b\} \text{ mit } P(\{a, b\}) = 0.3.$$

Wir ersetzen nun

$$\{a, b\} \mapsto \{x_1\},$$

so dass unser neues Alphabet ist

$$\{x_1, c, d\}, \text{ wobei } P(x_1) = 0.3.$$

Nun machen wir ebenso weiter: im neuen Alphabet ist das Buchstabenpaar mit der geringsten Wahrscheinlichkeit $\{x_1, c\}$, also ersetzen wir

$$\{x_1, c\} \mapsto \{x_2\}$$

mit dem resultierenden Alphabet

$$\{x_2, d\}, \text{ wobei } P(x_2) = P(\{x_1, c\}) = 0.6.$$

Nun machen wir den Schritt ein letztes Mal: das resultierende Alphabet ist

$$\{x_3\} \text{ mit } P(x_3) = 1.$$

Nun "entpacken" wir das ganze wieder.

- Wir nehmen an, x_3 wird kodiert durch das leere Wort ϵ.

ϵ steht dann aber eigentlich für 2 Buchstaben: x_2 und d. Das erste ist wahrscheinlicher.

- Also kodieren wir x_2, indem wir eine 0 an unser Kodewort hängen, d mit einer 1.

Nun steht x_2 (bzw. 0) wiederum für zwei Buchstaben: x_1, c.

- Wir setzen $x_1 = 00, c = 01$ (in diesem Fall ist es egal, die Wahrscheinlichkeiten sind gleich).

Nun dasselbe mit x_1 (bzw. 00);

- in diesem Fall bekommen wir 000 für b, 001 für a.

Wir bekommen also:

- $\phi(a) = 001$
- $\phi(b) = 000$
- $\phi(c) = 01$
- $\phi(d) = 1$

Wir nehmen nun X wie oben, und bekommen:

(311) $\mathcal{E}(X_\phi) = 0.1 \cdot 3 + 0.2 \cdot 3 + 0.3 \cdot 2 + 0.4 \cdot 1 = 1.9$

Der Kompressionsfaktor ist also $\frac{1}{1.9}$. Natürlich kommt dasselbe raus, wenn wir im Kode einfach 0 und 1 vertauschen. Wenn wir das vergleichen mit dem folgenden Block-Kode

- $\chi(a) = 00$
- $\chi(b) = 01$
- $\chi(c) = 10$
- $\chi(d) = 11$

(der auch Präfix-frei ist), dann bekommen wir

(312) $\mathcal{E}(X_\chi) = 0.1 \cdot 2 + 0.2 \cdot 2 + 0.3 \cdot 2 + 0.4 \cdot 2 = (0.1 + 0.2 + 0.3 + 0.4)2 = 2$

Der Kompressionsfaktor beträgt also nur $\frac{1}{2}$.

Wie ist die Entropie für P?

(313)
$H(P) = -(0.1log_2(0.1)+0.2log_2(0.2)+0.3log_2(0.3)+0.4log_2(0.4)) = 1.846439$

Nehmen wir an, dagegen an dass

$$P'(a) = \ldots = P'(d) = 0.25$$

ändert sich die Lage: in diesem Fall ist natürlich χ die optimale Kodierung. Die Entropie ändert sich wie folgt:

(314) $H(P') = -(4 \cdot (0.25log_2(0.25)) = 2$

Die Entropie ist größer, daher wird auch die Kompressionsrate schlechter sein. Am Ende gilt:

Entropie und Kodierung Sei P eine Wahrscheinlichkeitsverteilung über Σ, ϕ ein (beliebiger) Kode über $\{0, 1\}$. Dann kann der Kompressionsfaktor von ϕ niemals grösser sein als $\frac{1}{H(P)}$, berechnet zur Basis 2. Noch einfacher:

(315) $H(P) \leq \mathcal{E}(X_\phi)$

Die Entropie ist also die untere Schranke dafür, wie kompakt wir kodieren können. Für den Huffmann Kodierung ϕ_H gilt übrigens auch:

(316) $H(P) \leq \mathcal{E}(X_{\phi_H}) \leq H(P) + 1$

24.3 Bedingte Entropie

Die bedingte Entropie von zwei Variablen (über demselben Wahrscheinlichkeitsraum) ist wie folgt definiert (hier bedeutet $y \in Y$ soviel wie: y ist ein Wert, den Y annehmen kann):

$$(317)\quad H(X|Y) = -\sum_{y \in Y} P(Y = y) H(X|Y = y)$$

Wenn wir diese Definition auflösen, bekommen wir:

$$(318)\quad H(X|Y) = -\sum_{x \in X, y \in Y} P(X^{-1}(x) \cap Y^{-1}(y)) log\left(\frac{P(X^{-1}(x) \cap Y^{-1}(y))}{P(Y^{-1}(y))}\right)$$

Die bedingte Entropie ist also ein Maß dafür, wie stark die Werte einer Zufallsvariable Y die Werte einer Zufallsvariable X festlegen. Wenn der Wert von X durch den Wert von Y – egal wie er ist – immer festgelegt ist, dann ist

$$(319)\quad H(X|Y) = 0$$

insbesondere also:

$$(320)\quad H(X|X) = 0$$

Umgekehrt, falls der Wert von Y keinerlei Einfluss hat auf die Wahrscheinlichkeitsverteilung des Wertes von X, dann haben wir

$$(321)\quad H(X|Y) = H(X)$$

Es ist klar dass das hier nur für diskrete Wahrscheinlichkeitsräume funktionieren kann; in kontinuierlichen Räumen funktionieren diese Dinge etwas anders.

Es gibt auch eine Kettenregel für bedingte Entropie:

$$(322)\quad H(X|Y) = H(\langle X, Y \rangle) - H(Y)$$

wobei $\langle X, Y \rangle$ eine neue Variable ist, mit

$$(323)\quad P(\langle X, Y \rangle = (x, y)) = P(X = x, Y = y) = P(X^{-1}(x) \cap Y^{-1}(y))$$

Wir nehmen also die Entropie der **Verbundverteilung**, und ziehen die Entropie von $H(Y)$ ab.

(das entspricht der Definition

$$P(x|y) = \frac{P(x,y)}{P(y)} \tag{324}$$

)

Damit bekommen wir ausserdem folgende Rechenregeln:

$$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y) \tag{325}$$

Daraus lässt sich leicht folgern:

$$H(X) \leq H(X,Y) \tag{326}$$

$$H(X|Y) \leq H(X) \tag{327}$$

Das ist die bekannte Produktregel in neuem Gewand – im Logarithmus wird Multiplikation zu Addition! Damit bekommen wir auch die **wechselseitige Information** zweier Variablen:

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{328}$$

Das ist ein Mass dafür, wie informativ der Wert einer Variable für die Verteilung der anderen Variable ist. Dieses Mass ist symmetrisch – sowie Information ein symmetrisches Konzept ist!

24.4 Bedingte Entropie – ein Rechenbeispiel mit Anwendung

(Das Beispiel wird ausführlicher besprochen im Kapitel Entscheidungsfunktionen) Um mit dem Konzept vertraut zu werden, erstmal folgendes Beispiel (aus Russel & Norvig): es geht um die Entscheidung, ob wir in einem Restaurant warten, bis wir einen Tisch zugewiesen bekommen, oder weitergehen; also eine binäre Entscheidung. An dieser Stelle sucher erstmal (vorläufig) dasjenige Merkmal, welches am *informativsten* ist für unsere Entscheidung, ob wir warten oder nicht, mithilfe unseres Datensatzes. Informativität messen wir hier mit bedingter Entropie. Unsere Merkmale sind:

1. Alternativen: gibt es passende Alternativen in der Nähe?
2. Theke: können wir uns an die Theke setzen und schonmal ein Bier trinken?
3. Fr/Sa: ist es Freitag oder Samstag?
4. Betrieb: wie viel Betrieb ist im Lokal? (Werte: leer, einige Leute, voll)
5. Regen: regnet es draußen?
6. Reservierung: haben wir reserviert?
7. Typ: was für eine Art Restaurant haben wir (französisch, italienisch, deutsch)
8. Geschätzte Wartezeit (von uns geschätzt): 0-10,10-30,30-60,>60

Unser Datensatz sieht wie folgt aus:

Was wir nun berechnen wollen ist $H(warten|X)$, für $X \in \{\text{Alt,Theke,Fr/Sa,...}\}$, und was uns natürlich interessiert ist

$$\underset{X}{argmin} H(warten|X) \tag{329}$$

Denn dasjenige X, welches die bedingt Entropie minimiert, reduziert die verbliebene Unsicherheit am meisten, ist also der beste Prädiktor!

Um das auszurechnen, nutzt man am besten Formel 318, hier wiederholt:

Bsp.	Alt	Theke	Fr/Sa	Bet	Reg	Res	Typ	Wart	Warten?
d1	1	0	0	halb	0	1	fr	0-10	1
d2	1	0	0	voll	0	0	it	30-60	0
d3	0	1	0	halb	0	0	de	0-10	1
d4	1	0	1	voll	1	0	it	10-30	1
d5	1	0	1	voll	0	1	fr	>60	0
...									

Table 1: Ein Ausschnitt aus unserem Datensatz

$$(330)\quad H(X|Y) = \sum_{x\in X, y\in Y} P(X^{-1}(x)\cap Y^{-1}(y)) log\left(\frac{P(X^{-1}(x)\cap Y^{-1}(y))}{P(Y^{-1}(y))}\right)$$

Hier entspricht X dem Zielmerkmal, Y einem Prädiktor. Man addiert also f.a. möglichen Werte, die jeweils X, Y annehmen können, den entsprechenden Wert (Wahrscheinlichkeiten sind hier relative Häufigkeiten der Werte im Datensatz).

Übung Wir berechnen

$$(331)\quad H(warten|Bet)$$

Da beide Merkmale jeweils vier Werte annehmen können, bekommen wir eine Summe

$$
\begin{aligned}
H(warten|Bet) &= -\sum_{x\in X, y\in Y} P(X^{-1}(x)\cap Y^{-1}(y)) log\left(\frac{P(X^{-1}(x)\cap Y^{-1}(y))}{P(Y^{-1}(y))}\right) \\
&= -(P(X^{-1}(0)\cap Y^{-1}(h)) log\left(\frac{P(X^{-1}(0)\cap Y^{-1}(h))}{P(Y^{-1}(h))}\right) \\
&\quad +P(X^{-1}(0)\cap Y^{-1}(v)) log\left(\frac{P(X^{-1}(0)\cap Y^{-1}(v))}{P(Y^{-1}(v))}\right) \\
&\quad +P(X^{-1}(1)\cap Y^{-1}(h)) log\left(\frac{P(X^{-1}(1)\cap Y^{-1}(h))}{P(Y^{-1}(h))}\right) \\
&\quad +P(X^{-1}(1)\cap Y^{-1}(v)) log\left(\frac{P(X^{v}{-}1(1)\cap Y^{-1}(v))}{P(Y^{-1}(v))}\right)) \\
&= -(0+2/5\cdot log\frac{2/5}{3/5}+0+1/5\cdot log\frac{1/5}{3/5}) \\
&\approx 0.55
\end{aligned}
$$

(alle Logarithmen zur Basis 2)

Übung

1. Berechnen Sie die unbedingte Entropie $H(warten)$ (einfach; Wahrscheinlichkeiten sind relative Häufigkeiten)

2. Berechnen Sie die bedingte Entropie $H(warten|Res)$.

3. Schreiben Sie ein Programm, welches als Eingabe nimmt: zwei Vektoren *vec1*, *vec2* gleicher Länge mit Werten in $\{0, 1\}$, und als Ausgabe liefert: $H(vec1|vec2)$ (die Wahrscheinlichkeiten von $0, 1$ sind natürlich die relativen Häufigkeiten im Vektor, genau wie im Datensatz)

Teil 3 besteht also darin, die Arbeit aus Teil 2 den Computer machen zu lassen!

24.5 Kullback-Leibler-Divergenz

Die KL-Divergenz ist eine andere Art zu messen, wie ähnlich sich zwei Wahrscheinlichkeitsverteilungen P und Q sind. Die Definition ist wie folgt:

$$D_{KL}(P\|Q) = \sum_{\omega\in\Omega} P(\omega) log \frac{P(\omega)}{Q(\omega)} \tag{332}$$

An dieser Definition kann man ablesen:

1. $D_{KL}(P\|Q) = 0$ gdw. für alle $\omega \in \Omega$ gilt: $P(\omega) = Q(\omega)$; denn $log(1) = 0$.
2. In allen anderen Fällen ist $D_{KL}(P\|Q) > 0$ (das ist nicht wirklich leicht zu sehen).
3. $D_{KL}(P\|Q) \neq D_{KL}(Q\|P)$, d.h. wir haben ein asymmetrisches Maß.
4. Man kann es jedoch symmetrisch machen auf folgende Art und Weise:

$$D_2(P\|Q) = D_{KL}(P\|Q) + D_{KL}(Q\|P) = D_2(Q\|P)$$

Sie gibt uns also ein Maß dafür, wie weit Q von P **entfernt** ist. Dadurch unterscheidet sie sich konzeptuell von $H(X|Y)$, dass bestimmt wie stark X von Y *determiniert* wird.

Man bezeichnet $D_{KL}(P\|Q)$ auch als den **Informationsgewinn**, den man mit P gegenüber Q erzielt. Wenn wir z.B. das obige Kodierungsbeispiel fortführen, dann sagt uns $D_{KL}(P\|Q)$, wieviel Platz wir (im Durchschnitt) verschwenden, wenn wir eine Kodierung auf Q basieren, während die zugrundeliegende Wahrscheinlichkeitsverteilung P ist.

Dementsprechen nutzt man $D_{KL}(P\|Q)$ oft im Kontext, wo P die tatsächliche Verteilung ist (was immer das konkret heißt), Q unser Modell, dass wir geschätzt haben, das wir verbessern wollen.

24.6 Kreuzentropie

Ein ähnliches Konzept ist das der sog. Kreuzentropie:

$$H(X, P, Q) = H(X_P) + D(P\|Q) = -\sum_{x \in \Omega} P(X = x) log Q(X = x) \tag{333}$$

(nur bei diskreten Verteilungen) wobei P die Verteilung unseres Systems ist, Q die "wahre Verteilung" (ergibt sich aus den "korrekten Vorhersagen"). Das resultiert also in einer differenzierbaren Funktion die heuristisch und analytisch optimiert werden kann, und daher im Machine Learning eine große Rolle spielt.

Das funktioniert wie folgt:

- Wir trainieren einen Klassifikator $f : X \to C$, wobei C eine endliche Menge von Klassen ist.
- $f(x)$ ist aber normalerweise eine Wahrscheinlichkeitsverteilung über C (das reduziert man dann, indem man die wahrscheinlichste Klasse wählt
- Die Trainingsdaten sagen: $P(c_i|x) = 1$ – also eine Verteilung, die einer Klasse die 1 zuweist.
- Kosten sind die Kreuzentropie $H(X|P, f(X))$

Übung

Wir nehmen eine diskrete Wahrscheinlichkeitsverteilung P über das kartesische Produkt $\Omega = \{a, b\} \times \{1, 2, 3\}$, mit
$P(a, 1) = 0.4$
$P(a, 2) = 0.1$
$P(a, 3) = 0.05$
$P(b, 1) = 0.05$
$P(b, 2) = 0.1$
$P(b, 3) = 0.3$

(Z.B.: 0,1,2 sind die Ampelphasen grün, gelb, rot, a ist gehen, b ist stehen, Wir definieren 2 Zufallsvariablen $X_1(x, y) = x$, $X_2(x, y) = y$; so ist z.B. $P(X_2 = 1) = P(\{(a, 1), (b, 1)\})$ etc.

1. Berechnen Sie $H(X_1)$, $H(X_2)$.

2. Berechnen Sie $H(X_1|X_2)$. Was bedeutet das Ergebnis?

RX8 Übung

Berechnen Sie die Entropie in der Spalte verbs$ RealizationOfRecipient. Weiterhin berechnen Sie die bedingte Entropie (immer noch von RealizationOfRecipient), gegeben jeweils eine andere Spalte.

Was sagt uns das Ergebnis?

25 Maximum Entropie Methoden

25.1 Definition

Maximum Entropie (ME) Methoden sind sehr allgemein und mächtig, und wir werden uns nur einen besonderen Fall anschauen. In der Praxis haben wir manchmal (oft) den Fall, dass wir eine Wahrscheinlichkeitsfunktion schätzen möchten, aber die Verteilung mehr Parameter hat, als durch die Daten vorgegeben werden. Das kann verschiedene Gründe haben:

1. Unser Weltwissen leitet uns zu der Annahme, dass es relevante Parameter gibt, die wir nicht direkt beobachten können (z.B. syntaktische Kategorien, Wortarten, Bedeutungen in der Sprachverarbeitung).

2. Unser Weltwissen leitet uns zu der Annahme, dass Parameter, die auf den ersten Blick relevant erscheinen, eigentlich irrelevant sind und nicht zur Schätzung hinzugezogen werden sollten (z.B. wenn wir annehmen wir haben die Markov Eigenschaft erster Ordnung).

Theoretisch sieht das normalerweise wie folgt aus: wir haben eine Reihe von Zufallsvariablen $X_1, ..., X_i$, die jeweils eine bedingte Verteilung haben für Zufallsvariablen $Y_1, ..., Y_i$ (sie hängen also von Y ab). Wir suchen also Verteilungen

$$P(X_1|Y_1) \tag{334}$$

$$P(X_2|Y_2) \tag{335}$$

$$\vdots \tag{336}$$

$$P(X_i|Y_i) \tag{337}$$

$Y_1, ..., Y_i$ sind also **Prädiktoren** für $X_1, ..., X_i$; sie bestimmen deren Verteilung. Ein typisches Beispiel wäre folgendes:

- Die X_n sind syntaktische Kategorien;

- die Y_n sind tatsächliche Worte.

Das Problem das sich dann ergibt ist bekannt als POS-tagging.

Das Problem ist aber folgendes: die dünne Datenlage erlaubt aber keine vollständige Schätzung nach ML (oder anders); alles was wir haben ist der Erwartungswert (und vielleicht Varianz, Standardabweichung etc.). Das

äußert sich dann in einer Reihe gewisser Bedingungen, die unsere Funktionen erfüllen müssen, ohne dass sie dadurch vollständig determiniert wären; wir haben also eine Reihe von Beschränkungen der Form

$$\sum_{x \in X_1} \hat{P}(X_1 = x | Y_1 = y) \cdot x_1 = \alpha_1$$

(338) $$\vdots$$

$$\sum_{x \in X_i} \hat{P}(X_i = x | Y_i = y) \cdot x_i = \alpha_i$$

Diese Gleichungen liefern uns Bedingungen, die $\hat{P}$ erfüllen muss. Wenn wir abstrakt von so einer Liste von Gleichungen ausgehen, dann gibt es keine Garantie, dass es tatsächlich ein $\hat{P}$ gibt, dass alle Gleichungen erfüllt. Da wir aber alle Gleichungen von *denselben Daten* schätzen, ist klar dass es mindestens eine Verteilung $\hat{P}$ gibt, die alle erfüllt (nämlich die volle ML-Schätzung für alle Parameter). Das Problem ist eher das umgekehrte: es gibt normalerweise viele, genauer gesagt unendlich viele Verteilungen, die diese Gleichungen erfüllen. Unsere Frage ist: welche sollen wir wählen? Und hier kommt der Ansatz der **Maximum Entropie (ME) Methode**.

> Wir wählen die Verteilung $\hat{P}$, die die obigen Gleichungen erfüllt *und* die maximale Entropie hat.

Intuitiv bedeutet das: da Entropie ein Maß für Information bzw. Unsicherheit ist, wir möchten dass unsere Verteilung alle relevante Information beinhaltet, aber keine weitere Information darüber hinaus.

Das ist leicht gesagt; es ist auch leicht in eine Formel geschrieben; sei $\mathcal{C}$ die Menge aller Verteilungen $\hat{P}$, die den obigen Gleichungen genüge tun. Was wir suchen ist

(339) $argmax_{\hat{P} \in \mathcal{C}} H(\hat{P})$

Das Problem ist: diese Formel zu finden. Und hier werden die Dinge spannend.

25.2 Ein einfaches Beispiel

Nehmen Sie an, sie reden mit einem Kollegen über seinen Arbeitsweg. Er sagt Ihnen:

- Mit den öffentlichen Verkehrsmitteln brauche ich durchschnittlich 45min;
- mit dem Auto durchschnittlich 40;
- mit dem Fahrrad durchschnittlich 35.

Natürlich hängt die Wahl des Verkehrsmittels von einer Menge Faktoren ab (das Wetter, der Verkehr etc.); außerdem haben wir nur die Erwartungswerte (also den Durchschnitt), keinesfalls die Wahrscheinlichkeitsverteilung: z.B. kann es sein dass beim Auto (wg. Verkehr) die Streuung sehr hoch ist, bei den öffentlichen eher gering. Davon wissen wir aber nichts!

Was wir aber wissen ist folgendes:

- Unser Kollege braucht durchschnittlich 41min für seinen Arbeitsweg.

Die Aufgabe ist nun: wir sollen $\hat{P}(\ddot{O}), \hat{P}(A), \hat{P}(F)$ schätzen. Alles was wir wissen sind die obigen Erwartungswerte, sowie

$$\hat{P}(\ddot{O}) + \hat{P}(A) + \hat{P}(F) = 1 \tag{340}$$

Um das ganze in eine einheitliche Form zu bringen, führen wir eine Zufallsvariable X ein, mit

$$\begin{aligned} P(X = 45) &= \hat{P}(\ddot{O}) \\ P(X = 40) &= \hat{P}(A) \\ P(X = 35) &= \hat{P}(F) \end{aligned} \tag{341}$$

Unsere Aufgabe ist nun, eine Verteilung P zu finden so dass gilt:

$$\sum_{x \in \{35,40,45\}} P(X = x) = 1 \qquad \mathcal{E}(X) = \sum_{x \in \{35,40,45\}} P(X = x) \cdot x = 41 \tag{342}$$

Intuitiv ist klar, dass das auf viele Arten und Weisen geschehen kann: es kann z.B. sein dass Ihr Kollege praktisch nie mit dem Auto, oft mit Öffis und noch öfter mit dem Fahrrad fährt; es kann aber genauso gut sein, dass er praktisch immer mit dem Auto und nur ausnahmsweise mit Öffis fährt. Wir

haben aber keinerlei Wissen über diese Sachen, und beide Annahmen sind gleichermaßen ungerechtfertigt gegeben unser Wissensstand.

Was wir daher anwenden ist dass ME-Prinzip, dass nichts weiter ist als eine Generalisierung des Prinzips der Indifferenz, das besagt:

> Falls wir kein relevantes Vorwissen haben, nehmen wir die uniforme Verteilung an.

Wir haben nun aber durchaus relevantes Vorwissen. Unser ME-Prinzip sagt daher (da Entropie ein Maß der Unsicherheit ist):

> Von allen Verteilungen, die mit unserem Vorwissen konform sind, nehmen wir immer diejenige an, die die Maximale Entropie hat.

Auf diese Weise sind wir sicher, dass wir nicht mehr in die Verteilung hineinstecken, als wir wirklich wissen; wir bleiben uns also unserer Unsicherheit bewußt. Das Problem ist: wie errechnen wir diese Verteilung? Zunächst das analytische Problem: wir suchen

(343)
$$\underset{P}{argmax}\, H(P) : \sum_{x\in\{35,40,45\}} P(X=x) = 1 \quad \& \quad \sum_{x\in\{35,40,45\}} P(X=x)\cdot x = 41$$

wobei gilt:

(344) $$H(P) = \sum_{x\in\{35,40,45\}} P(X=x) log_2(P(X=x))$$

Wie berechnen wir das? In unserem Fall geht das mit elementaren Methoden (die wir aus der Schule kennen); denn wir haben:

(345) $$35 = 35P(X=35) + 35P(X=40) + 35P(X=45)$$

und

(346) $$41 = 35P(X=35) + 40P(X=40) + 45P(X=45)$$

Nun subtrahieren wir die beiden Terme voneinander, damit bekommen wir:

(347) $$6 = 5P(X=40) + 10P(X=45)$$

also

$$P(X = 40) = \frac{6 - 10P(X = 45)}{5} = \frac{6}{5} - 2P(X = 45) \tag{348}$$

Wir können also $P(X = 40)$ loswerden; dasselbe gilt natürlich auch für $P(X = 35)$, indem wir (348) einsetzen in (340):

$$\begin{aligned} 1 &= P(X = 35) + \frac{6}{5} - 2P(X = 45) + P(X = 45) \\ P(X = 35) &= P(X = 45) - \frac{1}{5} \end{aligned} \tag{349}$$

Wir können also beide Wahrscheinlichkeiten ausdrücken können als Formeln mit der einzigen Variable $P(X = 45)$. Daraus wiederum folgt:

$$\sum_{x \in \{35,40,45\}} P(X = x) log_2(P(X = x)) \tag{350}$$

lässt sich schreiben also Funktion mit *einer* einzigen Variable, für die wir also nur den Maximalwert suchen müssen:

$$\begin{aligned} & \underset{P(X=45) \in [0,1]}{argmax} \\ & f_1(P(X = 45)) \cdot log_2(f_1(P(X = 45))) \\ + & f_2(P(X = 45)) \cdot log_2(f_2(P(X = 45))) \\ + & P(X = 45) \cdot log_2(P(X = 45)) \end{aligned} \tag{351}$$

wobei sich f_1 und f_2 jeweils aus (349) und (348) ergeben. Das lässt sich mit den gewöhnlichen analytischen Methoden (Nullstelle der Ableitung) leicht ausrechnen.

25.3 ME in der Computerlinguistik

In unserem Beispiel ließ sich der Wert gut berechnen, da er im Prinzip nur eine nicht-triviale Gleichung erfüllen musste. Gibt es eine ganze Reihe von Gleichungen, sind die Berechnungen kompliziert und man braucht fortgeschrittene Methoden (Lagrange-Multiplikatoren). Mittlerweile macht solche Sachen aber der Computer; der wichtigste Algorithmus heißt **generatilized iterative scaling**. Wichtig ist es zu verstehen worum es geht: die Schätzung verborgener Parameter, die durch unsere Daten nicht ausreichend determiniert sind.

In NLP Anwendungen kommt es oft dazu, dass wir gewisse Merkmale benutzen, uns aber über deren Bedeutung nicht ganz im Klaren sind. Wir schauen uns jetzt ein etwas komplizierteres Beispiel an. Unser Ziel ist es, Sequenzen zu modellieren, also nach wir vor (wie bei Markov-Ketten) eine Sprachmodell zu erstellen. Bei Markov Ketten war der Grundsatz die **Kettenregel**:

(352) $P(w_1...w_n) = P(w_1)P(w_2|w_1)....P(w_n|w_1...w_{n-1})$

Das war soz. das Grundsatzprogramm; das ganze wurde technisch machbar, indem wir sagten: wir haben für ein nicht zu großer k die Markov Eigenschaft $k$ter Ordnung, also:

(353) $P(w_n|w_1...w_{n-1}) = P(w_n|w_{n-k}...w_{n-1})$

Das Problem hierbei ist natürlich folgendes:

- Falls wir k zu klein wählen, wird das ganze sehr inadäquat;
- falls wir k zu groß wählen, haben wir zuwenig Daten für eine vernünftige Schätzung.

In jede Richtung treffen wir also Beschränkungen. ME Modelle können helfen, diese Probleme zu lösen. Anstatt dass wir uns uns auf die Kettenregel fokussieren und sagen, die Wahrscheinlichkeit jedes Wortes wird durch seine Vorgänger bestimmt, fokussieren wir uns auf **Merkmale**. Jeder Satz hat gewisse Merkmale, und die bestimmen ob er grammatisch ist. Das Modell ist also folgendes: für einen Satz $\overline{s}$ gilt

(354) $P(\overline{s}) = \frac{1}{Z} exp(\sum_i \lambda_i f_i(\overline{s}))$

Das ist ein sog. loglineares Modell; jedes f_i ist hier ein **Merkmal**. Wir können der Einfachheit annehmen, dass die Merkmale binär sind, also

(355) $f_i : \Sigma^* \to \{0, 1\}$

λ_i ist das sog. **Gewicht** des Merkmals: je stärker es gewichtet ist, desto wichtiger ist es für die Rechnung. $1/Z$ ist nur eine **Normalisierungskonstante**; die soll sicherstellen dass am Ende tatsächlich eine konsistente Wahrscheinlichkeitsverteilung herauskommt.

Das Problem ist: wie bekommen wir die Merkmale f_i und die gewichte λ_i? Hier brauchen wir ME. Der Prozess ist folgender:

- Für jedes f_i schauen wir, wie gut wir damit vorhersagen können, ob ein Satz grammatisch ist d.h.

  (356) $$\frac{(\text{Anzahl Sätze } \overline{s}) \cdot f_i(\overline{s})}{(\text{Anzahl Sätze } \overline{s})}$$

- Das liefert uns eine von Erwartungswerten, für jedes Merkmal einen.
- Nun suchen wir die Wahrscheinlichkeitsverteilung, die uns alle diese Erwartungswerte generiert.
- Das bedeutet: wir suchen die λ_i, die uns diese Verteilung liefern. Natürlich gilt: dadurch dass wir die Normalisierungskonstanten Z haben, gibt es viele solche λ.
- Wir suchen die Verteilung mit der maximalen Entropie!

26 Klassifikation mittels Entropie: Entscheidungsbäume

26.1 (Boolesche) Entscheidungsfunktionen

Klassifikation ist ein erstes Beispiel für induktive Inferenz. Was dabei induziert wird ist eine Funktion f, und zwar eine diskrete Funktion, d.h. eine Funktion die nur endlich viele verschiedene Eingaben nimmt und damit nur endliche viele Ausgaben liefert. Wir werden uns hauptsächlich einen Spezialfall der Klassifikation anschauen, nämlich die **Boolesche Klassifikation**. Boolesche Klassifikation ist deswegen speziell, weil wir eine Boolesche (Wahrheits-)Funktion lernen. Wir suche eine Funktion,

- die für eine Eingabe x entweder "ja" oder "nein" liefert;
- wir fassen "ja" als 1, nein als 0 auf;
- weiterhin basiert eine solche Funktion auf einer Menge von **Attributen**, die auch entweder den Wert 0 oder 1 haben (das werden wir lockern), also erfüllt sind oder nicht.

Wir haben also eine Funktion

(357) $f : \{0,1\}^n \to \{0,1\}$

Um mit dem Konzept vertraut zu werden, erstmal folgendes Beispiel (aus Russel & Norvig): es geht um die Entscheidung, ob wir in einem Restaurant warten, bis wir einen Tisch zugewiesen bekommen, oder weitergehen; also eine binäre Entscheidung. NB: wir suchen also unsere eigene Entscheidungsfunktion, möchten also eine Funktion die uns für jedes Restaurant sagt, ob wir warten würden!

Die Attribute sind hier nicht alle binär, aber das tut erstmal nichts zur Sache. Als erstes stellen wir die Liste der Merkmale zusammen, die für unsere Entscheidungsfunktion relevant sind (schöner wäre es natürlich, wenn wir diese Attribute automatisch erstellen könnten, dazu später mehr). Unsere Merkmale sind:

1. Alternativen: gibt es passende Alternativen in der Nähe?
2. Theke: können wir uns an die Theke setzen und schonmal ein Bier trinken?
3. Fr/Sa: ist es Freitag oder Samstag?
4. Betrieb: wie viel Betrieb ist im Lokal? (Werte: leer, einige Leute, voll)
5. Regen: regnet es draußen?
6. Reservierung: haben wir reserviert?
7. Typ: was für eine Art Restaurant haben wir (französisch, italienisch, deutsch)
8. Geschätzte Wartezeit (von uns geschätzt): 0-10,10-30,30-60,>60

Das sind also die Faktoren, die bestimmen, ob wir auf einen freien Tisch warten. Nicht alle Attribute sind binär; wie können sie aber leicht darauf reduzieren; z.B. Attribut 4. kann aufgespalten werden in 2 Attribute: Leer: ja/nein und Voll: ja/nein. Unser Hypothesenraum besteht also aus allen Funktionen

(358) $h : \{0,1\}^3 \times \{0,1,2\} \times \{0,1\}^2 \times \{0,1,2\} \times \{0,1,2,3\} \to \{0,1\}$

Wie viele solche Funktionen gibt es? Nehmen wir einfachheitshalber mal an, **H** wäre die Menge aller Funktionen

(359) $h' : \{0,1\}^8 \to \{0,1\}$

Wie groß ist unser Hypothesenraum? Man könnte meinen er wäre nicht übermäßig groß; aber der Eindruck täuscht:

$$\text{es gibt } 2^{2^8} \text{ solche Funktionen, also } 2^{64}$$

– eine wahnsinnig große Zahl. Unser Hypothesenraum ist also riesig! Unser Ziel muss es sein, eine möglichst einfache Funktion aus diesem Raum zu wählen, die (nach unseren Begriffen) gut verallgemeinert. Hierbei greift man auf die sogenannten **Entscheidungsbäume** zurück.

Bsp.	Alt	Theke	Fr/Sa	Bet	Reg	Res	Typ	Wart	Warten?
d1	1	0	0	halb	0	1	fr	0-10	1
d2	1	0	0	voll	0	0	it	30-60	0
d3	0	1	0	halb	0	0	de	0-10	1
d4	1	0	1	voll	1	0	it	10-30	1
d5	1	0	1	voll	0	1	fr	>60	0
...									

Table 2: Ein Ausschnitt aus unserem Datensatz

26.2 Entscheidungsbäume

Boolesche Funktionen lassen sich einfach als Tabellen auffassen; wir nehmen nun wieder unser Beispiel, um das darzustellen: Tabelle 1 ist nur ein kleiner Ausschnitt unserer Funktion; wir können auch annehmen, es handelt sich um unseren Datensatz D. Ein **Entscheidungsbaum** ist einfach ein Baum,

1. in dem jeder Knoten ein Merkmal repräsentiert,
2. jedes Blatt einen Wert, den die Funktion annimmt;
3. auf jedem Pfad von der Wurzel zu einem Blatt kommt dabei jedes Merkmal höchstens einmal vor.

Jede Boolesche Funktion lässt sich als als Entscheidungsbaum darstellen: wir können einfach den Baum nehmen, in dem jede *Zeile* unserer Tabelle einem *Pfad* entspricht. Es gibt gewisse Boolesche Funktionen, die lassen sich nicht oder nur sehr schwer kompakt repräsentieren, z.B.

> die **Paritätsfunktion** (f nimmt den Wert 1 an, wenn eine gerade Zahl von Argumenten den Wert 1 annimmt), oder
>
> die **Majoritätsfunktion** (f nimmt den Wert 1 an, falls mindestens die Hälfte seiner Argumente den Wert 1 annimmt).

Allerdings gibt es auch Entscheidungsbäume, die eine wesentlich kompaktere Darstellung erlauben. Wenn wir das obige Beispiel betrachten, dann fällt uns z.B. auf dass wann immer die geschätzte Wartezeit >60 Minuten beträgt, dann warten wir niemals darauf dass ein Tisch frei wird. Wenn sich dieses Muster durch alle unsere Beobachtungen zieht, dann können wir also

dieses Merkmal an die Wurzel unseres Baumes setzen, und dann können wir in einigen Fällen den Baum an dieser Stelle schon mit dem Blatt 0 beenden. Algorithmen zur Induktion von Entscheidungsbäumen beruhen genau auf dieser Beobachtung:

> Wir können die Komplexität von Booleschen Funktionen messen nach der Komplexität der Entscheidungsbäume.

Das wiederum passt zu unseren obigen Beobachtung, dass einfache Funktionen eher sinnvolle, interessante Generalisierungen liefern als komplexe. Wir bekommen also folgendes:

> Gegeben eine Menge D von Daten, finde den enfachsten Entscheidungsbaum, der mit D konsistent ist; die zugehörige Boolesche Funktion ist unsere Hypothese h.

Wie finden wir die? Man benutzt hier das sog. **Splitting**: wir nehmen das Merkmal, dass für unsere Unterscheidung **am informativsten** ist, und setzen es an die Wurzel des Entscheidungsbaumes. Dann nehmen wir das nächst-informativste Merkmal, setzen es als nächsten Knoten etc. Wie macht man das? Hier nutzen wir wieder einmal das Konzept der **Entropie**. Dafür müssen wir zunächst etwas arbeiten:

- Unser zugrundeliegende Raum ist eine Menge von Funktionen $X : M_1 \times M_2 \times \ldots \times M_i \to \{0, 1\}$.
- Wenn wir nun ein $n : 1 \leq n \leq i$ wählen, dann haben wir eine Funktion $X_n : M_n \mapsto (M_1 \times \ldots \times M_{n-1} \times M_{n+1} \times \ldots \times M_i \to \{0, 1\})$
- Das bedeutet: für jedes Merkmal, dass einen gewissen Wert annimmt, bekommen wir eine neue Funktion über die verbliebenen Merkmale.
- Wir möchten das Merkmal finden, das uns am besten die Menge der verbliebenen Funktionen aufteilt; insbesondere sollten die Teilmengen disjunkt sein!

Wir suchen also erstmal Merkmale M_n, für die gilt:

> Falls $m, m' \in M_n$, $m \neq m'$, dann ist $X_n(m) \cap X_n(m') = \emptyset$.

Das ist aber ein Kriterium, das gleichzeitig zu schwach (viele Merkmale können es erfüllen) und zu stark ist (in manchen Fällen wird es kein Merkmal geben, dass dieses Kriterium erfüllt.

Wir müssen also mal wieder Zuflucht zu Wahrscheinlichkeiten nehmen. Wir bauen daher den Wahrscheinlichkeitsraum $\mathfrak{A}$, wobei gilt:

1. $\Omega = M_1 \times M_2 \times \ldots \times M_i \rightarrow \{0, 1\}$ (die Menge der Ereignisse),
2. und für jedes $d \in \Omega$ gilt:

$$P(d) = \frac{1}{|D|} \text{ falls } f \in D, \text{ wobei } D \text{ unser Datensatz ist.}$$

Auf diesem Raum können wir nun eine Reihe von Zufallsvariablen $X_n : n \leq i$ definieren (wir fassen hier den Begriff etwas allgemeiner):

$$\text{Für } d = (m_1, \ldots, m_n, \ldots, m_i, x) \ (x \in \{0, 1\}),$$

gilt:

$$X_n(d) = m_n.$$

Man beachte, dass der **Zielwert** x (0 oder 1) hier nur ein weiteres Merkmal unter vielen ist! Nun hat jede dieser Zufallsvariablen eine Entropie, die sich errechnet als

$$H_P(X_n) = \sum_{m \in M_n} P(X_n = m) \cdot log(P(X_n = m)) \tag{360}$$

Damit bemessen wir, wie informativ eine Variable ist, und da die Variablen einem Merkmal entsprechen, bemessen wir also indirekt, wie informativ ein Merkmal ist. Das wäre aber zu allgemein: wir möchten ja nicht irgendein Merkmal vorhersagen, sondern ein ganz bestimmtes, unser **Zielmerkmal**. Hierzu brauchen wir das Konzept der **bedingten Entropie**:

$$\begin{aligned} H(X|Y) &= \sum_{y \in Y} H(X|Y = y)P(Y = y) \\ &= \sum_{x \in X, y \in Y} P(X^{-1}(x) \cap Y^{-1}(y)) log \left(\frac{P(X^{-1}(x) \cap Y^{-1}(y))}{P(Y^{-1}(y))} \right) \end{aligned}$$

Insbesondere interessiert uns die Entropie der Variable X_{Ziel}, also des Zielwertes, gegeben dass wir den Wert eines Merkmals kennen:

$$H_P(X_{Ziel}|X_n) \tag{361}$$

Was jedoch wichtiger ist als dieser Wert (der ja auch sehr extrem sein kann, auch wenn M_n keinen Einfluss darauf hat) ist der **Informationsgewinn**; der ist wie folgt definiert:

(362) $IG_P(X_{Ziel}|X_n) = H_P(X_{Ziel}) - H_P(X_{Ziel}|X_n)$

Je geringer die bedingte Entropie im Vergleich zur unbedingten ist, desto größer ist der Informationsgewinn. Falls

(363) $H_P(X_{Ziel}|X_n) = 0$

also der Wert von X_{Ziel} vollständig von X_n bestimmt wird, dann ist

(364) $IG_P(X_{Ziel}|X_n) = H_P(X_{Ziel})$

Das bedeutet: wir gewinnen sämtliche Information, die in X_{Ziel} enthalten ist. Was wir damit also suchen ist:

(365) $\underset{1 \leq n \leq i}{argmax}\ IG_P(X_{Ziel}|X_n)$

Das liefert uns das Merkmal, welches wir ganz oben in unseren Entscheidungsbaum stellen. Danach iterieren wir das mit den verbliebenen Variablen/Merkmalen: als nächstes interessiert uns

(366) $\underset{1 \leq n \leq i}{argmax}\ H_P(X_{Ziel}|X_{max}) - H_P(X_{Ziel}|X_{max}, X_n)$

und so weiter, so dass wir also $i!$ Schritte benötigen (ein Schritt ist hier die Berechnung der bedingten Entropie). Das ist ein gutes Ergebnis, da die Anzahl der Merkmale normalerweise überschaubar ist!

26.3 Overfitting I

Vorher haben wir die Tatsache benutzt, das gewisse Merkmale informativer sind als andere. Es gibt hierbei aber ein mögliches Problem: dass ein Merkmal *zu informativ* ist, nämlich keine Generalisierung enthält. Das passiert insbesondere, wenn das Merkmal viele Werte annehmen kann, schlimmstenfalls mehr als unser Datensatz an Punkten enthält. Ein Beispiel hierfür wäre, wenn wir ein Merkmal **Datum** hinzunehmen. Unter der Annahme, dass wir an jedem Tag nur einmal essen gehen, ist klar dass wir damit einen perfekten

Bsp.	Alt	Theke	Fr/Sa	Bet	Reg	Res	Typ	Wart	Tag	Warten?
d1	1	0	0	halb	0	1	fr	0-10	5	1
d2	1	0	0	voll	0	0	it	30-60	18	0
d3	0	1	0	halb	0	0	de	0-10	9	1
d4	1	0	1	voll	1	0	it	10-30	26	1
d5	1	0	1	voll	0	1	fr	>60	17	0
...										

Table 3: Die Daten mit dem Tag des Monats

Prädiktor für unseren Datensatz haben: das Datum gibt uns eindeutig die richtige Klassifizierung. Das Problem ist: es gibt dabei keine Generalisierung! Das bleibt bestehen wenn wir ein Merkmal haben **Tag des Monats** – auch das mag bei einem relativ kleinen Datensatz ein guter Prädiktor sein, hat aber vermutlich keine Relevanz.

Das zugrundeliegende Problem ist also, dass

$$\frac{|M|}{|D|},$$

der relativ groß ist, im schlimmsten Fall > 1. Wie gehen wir mit diesem Merkmal um? Wir können ja nicht davon ausgehen, dass die Irrelevanz eines Merkmals derart offen zutage liegt. Hier können wir die klassische statistische Analyse nutzen: die **Nullhypothese** ist, dass das Merkmal keinen Einfluss hatte auf unsere jeweilige Entscheidung. Wir können nun versuchen, diese Hypothese zu widerlegen: wir müssen belegen, dass es wahrscheinlich ist, dass die Verteilung des Merkmals M rein zufällig ist.

Dafür überlegen wir zunächst:

- Wie viele Werte kann das Merkmal M annehmen? Wir nennen diese Zahl $|M|$.
- Wie würde es aussehen, wenn diese Merkmale rein zufällig über die anderen verteilt würden? Es würde zunächst gleichmäßig gestreut sein, d.h. keine besondere Ko-Okkurenz mit anderen Merkmalen haben.

Den zweiten Punkt kann man wie folgt verdeutlichen: da $|M|$ in kritischen Fall relativ groß ist. muss man ein Merkmal M' nehmen mit möglichst kleinem $|M'|$. Ein besonderes Beispiel hierfür wäre das "Zielmerkmal" $\{0, 1\}$, das wir eigentlich vorhersagen möchten. In diesem Fall ist die **Nullhypothese** klar numerisch formulierbar; wir benutzen unsere Zufallsvariablen X_n, setzen fest (qua Definition):

$$M = M_j \qquad M' = M_k$$

Nun sollte laut Nullhypothese gelten:

für alle $m \in M_j, m' \in M_k$, $d \in D$:
$$P(X_j = m | X_k = m') \approx P(X_j = m)$$

Da $P(X_j = m)$ aber naturgemäss (qua Annahme dass $\frac{|M|}{|D|}$ relativ groß ist) eine Zahl ist, die für uns schwierig von 0 zu unterscheiden ist, ist das noch problematisch; wir können aber folgendes machen: nehmen wir Einfachheit halber an, alle anderen Merkmale außer M sind binär. Dann können wir eine neue Zufallsvariable Y annehmen, die eine Summe von Werten denotiert:

$$(367)\ Y(M) = \sum_{j \neq k} P(X_j = m | X_k = m')$$

(wobei i die Gesamtanzahl der Merkmale ist) als das Ergebnis eines $i-1$-Fach wiederholten Zufallsexperimentes lesen, wobei jeweils mit einem sehr großen Würfel geworfen wurde. Dementsprechend haben wir also eine Multinomialverteilung mit einem Erwartungswert

$$(368)\ \mathcal{E}(Y) = \frac{i-1}{|M|}$$

mit einer entsprechenden symmetrischen Verteilung, Varianz und Standardabweichung. Das bedeutet: wir können die üblichen Methoden der Vertrauensgrenzen etc. ohne weiteres anwenden.

26.4 Overfitting II

Wir können auch im Rahmen unserer Methodik der Informationstheorie bleiben, und den Begriff der **bedingten Entropie** nutzen. Hier nochmals die Definition:

$$\begin{aligned} H(X|Y) &= \sum_{y \in Y} H(X|Y = y) \\ &= \sum_{x \in X, y \in Y} P(X^{-1}(x) \cap Y^{-1}(y)) log \left(\frac{P(X^{-1}(x) \cap Y^{-1}(y))}{P(Y^{-1}(y))} \right) \end{aligned}$$

Nach unseren Annahmen für $M = M_j$ hat die Zufallsvariable X_j sicher eine hohe/maximale Entropie. Es ist also genau die Eigenschaft, die sie eigentlich

positive hervorheben, die sie auch problematisch macht! Hier sehen wir die zwei Seiten derselben Medaille: je größer $|M_j|$, desto größer die Entropie von $H_P(X_j)$; aber je größer $|M_j|$, desto größer die Gefahr, dass das Merkmal eigentlich keine relevante Information enthält. Wir können uns nun mit der bedingten Entropie helfen: sei X_{Ziel} die Zufallsvariable, die das Zielmerkmal unserer Daten liefert. Wir können nun z.B.

(369) $H_P(X_j|X_{Ziel})$

berechnent. Falls nun gilt:

(370) $H_P(X_j|X_{Ziel}) \approx H_P(X_j)$

dann wissen wir, dass das Ergebnis einen geringen Einfluss auf X_j hat (den Tag des Monats). Im Umkehrschluss bedeutet dass, dass auch andersrum wenig Information fließt; wir haben zwar diesen Eindruck, aber das ist nur der Größe $|M_j|$ geschuldet.

26.5 Implementierung eines Algorithmus

Wir betrachten wiederum den dataframe **verbs**:

```
> library(languageR)
> head(verbs)
```

Wie wir gesagt haben ist die Zielvariable des Frames die Variable RealizationOfRec, mit den beiden levels NP und PP. Das heißt wir möchten die englische Dativalternation möglichst gut vorhersagen. Wir möchten nun feststellen, welches Merkmal der beste Indikator hierfür ist. Hier gibt es zunächst ein Problem: wir LengthOfTheme haben wir eine Variable, die nicht diskret ist; das würde natürlich zu overfitting führen. Daher machen wir folgendes Preprocessing:

```
> for(i in 1:nrow(verbs)){
+       if(verbs[i,"LengthOfTheme"]>mean(verbs$LengthOfTheme)){
+            verbs.disk[i,"LengthOfTheme"] = "long"
+            }
+       else {verbs.disk[i,"LengthOfTheme"]= "short"}
+ }
```

Damit haben wir eine diskrete Variable mit zwei Werten: "long" heißt: länger als der Mittelwert, "short" heißt kürzer.

Jetzt brauchen wir den eigentlichen Algorithmus zur Bestimmung der bedingten Entropie. Dazu müssen wir for-Schleifen über die verschienenen levels laufen lassen. Es gibt den Befehl levels():

```
> levels(verbs.disk$Verb)
```

Allerdings ist das kein Vektor, wie wir ihn brauchen um eine for-Schleife laufen zu lassen. Stattdessen benutzen wir den Befehl **unique()**:

```
> levelVerb = unique(verbs.disk$Verb)
> levelTarget = unique(verbs.disk$RealizationOfRec)
```

Das ist nun ein Vektor. Um nun die Entropie

$$\begin{aligned} H(\text{RealizationOfRec}|\text{Verb}) &= \sum_{y \in Y} H(\text{ROR}|V = v) \\ &= \sum_{x \in X, y \in Y} P(ROR = x, V = v) log \left(\frac{P(ROR = x, V = v)}{P(V = v)} \right) \end{aligned}$$

auszurechnen, brauchen wir nur die obige Formel zu berechnen:

- Für jedes Paar (levelVerb,levelTarget) brauchen wir die relative Häufigkeit
- Für jedes levelVerb brauchen wir die relative Häufigkeit
- Sowie den Quotienten und Logarithmus hiervon.

Das ganze wird am Ende summiert. Wir/Sie haben folgende Aufgabe:

1. Berechnen Sie die (unbedingte) Entropie von der Target-Variable.
2. Schreiben Sie ein Programm, das für jede andere Spalte/Variable die bedingte Entropie berechnet der Zielvariable (RealizationOfRecipient) gegeben jeweils eine der anderen Variablen.

Der dickste Batzen an Programmierarbeit ist unten beschrieben. Wir berechnen für jedes Paar von Werten, die die Variablen annehmen können, die "lokale" Entropie; wir kriegen alle Paaren mit 2 for-Schleifen. Summiert wird mittels einer globalen Variable.

NB: das geht natürlich auch eleganter: z.B. mit einer Funktion condEntropy(), die ein Merkmal M aus verbs.disk nimmt und die bedingte Entropie berechnet $H(\text{ROR}|M)$. Das können Sie dann machen!

```
> nrow = nrow(verbs);
> condEntropy = 0;
> for(verb in levelVerb){
+     for(target in levelTarget){
+       count1 = 0;
+       count2 = 0;
+       for(n in 1:nrow) {
+             if(verbs.disk$Verb[n] == verb){
+                   count1 = count1+1;
+                   if(verbs.disk$RealizationOfRec[n] == target){
+                         count2 = count2+1;
+                          }
+                      }
+                  }
+            if(count2>0){
+                 condEntropy = condEntropy + (count2/nrow)*log2(count2/count1)
+           }
+      }
+ }
```

Aufgabe 10

Zu bearbeiten bis 10.7.2018. Berechnen sie die Informationsgewinne $IG(\text{RealizationOfRec}|X)$ für $X \in \{\text{LengthOfTheme, AnimacyOfTheme, Verb, AnimacyOfRec}\}$, mit LengthOfTheme als binärem Merkmal (long,short).

Lösung

Wir schreiben eine Funktion auf Basis des obigen Codes. Zunächst berechnen wir die Entropie der Variable RealizationOfRecipient:

```
> nrow = nrow(verbs.disk)
> countNP = 0
> for(i in 1:nrow){
+     if(verbs$RealizationOfRec] == "NP"){
+       countNP = countNP +1 }
+ }
```

```
> countPP = nrow - countNP
> HROR = -1*((countNP/nrow)* log2(countNP/nrow) + (countPP/nrow)*log2(countPP/nrow))
```

Nun haben wir die unbedingte Entropie der Variable. Als nächstes definieren wir eine Funktion, die für jedes Merkmal den Informationsgewinng berechnet:

```
> levelTarget = c("NP","PP")
> IG <-  function(x){
+     condEntropy = 0;
+     levels = unique(verbs.disk[,x])
+     for(level in levels){
+       for(target in levelTarget){
+         count1 = 0;
+          count2 = 0;
+          for(n in 1:nrow) {
+                if(verbs.disk[n,x] == level){
+                      count1 = count1+1;
+                      if(verbs.disk$RealizationOfRec[n] == target){
+                            count2 = count2+1;
+                             }
+                       }
+                    }
+               if(count2>0){
+                    condEntropy = condEntropy - (count2/nrow)*log2(count2/count1)
+               }
+         }
+          }
+      InfGain = HROR - condEntropy
+      return(InfGain)
+ }
```

Hier bekomme ich folgende Ergebnisse:

> IG("Verb")
[1] 0.3609774
> IG("LengthOfTheme")
[1] 0.07322383
> IG("AnimacyOfTheme")
[1] 0.001099158
> IG("AnimacyOfRec")
[1] 0.01104288

27 Probabilistische Graphische Modelle I - Bayesianische Netze

27.1 Korrelation und Kausalität

Ein (hoffentlich) bekannter Gemeinplatz ist, dass Kausalität und Korrelation nicht das gleiche sind. Hier einige Beispiele:

- Die Rückkehr der Störche nach Europa im Frühjahr fällt zusammen mit einem saisonalen Anstieg der Geburtenrate
- Der Fall in der bundesdeutschen Geburtenrate (Pillenknick) fällt zusammen mit einem massiven Rückgang der Population der Störche (Zerstörung der Lebensräume)
- Größe korrelliert mit Intelligenz in der allgemeinen Population (tatsächlich). Allerdings machen Wachstumshormone nicht intelligenter!
- Die weltweite Bevölkerung korrelliert mit dem Alter der britischen Königin (zumindest in den letzten 50Jahren)
- Ein interessantes Beispiel: eine Helmpflicht korreliert mit proportional mehr verletzten und toten Fahrradfahrern (so geschehen in der Schweiz, Australien)
- Die allgemeine Helmquote, nach Land, korreliert mit mehr verletzten und toten Fahrradfahrern (höhere Helmquote $\sim$ mehr Tote und Verletzte!)
- Vegetarier sind statistisch gesünder als Omnivoren. Ob das wirklich am Vegetarismus liegt ist aber nicht abschließend geklärt: Vegetarismus korreliert mit Bildung und sozio-ökonomischem Status, das wiederum korrelliert mit Gesundheit!
- Zuletzt gelesen: Vegetarier sind proportional öfter depressive als Omnivoren.

Folgende Beispiele sind besonders unklar:

- Kinder, die täglich mehr als 30min musikalische gefördert werden, schneiden in vielen Bereichen deutlich besser ab. Aber ist das wirklich die musikalische Förderung?

- Alzheimererkrankte haben (post mortem) eine erhöhte Konzentration von Pestiziden im Gehirn (wir alle essen täglich Pestizide). Bedeutet das, dass der erhöhte Verzehr von Pestiziden zu Alzheimer führt?

Diese Beispiele haben unterschiedliche Erklärungen, aber meistens eines gemeinsam: es gibt ungenannte, "verdeckte" Variablen, welche die Beobachtungen verbinden. Aber selbst wenn wir diese Variablen "offengelegt" haben, ist immer noch nicht klar, in welche Richtung wir einen kausalen Effekt haben. Üblicherweise lässt sich das mit dem gesunden Menschenverstand schnell sehen, aber es gibt hier einige Probleme:

- Nicht immer ist die Lage so klar wie in diesen Beispielen
- Der gesunde Menschenverstand ist nicht so universell wie wir das gerne hätten
- Und die Berufung auf Intuition ist per se immer problematisch bzw. unwissenschaftlich

Es stellt sich also die Frage:

Die Frage der Kausalität Können wir mit den Mitteln der Wahrscheinlichkeitstheorie bestimmen, dass Variable X einen kausalen Einfluss ausübt auf Variable Y und nicht umgekehrt?

Das ist eine gute Frage, unser bisheriges Mantra war doch: Wahrscheinlichkeitstheorie ist eine Theorie der Information, und Information ist wie Strom: sie fließt *immer* in beide Richtungen! Anders gesagt: die konditionale (Un)Anbhängigkeit zweier Variable ist ein *symmetrisches Konzept*, wenn X relevante Information für Y enthält, dann gilt dasselbe zwangsläufig auch umgekehrt! Also wissen wir bereits die Antwort, wenn es nur *zwei* Variablen gibt: **Nein**.

Aber: falls mehr Variablen involviert sind, lautet die Antwort: unter Umständen **ja**! Das liegt daran, dass gewisse Konstellationen von konditionalen (Un)abhängigkeiten nur mit Abhängigkeiten in gewissen Richtungen erklärbar sind.

Vegetarismus, genauer betrachtet

Betrachten wir einmal das letzte Beispiel:

1. Vegetarier sind öfter depressiv als Omnivoren

Das klingt natürlich erstmal nach einer Schlagzeile, wir sollten aber folgendes Bedenken:

2. Vegetarier sind öfter Frauen

3. Frauen sind öfter (diagnostiziert!) depressiv

Das ändert die Sachlage: wir haben nun drei Variablen, und die Korrelation zweier Variablen kann evtl. durch die dritte gemittelt bzw. "erklärt" werden. Aber wie machen wir das?

Zwei erleuchtende Beispiele

Ein Beispiel: das Stanford-Musiker Paradox Stanford ist eine Elite-Uni in Kalifornien, die nur sehr guten Studenten zugang gewährt. Es gibt aber auch spezielle Stipendien für herausragende Musiker. Wir haben also drei Variablen:

1. I, gibt an die Intelligenz eines Individuums (wie auch immer gemessen)
2. M, gibt an die Musikalität eines Individuums (wie auch immer gemessen)
3. S gibt an, ob das Individuum in Stanford ist.

Wir haben nun folgende Korrelationen:

- $P(S = 1|I > x) > P(S = 1)$ (Intelligenz korreliert mit Zulassung in Stanford)
- $P(S = 1|M > x) > P(S = 1)$ (Musikalität korreliert mit Zulassung in Stanford)
- Beide Korrelation gelten natürlich auch umgekehrt, also $P(M > x|S = 1) > P(M > x)$ etc.
- außerdem vielleicht: $P(M > y|I > x) > P(M > y)$ denn Musikalität korreliert allgemein schwach positiv mit Intelligenz

Nun werden wir aber folgendes finden: wenn ein Mensch in Stanford studiert, und er ist *nicht* musikalisch, dann wird er eher intelligent sein, als ein Student der sehr musikalisch ist. Warum? Naja, wir wissen ja dass seine Zulassung nicht an der Musik gelegen haben kann! Also:

$$P(I > x|S = 1, M < y) > P(I > x|S = 1, M > y)$$

(und natürlich gilt das ganze auch umgekehrt). Ein solches Phänomen ($S = 1$ verkehrt die Korrelation von M und I) ist nur dann möglich, wenn es eine **Direktationalität** gibt der Beeinflussung von M nach S und von von I nach S:

$$M \longrightarrow S \longleftarrow I$$

Noch ein Beispiel Menschen geben Geld für sinnlose Sachen aus, aber (fast) nur reiche Menschen geben *sehr viel* Geld für sinnlose Sachen aus. Diese Tatsache öffnet eine ganze Reihe von statistischen Korrelationen: die Tatsache z.B. dass jemand einen BMW-X3 besitzt macht es für uns wahrscheinlicher, dass er ein komplettes Golf-Schlägersortiment besitzt. Aber nur aus einem Grund:

- es ist Evidenz dafür, dass er reich ist (Evidenz ist (eine Funktion der) likelihood!)
- und das erhöht die Wahrscheinlichkeit, Golfschläger zu besitzen.

Wir haben also eine natürliche Abhängigkeit der Variablen G und RR.

Wenn wir aber wissen, ob jemand reich ist, dann verschwindet dieser Zusammenhang: denn warum sollten reiche Leute, die Geld für überteuerte Autos ausgeben, eher Geld für überteuerte Sportarten ausgeben, als andere reiche Leute?

Dieses Szenario ist also genau umgekehrt zum Stanford Szenario: dort schuf die mittlere Variable eine Abhängigkeit, hier schafft sie eine **Unabhängigkeit**. Wie wir sehen werden, liefert uns auch das (begrenzt) Aufschluss auf die Direktionalität der Beeinflussung, wir haben nämlich folgendes Szenario:

$$\mathit{Golf} \longleftarrow \mathit{Reich} \longrightarrow \mathit{BMW\text{-}X3}$$

Was wir also hier sehen ist folgendes: wenn wir ausreichend viele Variablen haben, sowie ausreichende Information über konditionale Abhängigkeit/ Unabhängigkeit, dann können wir durchaus Rückschlüsse auf kausale Abhängigkeit ziehen!

Bsp.	Alt	Theke	Fr/Sa	Bet	Reg	Res	Typ	Wart	Warten?
d1	1	0	0	halb	0	1	fr	0-10	1
d2	1	0	0	voll	0	0	it	30-60	0
d3	0	1	0	halb	0	0	de	0-10	1
d4	1	0	1	voll	1	0	it	10-30	1
d5	1	0	1	voll	0	1	fr	>60	0
d6	1	0	1	voll	1	0	it	10-30	0
d7	0	1	0	halb	1	0	de	10-30	1
...									

Table 4: Ein Datensatz, der unsere Entscheidung nicht funktional bestimmt.

27.2 Einleitung 2: Entscheidungsfunktionen generalisieren

Nehmen wir einmal an, unser Datensatz ist so gestrickt, dass er keine Funktion mehr ist: das Zielmerkmal ist nicht mehr eindeutig durch die übrigen Merkmale determiniert. In Tabelle 3 etwa unterscheiden sich d4 und d6 nur durch den Zielwert, alle anderen Merkmale sind gleich!

Es ist klar, dass wir in diesem Fall keinen Entscheidungsbaum induzieren können: die Entscheidungen sind ja durch keinen Baum eindeutig bestimmt! Eine häufige Ursache für derartige Konstellationen ist, dass unsere gelisteten Faktoren nicht die einzig relevanten sind. Z.B. unsere Laune, Hunger, Begleitung etc. mag ebenfalls eine Rolle spielen, nur dass das Faktoren sind, über die wir keine Information haben. Das kann verschiedene Gründe haben:

- Die Information ist nicht oder nur schwer beobachtbar (z.B. unsere Laune)
- Der Indikator hat zu viele Werte, um sinnvoll benutzt zu werden (z.B. Begleitung)
- Es gibt einfach Nicht-determinismus!

Das bedeutet aber natürlich nicht, dass wir keine wertvolle Information aus den Merkmalen bekommen für unser Zielmerkmal: wir können z.B. leicht sehen dass uns die Wartezeit immer noch eine ziemlich relevante Information

liefert: indem sie nämlich die Wahrscheinlichkeitsverteilung ändert:

$$(371)\quad P(\text{Warten} = 1) = \frac{4}{7}$$

– d.h. wir haben wenig Information; aber wenn wir nach der obigen Methode die bedingte Wahrscheinlichkeit schätzen, dann bekommen wir:

$$(372)\quad P(\text{Warten} = 1|\text{Wartezeit} < 30) = \frac{4}{5}$$

Wir können insbesondere leicht sehen, dass die vorher angewandte Methodik der bedingten Wahrscheinlichkeit, bedingten Entropie nach wie vor problemlos angewendet werden kann. Aber was machen wir mit dieser Information? Die Frage ist also, welches Modell wir nutzen sollen; unsere allgemeinen Erwägungen bringen uns zu der Auffassung, das wir die in den Daten enthaltene Information nutzen sollten, um das Modell *möglichst einfach* zu gestalten. Wir werden hier ein besonders interessantes Modell betrachten, die sog. Bayesianischen Netze.

Bsp.	Jahreszeit	Temperatur	Regen	Schnee/Eis	Unfälle/Tag
d1	Sommer	mittel	1	0	40
d2	Winter	niedrig	0	1	90
d3	Winter	niedrig	0	0	31
d4	Sommer	hoch	0	0	30
d5	Winter	niedrig	0	0	45
...					

Table 5: Abhängigkeiten und bedingte Unabhängigkeiten

27.3 Verdeckte Variablen/Rauschen

Die Intuition hinter diesen Strukturen ist folgende: es mag durchaus sein, dass eine ganze Reihe von Faktoren Einfluss hat auf die Wahrscheinlichkeit eines Ereignisses; allerdings ist die Wahrscheinlichkeit bereits von einer Teilmenge der Ereignisse bestimmt – wenn wir gewisse Dinge wissen, dann spielen andere keine Rolle, die sonst allerdings eine Rolle spielen würden. Wir können z.B. den Datensatz in Tabelle 4 betrachten:

Hier entspricht jeder Datenpunkt einem bestimmten Tag. Es ist klar, dass jeder einzelne Faktor einen Einfluss auf die Zahl der Unfälle hat, und dementsprechend auf die Unfallwahrscheinlichkeit (wir können das errechnen als Unfälle/Bevölkerung, als einfachste Lösung).

Des weiteren ist aber auch folgendes klar: wenn wir wissen, dass Schnee/Eis positiv ist, dann spielen Temperatur und Jahreszeit keine Rolle mehr. Sommer/Winter mögen relevant sein, aber nur insofern, als sie die Häufigkeit von Regen beeinflussen; ebenso weil Winter die Wahrscheinlichkeit von Schnee/Eis erhöht, wodurch die Unfallzahlen am stärksten steigen. Natürlich spielen noch eine Reihe anderer Faktoren eine wichtige Rolle; insbesondere z.B. das *Verkehrsaufkommen* – darüber haben wir aber keine gesicherten Information, daher müssen wir das als einen reinen **Störfaktor** auffassen, also einen Faktor, der sich auf unberechenbare Art und Weise in unseren Wahrscheinlichkeiten widerspiegelt. Z.B. d5 ließe sich auf diese Art und Weise erklären. Allerdings gibt es hier zu beachten: es kann natürlich auch sein, dass dieser Faktor *systematisch* wirkt, nämlich dadurch, dass winters ein höheres Verkehrsaufkommen ist als im Sommer.

Dadurch, dass es im Normalfall derartige Störfaktoren gibt, wird man auch mit aus solchen Datensätzen geschätzten Wahrscheinlichkeiten *praktisch niemals* eine echte konditionale Unabhängigkeit finden – das wäre sogar

dann unwahrscheinlich, wenn die Daten von einem echten Bayesianischen Netz generiert würden, rein aus Zufall. Daher ist es nicht immer einfach, den korrekten unterliegenden Graphen zu finden, und man wendet oft eine Mischung aus *common sense* und Datenanalyse an, um zu einem befriedigenden Ergebnis zu kommen. Dazu später mehr!

27.4 Definitionen

Ein **Graph** ist eine Struktur (V, E), wobei V eine Menge von Knoten ist (*vertices*), $E \subseteq V \times V$ die Kanten (*edges*). In Graphen gilt normalerweise: falls $(v_1, v_2) \in E$, dann $(v_2, v_1) \in E$, d.h. die Kanten haben keine Verbindung, sie repräsentieren nur Verbindungen. Ein **gerichteter** Graph ist ein Graph, in dem diese Bedingung fallengelassen wird: Kanten sind gerichtet. Ein gerichteter azyklischer Graph ist ein gerichteter Graph, in dem folgendes gilt: ein **Zyklus** ist eine Sequenz von Kanten

$$(v_1, v_2), (v_2, v_3), ..., (v_{i-1}, v_i), \text{ bei der } v_i = v_1$$

gilt. Wir folgen also den (gerichteten) Kanten und kommen zum Ausgangspunkt zurück. Ein **gerichteter azyklischer Graph** ist nun einfach ein gerichteter Graph, der keine Zyklen enthält. Wir werden hier normalerweise endliche Graphen betrachten.

Wir haben bereits Markov-Ketten kennengelernt; was wir nun machen ist folgendes: wir betrachten Ketten als Spezialfälle von gerichteten azyklischen Graphen mit der **Markov Eigenschaft**, und wollen nun zum allgemeineren Fall. Zunächst müssen wir die Definition betrachten: eine Markov-Kette hatte die Form

$$X_1, X_2, X_3, ..., X_n, ...$$

wobei jedes X_i eine Zufallsvariable war. Weiterhin gilt:

Fall $i < j < k$, dann ist $P(X_k = y_k | X_j = y_j, X_i = y_i) = P(X_k = y_k | X_j = y_j)$

D.h. bedeutet der Informationsfluss entlang der Kette wird dadurch, dass wir den Wert eines Zwischengliedes kennen, *blockiert*. (Erinnern Sie sich, dass viele andere Dinge, die man auf den ersten Blick meinen würde, nicht gelten!) Anders gesagt, wenn wir den Wert eines Gliedes kennen, sind die Werte aller vorigen Glieder irrelevant.

Wie verallgemeinern wir das? Zunächst folgende Begriffe: ein GAG entspricht einer **partiellen Ordnung** $\leq$, welche die transitive Hülle der Kanten E ist. Das heißt sie erfüllt folgende Axiome:

1. Reflexivität: $x \leq x$
2. Antisymmetrie: $x \leq y \ \& \ y \leq x \Rightarrow x = y$
3. Transitivität: $x \leq y \ \& \ y \leq z \Rightarrow x \leq z$

(Das entspricht der natürlichen Ordnung der Zahlen oben). $<$ ist die irreflexive Variante von $\leq$. Ein weiterer Begriff ist der des unmittelbaren Vorgängers. Wir definieren:

$$eltern(v) = \{v' : v' < v, \text{ es gibt kein } v'' : v' < v'' < v\}$$

Bayesianische Netze basieren darauf, dass wir einen GAG (V, E) haben, und jedem $v_k \in V$ eindeutig eine Zufallsvariable X_k zugewiesen wird. Wir können bereits einen der wichtigsten Begriffe der Bayesianischen Netze formulieren: wir sagen X_k **blockiert** einen Pfad von X_i nach X_j, falls v_k auf diesem Pfad von v_i nach v_j liegt.

Nun kommt die Definition von Bayesianischen Netzen: ein solches Netz ist eine Struktur $(V, E, \mathbf{X})$, wobei

(373) $\mathbf{X} = \{X_v : v \in V\}$

eine Menge von Zufallsvariablen ist die eindeutig Knoten in V zugeordnget werden, und die die Markov-Eigenschaft im Hinblick auf (V, E) erfüllen. Aber was genau heißt das? Tatsächlich ist das keine leichte Frage, und die Antwort hält einige Überraschungen bereit.

Definition 18 *Eine Menge von Verteilungen $\boldsymbol{X}$ erfüllt die Markov Eigenschaft im Hinblick auf einen GAG (V, E), falls für alle $X_k \in \boldsymbol{X}$, $\boldsymbol{Y} \subseteq X$ gilt: falls $Y < X$ für alle $Y \in \boldsymbol{Y}$, dann ist*

$$P(X|\,\boldsymbol{Y}, eltern(X)) = P(X|eltern(X))$$

Das bedeutet: um die genaue Verteilung einer Variable gegeben eine Teilmenge ihrer Vorgänger zu kennen, reicht es aus, die Werte der Eltern zu kennen. Die Beschränkung auf Vorgänger ist sehr wichtig, wie wir später sehen werden! Für uns ist zunächst das wichtigste: ein Bayesianisches Netz wird induziert durch eine Reihe bedingter Wahrscheinlichkeitsverteilungen

$$P(X_v|X_{v_1}, ..., X_{v_k}), \text{ wobei } \{v_1, ..., v_k\} = eltern(v).$$

Das ganze sieht normalerweise wie folgt aus (praktisch): wir nehmen an, die Zufallsvariablen nehmen nur endlich viele Werte an (aber das ist nur eine pädagogische Vereinfachung). Das bedeutet: wir spezifizieren

$$P(X_v = x|X_{v_1} = y_1, ..., X_{v_k} = y_k),$$

für alle

$$x \in X_v,\ y_1 \in X_{v_1},\ ...,\ y_k \in X_{v_k},$$

also jeden Wert, den die Variablen annehmen können. Damit, und mit den Regeln der Wahrscheinlichkeitstheorie, ist die resultierende Wahrscheinlichkeitsverteilung vollkommen determiniert (und sie ist die Verteilung eines Bayesianischen Netzes!).

27.5 Rechnen mit BNs

Für Markov Ketten haben wir bereits folgende Beobachtung gemacht:

> Information läuft immer in beide Richtungen der Kette, (Un-) Abhängigkeit ist immer symmetrisch.

Dasselbe gilt für BNs; es gibt allerdings auch einen sehr wichtigen Unterschied: für Markov Ketten gibt es eine **Symmetrie**: die Verteilungen selber geben uns niemals Aufschluss über Direktionalität; wir wählen sie immer willkürlich. Für BNs gilt das **nicht**; das ist eine der wichtigsten Beobachtungen in diesem Kontext. Um das zu sehen, müssen wir zunächst lernen, wie wir Wahrscheinlichkeiten in einem BN effektiv berechnen (gegeben $(V, E, \mathbf{X})$) und entsprechende bedingte Wahrscheinlichkeitsverteilungen

$$P(X|eltern(X)), \text{ für alle } X \in \mathbf{X}$$

Zunächst berechnen wir einfache *unbedingte Wahrscheinlichkeiten* der Form $P(X_v = x)$. Das ergibt sich aus folgender Gleichung:

(374)
$$P(X_v = x) = \sum_{y_1 \in X_1, ..., y_i \in X_i} P(X_v = x | X_1 = y_1, ..., X_i = y_i) P(X_1 = y_1, ..., X_i = y_i),$$

wobei $\{X_1, ..., X_i\} = eltern(X)$. 374 ist nichts weiter als die bekannte Regel der **Marginalisierung**. Das bedeutet aber: um die unbedingte Wahrscheinlichkeit eines *einzelnen* Ereignisses $X_v = x$ auszurechnen, muss man bereits die gesamten unbedingten Vorgängerwahrscheinlichkeiten berechnen – für alle Werte, die sie annehmen können!

> Es ist leicht zu sehen dass damit der Rechenaufwand um $P(X_v = x)$ zu berechnen *exponentiell* ist in der Anzahl von Vorgängern von X_v.

Das ist natürlich ein Problem, dass sich allerdings nur stellt, wenn uns diese Frage wirklich *ad-hoc* interessiert (was normalerweise eher selten der Fall ist). Angesichts dessen ist auch klar, warum wir die Anzahl der Knoten/Kanten immer möglichst klein halten sollten: abgesehen von allgemeinen Erwägungen (Ockhams Rasiermesser) sprechen auch ganz konkrete Berechenbarkeitsüberlegungen dafür!

Als nächstes interessiert uns die Frage, wie wir *beliebige bedingte Wahrscheinlichkeiten* brerechnen. Das geht nach folgender Gleichung:

(375)
$$P(X_v = x|\mathbf{Y}) = \sum_{y_1 \in X_1, ..., y_i \in Y_i} P(X_v = x|\mathbf{Y}, X_1 = y_1, ..., X_i = y_i) P(X_1 = y_1, ..., X_i = y_i|\mathbf{Y}),$$

wobei wiederum $\{X_1, ..., X_i\} = eltern(X) - \mathbf{Y}$, also die Menge der Eltern-Variablen ist, die nicht in $\mathbf{Y}$ enthalten ist. (375) liefert einen allgemeinen Fall; die Berechnung kann relativ einfach sein, falls

$$\mathbf{Y} \cap eltern(X)$$

verhältnismäßig groß ist, oder aber $\mathbf{Y}$ aus Vorgängern von X_v besteht. Falls $\mathbf{Y}$ aber *Nachfolger* von X_v enthält, wird die Berechnung nochmals aufwändiger. Die Berechnung einer Verteilung

$$P(X_1 = x_1, ..., X_i = x_i|Y_1 = y_1, ..., Y_i = y_i)$$

Lässt sich wiederum sehr einfach auflösen nach der Produktregel.

27.6 Konditionale (Un-)Abhängigkeit

Einer der wichtigsten Begriffe ist der der konditionalen Unabhängigkeit in BNs. Sie ist wie folgt definiert:

$(X \| Y | \mathbf{Z})$ gdw. für alle $x \in X$, $y \in Y$, $P(X = x | Y = y, \mathbf{Z}) = P(X = x | \mathbf{Z})$

In Worten bedeutet das soviel wie: wenn wir die Werte der Variablen $\mathbf{Z}$ kennen, dann spielt der Wert von Y keine Rolle für die Verteilung von X. Insofern ist die Markov Bedingung in BNs nur ein Fall von konditionaler Unabhängigkeit, der erfüllt sein muss. Es gilt aber noch viel mehr:

(376) $(X \| Y | Y)$

gilt immer, ebenso

(377) $(X \| Y | \mathbf{Y})$, falls $Y \in \mathbf{Y}$

Weiterhin gilt:

(378) $(X \| Y | \mathbf{Z}) \Leftrightarrow (Y \| X | \mathbf{Z})$

Konditionale Unabhängigkeit ist also eine symmetrische Eigenschaft (wie wir das erwarten würden: Information fließt immer in beide Richtungen.

Interessanterweise lässt sich die konditionelle Unabhängigkeit zweier Variablen auf rein strukturelle Eigenschaften des unterliegenden Graphen reduzieren, nämlich durch das Konzept der *d*-**Separation**. Ein wichtiger Begriff ist der eines **Pfades**, der etwas unintuitiv definiert wird; ein Pfad in einem Netz ist einfach eine Sequenz von Kanten

$$(v_1, v_2), (v_2, v_3), ..., (v_{i-1}, v_i)$$

in dem

1. adjazente Knoten jeweils identisch sind, und
2. für alle (v_l, v_{l+1}) des Pfades gilt: entweder $(v_l, v_{l+1}) \in E$ oder $(v_{l+1}, v_l) \in E$.

Wir sagen v liegt auf dem Pfad P mit der offensichtlichen Bedeutung dass für

$$P = (v_1, v_2), (v_2, v_3), ..., (v_{i-1}, v_i)$$

wir folgendes haben:

$$v = v_k \text{ für ein } k \text{ so dass } 1 \leq k \leq i$$

Teilpfade sind zusammenhängende Teilsequenzen eines Pfades. Wir unterscheiden zwei/drei Arten von (Teil-)Pfaden, nämlich

1. eine *Kette* hat die Form $(v_1, v_2), (v_2, v_3)$, wobei $(v_1, v_2) \in E$ und $(v_2, v_3) \in E$; (also $v_1 \rightarrow v_2 \rightarrow v_3$)

2. eine *Gabel* hat die Form $(v_1, v_2), (v_2, v_3)$, wobei $(v_2, v_1) \in E$ und $(v_2, v_3) \in E$; (also $v_1 \leftarrow v_2 \rightarrow v_3$)

3. ein V hat die Form $(v_1, v_2), (v_2, v_3)$, wobei $(v_1, v_2) \in E$ und $(v_3, v_2) \in E$; (also $v_1 \rightarrow v_2 \leftarrow v_3$)

Die entscheidende Definition ist folgende:

Definition 19 *Ein Pfad P in (V, E) ist d-separiert von einer Menge von Knoten $M \subseteq E$, falls eines der folgenden gilt:*

1. *P enthält einen Teilpfad $(v_1, v_2), (v_2, v_3)$, die entweder eine Kette oder eine Gabel ist, und $v_2 \in M$; oder*

2. *P enthält einen Teilpfad $(v_1, v_2), (v_2, v_3)$, der ein V ist, und $v_2 \notin M$.*

Zwei Knoten v, v' in (V, E) sind d-separiert von einer Menge $M \subseteq V$, genau dann wenn alle Pfade in (V, E) von v nach v' d-separiert sind von M.

Wir haben Knoten im Graphen eindeutig Zufallsvariablen zugeordnet. Dementsprechend können wir sagen dass in einem BN $(V, E, \mathbf{X})$ eine Variable Z zwei Variablen X, Y d-separiert. Das entscheidende Ergebnis ist folgendes:

Theorem 20 *Für jedes BN $(V, E, \boldsymbol{X})$, $X, Y \in \boldsymbol{X}$, $\boldsymbol{Z} \subseteq \boldsymbol{X}$ gilt: $(X \| Y | \boldsymbol{Z})$ genau dann wenn X, Y d-separiert sind von $\boldsymbol{Z}$ in $(V, E, \boldsymbol{X})$.*

Das bedeutet: um das Kriterium der konditionellen Unabhängigkeit zu verifizieren reicht es, einen Blick auf den Graphen des BN zu werfen!

27.7 Minimalität und Direktionalität

Der Normalfall ist allerdings der, dass wir das BN nicht als gegeben bekommen; stattdessen haben wir eine Wahrscheinlichkeitsverteilung über die Werte der Variablen; also eine Funktion

$$(\#)\ P : M_1 \times M_2 \times ... \times M_i \to [0, 1],$$

die die üblichen Bedingungen eines Wahrscheinlichkeitsraumes erfüllt (man nennt das eine **multivariate Verteilung**. Wichtig ist aber: es handelt sich hier nicht um einen Produktraum im engeren Sinne, d.h. die Wahrscheinlichkeiten der Komponenten sind *nicht* voneinander unabhängig. Das Ziel ist nun folgendes: wir suchen ein graphisches Modell (d.h. für uns ein BN), dass **minimal** ist, d.h. eine minimale Anzahl von Abhängigkeiten hat. Die Motivation hierfür ist folgende: je weniger Anhängigkeiten ein Modell hat, desto *einfacher* ist, und je einfacher es ist, desto *besser*.

Es gibt hier aber noch eine wichtige Eigenschaft: wir haben bereits gesagt, dass Abhängigkeiten in BNs direktional sind, d.h. sie gehen von A nach B, nicht umgekehrt. Diese Direktionalität kann man verknüpfen mit dem Begriff der **Kausalität**: Kausalität ist – im Gegensatz zu konditionaler Abhängigkeit – ein asymetrischer Begriff. Dadurch wird klar, dass konditionale Abhängigkeit kein guter Indikator für Kausalität sein kann. Wir haben bereits gesehen, dass es oftmals unklar ist, in welche Richtung Kausalität fließt: erinnern wir uns an das Beispiel von Regen/Temperatur: Regen beeinflusst die Temperatur (kühlt ab); aber die Temperatur hat auch Einfluß auf den Regen (bei großer Kälte kein Regen, bei Wärme öfter Gewitter). Hier kommt uns nun unser Modell zur Hilfe:

> Wenn es für jede Verteilung P wie in (#) ein eindeutiges, minimales BN gäbe, dass P erzeugt, dann könnten wir an der Direktionalität seiner Kanten die Richtung der kausalen Wirkung ablesen.

Leider gilt das nur bedingt: wir wissen bereits, dass für ein Netz der Form

$$X_1 \to X_2 \to X_3$$

wir ein äquivalentes Netz haben der Form:

$$Y_1 \leftarrow Y_2 \leftarrow Y_3$$

Wir können das transformieren mittels:

$$(379)\quad P(Y_1|Y_2) = P(X_2|X_1)\frac{P(X_1)}{P(X_2)}$$

und

$$(380)\quad P(Y_2|Y_3) = P(X_3|X_2)\frac{P(X_2)}{P(X_3)}$$

wobei man $P(X_3)$ etc. nach den gewohnten Regeln ausrechnet. Ebenso für ein Netz der Form

$$X_1 \leftarrow X_2 \rightarrow X_3$$

Denn wir können auch hier die Richtungen umdrehen, z.B. zu

$$(+)\ Y_1 \rightarrow Y_2 \rightarrow Y_3$$

(Ein ähnliches Argument wie oben) Das gilt allerdings nicht mehr, wenn wir sog. V-Strukturen haben:

$$(*)\ X_1 \rightarrow X_2 \leftarrow X_3$$

Warum ist das so? Die Antwort hat mit dem Stanford-Musiker Paradoxon zu tun. Zunächst führen wir das Symbol $\dot{\sim}$ mit der Bedeutung der *konditionalen Abhängigkeit* ein, also

$$(381)\quad (X_1 \dot{\sim} X_2)|X_3 \Longleftrightarrow \neg(X_1 \| X_2|X_3)$$

Zunächst haben wir in (*) eine konditionale Abhängigkeit von X_2 und X_1, egal ob X_3 gegeben ist oder nicht; ebenso umgekehrt für X_2, X_3 gegeben X_1; also

$$(X_1 \dot{\sim} X_2)|X_3 \quad (X_3 \dot{\sim} X_2)|X_1$$

Das allein ist aber noch kein Argument, denn dasselbe gilt für (+). Allerdings haben wir auch noch folgendes:

$$(X_1 \dot{\sim} X_3)|X_2$$

Das gilt nun weder für Ketten noch für Gabeln, wie wir oben gesehen haben! Daraus können wir folgern: Für eine Verteilung über drei Variablen, die folgende 4 Bedingungen erfüllt:

$$(X_1 \not\perp\!\!\!\perp X_2)|X_3 \tag{382}$$

$$(X_3 \not\perp\!\!\!\perp X_2)|X_1 \tag{383}$$

$$(X_1 \not\perp\!\!\!\perp X_3)|X_2 \tag{384}$$

$$X_1 \| X_3 | \emptyset \tag{385}$$

In diesem Fall ist eine V-Struktur das eindeutige minimale BN. Das wiederum erlaubt es uns, die Direktionalität der Pfeile zu inferieren! Selbstverständlich gilt das auch *a fortiori* für größere Graphen/Verteilungen mit mehr Variablen, in denen diese Teilverteilungen auftreten.

> Gegeben eine Verteilung mit n Variablen können wir tatsächlich – bis auf Direktionalität in Gabeln und Ketten – einen eindeutiges minimales BN finden.

Das ist noch etwas ungenau; genauer wird es durch folgenden Definitionen und Ergebnisse (die auch noch etwas salopp formuliert sind):

Definition 21 *Zwei GAGs G_1, G_2 sind **probabilistisch äquivalent**, wenn für jedes BN, dass auf G_1 basiert, es ein auf G_2 basiertes BN gibt, dass dieselbe Wahrscheinlichkeitsverteilung beschreibt, und umgekehrt.*

Ein wichtiges Ergebnis ist folgendes:

Theorem 22 *Zwei GAGs G_1, G_2 sind probabilistisch äquivalent, genau dann wenn das gleiche Skelett (ungerichtete Kanten) und die gleiche Menge an V-Strukturen darüber haben.*

27.8 Von der Verteilung zum Graphen

Nun die Frage: wie komme ich von einer **multivariaten Verteilung** zu den bedingten Wahrscheinlichkeiten, die ich für mein BN brauche? Hier können wir einmal mehr unsereursprüngliche Definition der bedingten Wahrscheinlichhkeit ausgraben. Sei eine multivariate Verteilung

$$P : M_1 \times M_2 \times ... \times M_i \rightarrow [0, 1]$$

gegeben. Dann berechnen wir die bedingte Wahrscheinlichkeit $P(m_1|m_2, ..., m_i)$ wie folgt:

$$(386)\quad P(m_1|m_2, ..., m_i) = \frac{P(m_1, m_2, ..., m_i)}{P(m_2, ..., m_i)}$$

dasselbe für andere bedingten Wahrscheinlichkeiten. $P(m_2, ..., m_i)$, $P(m_2)$ etc. sind wiederum marginale Wahrscheinlichkeiten, die man nach den gewohnten Regeln berechnet:

$$(387)\quad P(m_1) = \sum_{m_2 \in M_2, ..., m_i \in M_i} P(m_1, m_2, ..., m_i)$$

Mit diesen beiden Regeln können wir also sämtliche bedingten und unbedingten (marginalen) Wahrscheinlichkeiten ausrechnen, und dementsprechend auch ein minimales BN konstruieren. Es sollte aber klar sein, dass das mit einem erheblichen Aufwand verbunden ist, v.a. wenn es eine große Zahl von Variablen gibt.

Übung

Nehmen Sie folgende Graphen: $G1 = (\{1,2,3,4\}, \{(1,3),(2,3),(2,4),(3,4)\}$
$G2 = (\{1,2,3,4,5\}, \{(1,2),(1,3),(4,3),(4,5)\}$
$G3 = (\{1,2,3,4\}, \{(1,2),(1,3),(2,3),(4,3)\}$
Nehmen Sie, jeder dieser Graphen beschreibt ein bayesianisches Netz mit Variablen $\{X_1, ..., X_5\}$, die den Knoten wie offensichtlich zugeordnet sind. Sagen Sie, ob folgendes gilt, mit Begründung:

1. in $G1$: $X_1 \| X_4 | X_3$
2. in $G2$: $X_2 \| X_5 | X_3$
3. in $G3$: $X_1 \| X_4 | X_3$

Übung

Nehmen sie folgende multivariate Wahrscheinlichkeitsverteilung

$$P : M_1 \times M_2 \times M_3 \to [0,1],$$

wobei

$$M_1 = M_2 = M_3 = \{0,1\}$$

Und die Wahrscheinlichkeitsverteilung wie folgt ist (es gibt $2^3 = 8$ Ereignisse):

Ereignis	Wahrscheinlichkeit
(0,0,0)	0.252
(0,0,1)	0.096
(0,1,0)	0.252
(0,1,1)	0.054
(1,0,0)	0.028
(1,0,1)	0.024
(1,1,0)	0.168
(1,1,1)	0.126

Liefern Sie die ein minimales BN mit Variablen X_1, X_2, X_3, das diese Verteilung generiert!

Lösung: $X_2 \to X_1 \leftarrow X_3$
$P(X_2 = 1) = 0.6$
$P(X_3 = 1) = 0.3$

$P(X_1 = 0|X_2 = 0, X_3 = 0) = 0.9$
$P(X_1 = 0|X_2 = 0, X_3 = 1) = 0.8$
$P(X_1 = 0|X_2 = 1, X_3 = 0) = 0.6$
$P(X_1 = 0|X_2 = 1, X_3 = 1) = 0.3$

27.9 BN im Machine Learning

Der Begriff der konditionalen Unabhängigkeit ist auch im maschinellen Lernen nicht unwichtig. Praktisch jedes Modell macht Annahmen über die Unabhängigkeit seiner Eingaben uns Ausgaben, allerdings praktisch immer nur implizit. Es hilft aber oft zum Verständnis eines Modell oder zum Vergleich verschiedener Modelle, wenn man diese Faktoren explizit macht. Dazu weiter unten mehr. (?)

27.10 BN implementieren (im Aufbau)

Ein BN von Grund auf zu implementieren ist sehr mühsam. Es gibt aber fertige Pakete die genau das erlauben. Das berühmteste ist **bnlearn**. Wie der Name bereits sagt liegt der Fokus darauf, die Graphenstruktur eines Netzwerkes zu induzieren von rohen Daten. Wir haben bereits gesehen wie das geht in der letzten Aufgabe; in der Praxis ist das allerdings nicht so einfach: Wahrscheinlichkeiten sind ja nicht gleich Häufigkeiten – zwar nähern sich Häufigkeiten bei grossen Zahlen den Wahrscheinlichkeiten an, aber es gibt immer etwas **Rauschen**, also Störfaktoren. Anders gesagt: die Häufigkeiten, die von einem BN generiert werden, werden nicht 1:1 die Wahrscheinlichkeiten abbilden; andersrum wird dann natürlich auch gelten: um von den Häufigkeiten zum BN zu kommen, müssen wir eine gewisse Abweichungstoleranz zulassen.

> library(bnlearn)

28 Induktives Lernen

28.1 Der Rahmen

Was ist maschinelles Lernen? Man spricht von maschinellem Lernen, und das klingt gut. Der Begriff ist aber etwas irrefhrend. Die Maschine lernt nicht, so wie wir, ein vorgegebens Ziel (rechnen, Integralrechnung etc.). Die Maschine soll etwas "lernen", was wir selber nicht kennen. Und deswegen können wir auch nie sagen, ob sie es gelernt hat. Ich bitte Sie, das immer im Kopf zu behalten; man tendiert sonst dazu, die Maschine grandios zu überschätzen.

Es wäre also etwas genauer, wenn man stattdessen von *induktiver Inferenz* spricht. Das bedeutet soviel: wir möchten anhand von (endlich vielen) Beobachtungen einen Schluss machen auf eine zugrundeliegende Regel – ohne wirklich zu wissen ob diese Regel am Ende korrekt ist. Ein ML-Algorithmus verhält sich also nicht wie ein Schulkind, sondern eher wie ein empirischer Wissenschaftler.

Das klingt philosophisch, man kann das auch etwas nüchterner Beschreiben. **Maschinelles Lernen ist die Approximation von Funktionen**, anhand von ihren Instanzen. Nimm z.B. einen Datensatz

$$D \subseteq \mathbb{R} \times \mathbb{R} \times \mathbb{R}$$

Wir haben also z.B. $D = \{(2, 6.4, 3.92), (4.9, 3.57, 2.51), ...\}$. Es stellt sich erstmal die Frage: **was wollen wir daraus lernen**? Hier gibt es verschiedene Möglichkeiten. Eine Antwort wäre:

Die Aufgabe Wir möchten die dritte Komponente vorhersagen, gegeben die ersten beiden. Das bedeutet: wir möchten approximieren eine Funktion

$$f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

und zwar anhand einer Menge von Datenpunkten: $(2, 6.4, 3.92) \in D$ bedeutet dann soviel wie:

(388) $f(2, 6.4) = 3.92$ (bzw. $f(2, 6.4) \approx 3.92$)

Sprich: jeder Datenpunkt ist eine **Instanz** der Funktion. Wir versuchen die komplette Funktion die rekonstruieren, oder bzw. besser: eben zu approximieren (die Gründe hierfür sind verschieden: wir möchten nicht unbedingt,

dass unsere Rekonstruktion genau mit den Instanzen übereinstimmt, denn 1. kann das zu overfitting führen, und 2. haben wir oft Rauschen/Störungen in den Daten).

Also nochmal kompakt:

1. Wir nehmen einen Datensatz (eine endliche Relation, normalerweise)
2. Wir entscheiden welche Komponenten die Argumente der Funktion sind, welches die Werte
3. Auf diese Art bekommen wir die *Instanzen* einer Funktion, der Form $f(a_i) = b_i$
4. Diese zugrundeliegende Funktion versuchen wir dann, mit unseren Mitteln, zu approximieren.

Das nennt man maschinelles Lernen.

Klassifikation, Regression Maschinelles Lernen zerfällt am Anfang in zwei Teile, nämlich Klassifikation und Regression. (Es gibt noch etwas mehr, aber das wird erst später relevant). Man unterscheidet die beiden *einzig anhand des Wertebereichs der Zielfunktion*:

- Falls f eine stetige Funktion ist (z.B. $f : \mathbb{R} \to \mathbb{R}$), dann spricht man von *Regression*,
- falls f nur endlich viele Ausgaben hat, spricht man von **Klassifikation**.

Hypothesenraum Das Grundproblem ist dass wir f normalerweise nicht kennen, d.h. wir können nie wissen, ob unsere Induktion erfolgreich war oder nicht. Wir werden die Zielfunktion f auch niemals kennenlernen. Deswegen spielt f keine wichtige Rolle, und alles, was wir vorschlagen, sind immer nur **Hypothesen**. Wir nennen unsere Hypothesen h, und natürlich, falls wir eine Funktion $f : M \to N$ approximieren wollen, dann haben wir

$$h : M \to N$$

Alles was wir wissen können ist ob bzw. wie gut h übereinstimmt mit f auf dem endlichen Datensatz D von Instanzen, den wir zur Verfügung haben. Umso wichtiger ist dafür er der sogenannte

Hypothesenraum H,

d.i. eine Menge von möglichen Funktionen, aus der wir h auswählen. Der Raum **H** ist durchaus nicht vorgegeben im Rahmen des Induktionsproblems, und die Wahl ist oft alles andere als einfach. Der Hypothesenraum entspricht, etwas moderner, dem **Modell**, das wir wählen, um unser Problem anzugehen. Jedes Modell hat seine Beschränkungen und Eigenarten, und legt daher die möglichen Hypothesen *a priori* fest!

Eine Beispielregression Das kann man sehr schön am Beispiel einer *Regression* sehen. Nehmen wir an, wir möchten eine Funktion

$$f : \mathbb{R} \to \mathbb{R}$$

induzieren, z.B. um die Korrelation

$$\text{Tagestemparatur} \sim \text{Straftaten}$$

an einem gewissen Ort zu bestimmen (letztere gemittelt über einen längeren Zeitraum). Hier sieht man übrigens bereits, warum man nicht unbedingt will, dass f.a. Instanzen $f(a) = b$ auch gilt $h(a) = b$ – denn es handelt sich ja nur um Mittelwerte, und die Kriminalität wird ja von vielen anderen Faktoren beeinflusst, nur indirekt von der Temperatur.

Unser Datensatz D hat die Form

$$(10, 25.3), \text{ also } f(10) = 25.3$$
$$(14, 27.8), \text{ also } f(14) = 27.8$$
$$\text{etc.}$$

Da D eine Teilmenge des **Funktionsgraphen** von f ist, ist unsere Aufgabe ist wie folgt umrissen:

Finde eine Funktion $h \in H$, so dass für alle $(x, y) \in D$, $h(x) \approx y$.

Man nennt das die **Konsistenzbedingung**: die Hypothese soll möglichst konsistent mit den Daten sein – je konsistenter, desto besser, denn das ist unser einziges gute Auswahlkriterium.

Und jetzt die Frage: was ist H, unser Hypothesenraum? Hier gibt es folgende Überlegungen:

Je einfacher h ist, desto überzeugender würden wir es finden.

Z.B.: sei

$$h_1(x) = \frac{x}{2} + k \tag{389}$$

(ist einigermaßen konsistent mit den Datenpunkten). Das wäre sehr schön, und wir könnten sagen:

> Ein Anstieg von 2° Celsius bedeutet eine zusätzliche Straftat.

Wir könnten auch sagen: wenn es im August im Schnitt 25° wärmer ist als im Dezember, dann haben wir im Schnitt 12.5 Straftaten mehr pro Tag. Das wäre also eine sehr interessante Entdeckung!

Andererseits, es ist sehr unwahrscheinlich dass ein so komplexer, mittelbarer Zusammenhang so einfach ist, und so ist es sehr unwahrscheinlich, dass für alle $(x, y) \in D$ wir tatsächlich $h_1(x) = y$ haben. Es gibt sicherlich eine Funktion h_2, die in dieser Hinsicht wesentlich besser ist, z.B.

$$h_2(x) = x^5 + 6x^4 - 14x^3 + 15x^2 - 8x \tag{390}$$

Nehmen wir an, h_2 ist genauer auf D als h_1. Würden Sie sagen, dass h_2 plausibler ist? Eher nicht: wir würden sagen, dass die Komplexität von h_2 ein Anzeichen dafür ist, dass sie "maßgeschneidert" ist auf D und

schlecht generalisiert.

Das liegt v.a. daran, dass h_2 extrem komplex ist im Vergleich zu h_1. Selbst wenn sie wirklich ernsthaft adäquater wäre, wäre Sie deutlich schlechter, einfach weil sie viel uninformativer ist. Das besagt also:

> Wenn die richtige Hypothese darin ist, dann ist ein kleiner Hypothesenraum immer besser – sie ist dann einfach besser zu finden!

Wir treffen hier auf ein sehr grundlegendes Prinzip, nämlich das sog. **Rasiermesser von Ockham** (*Ockham's razor*), das besagt:

> *Die beste Hypothese aus einer Anzahl von Hypothesen die konsistent sind mit den Daten ist die einfachste.*

Allerdings sieht man bereits an unserem Beispiel, dass das eine sehr weiche Bedingung ist: denn h_2 passt besser als h_1, und es hängt nun alles davon ab, wie wir Konsistenz definieren. Es handelt sich also um eine weiche Richtlinie (die nichtsdestotrotz von grundlegender Bedeutung ist).

Wir können dieses Problem evtl. vermeiden, indem wir unseren Hypothesenraum *a priori* beschränken. Z.B. können wir sagen: uns interessieren nur die Polynome 2ten Grades, also Funktionen der Form

(391) $x^2 + ax + b$

Dabei gibt es folgendes zu beachten:

- Je kleiner der Hypothesenraum H, desto einfacher ist es, zwischen den konsistenten Hypothesen einen Kandidaten auszuwählen.
- Aber: je kleiner die Hypothesenraum, desto größer ist auch die Wahrscheinlichkeit, dass die korrekte Funktion gar nicht darin enthalten ist, also $f \notin H$.

Es gibt also Gründe die dafür und dagegen sprechen, H zu verkleinern. Wenn z.B. $f \notin H$, dann haben wir natürlich keine Möglichkeit, die korrekte Funktion zu induzieren. Da wir f nicht kennen, gibt es keine Möglichkeit, dass auszuschließen.

Nehmen wir z.B. an, die korrekte Korrelation (die wir natürlich nicht kennen) wäre

(392) $f(x) = ax + b + c\sin(x)$

das bedeutet: wir haben eine wachsende Wellenfunktion: Kriminalität erlebt bei steigenden Temperaturen immer wieder Scheitelpunkte.

- Solange wir also annehmen dass H aus Polynomialen besteht, werden wir niemals die richtige Funktion finden, sondern immer unmöglichere Polynomialfunktionen suchen müssen, solange wir mit neuen Daten konfrontiert werden!

Wir sehen also wie wichtig der richtige Hypothesenraum ist!

28.2 Test und Trainigsdaten, Overfitting und Underfitting

Es gibt für das ML zwei große Probleme, oder besser gesagt: zwei Arten, wie unsere Ergebnisse danebenliegen können, nämlich

- *overfitting* und
- *underfitting*.

Underfitting Dies ist normalerweise weniger problematisch. Underfitting bedeutet einfach: unser Hypothesenraum ist zu klein. Es gibt keine Funktion in H, die das zugrundeliegende Muster von f ausreichend fassen kann. Underfitting kann man daran beobachten, dass es auf den Trainingsdaten einen ziemlich hohen Fehler gibt (dazu später mehr), als es gibt viele a_i so dass $f(a_i) \not\approx h(a_i)$. Deswegen ist dieser Fehler gut sichtbar.

Overfitting Das ist ein großes Problem im maschinellen Lernen, das uns permament verfolgen wird, und das man nur mit ausgefeilten Methoden in den Griff bekommt. Overfitting kann man mit Üntergeneralisierung (!) übersetzen: wir generalisieren Dinge, die reiner Zufall sind. Anders gesagt: die Hypothese ist soz. maßgeschneidert für die Daten, und wird nicht ausreichend verallgemeinern und von irrelevanten Dingen abstrahieren.

Overfitting ist deswegen problematisch, weil man es nicht am Trainingsfehler sieht, ganz im Gegenteil: wenn das Modell overfittet, dann ist der Trainingsfehler sehr klein! Was man also macht ist folgendes:

- Man teilt den Datensatz in Test- und Trainingsdaten auf (20 zu 80)
- Man trainiert das Modell auf den Trainingsdaten, d.h. man setzt die Parameter des Modells
- Erst wenn das Training beendet ist, das Modell fertig "gelernt" hat, schaut man: wie groß ist der (durchschnittliche) Fehler auf den Testdaten?

Der entscheidende Punkt:

> Wenn der Fehler auf den Testdaten deutlich größer ist also auf den Trainingsdaten, dann haben wir wahrscheinlich overfittet!

Deswegen ist es im Maschinellen Lernen unerlässlich, Test- und Trainingsdaten zu trennen. Allgemein gilt folgendes:

- Je mächtiger eine Modellklasse (größer der Hypothesenraum), desto eher laufen wir ins Overfitting.
- Je mächtiger eine Modellklasse (größer der Hypothesenraum), desto geringer wird der Fehler auf den Trainingsdaten sein. Aber: auf den Testdaten nicht unbedingt! Die Trennung erlaubt uns also, einfachere Modelle zu rehabilitieren - vorausgesetzt, sie underfitten nicht.
- Underfitting bedeutet: der Fehler sowohl auf Test- als auch Trainingsdaten ist groß.
- Im allgemeinen gilt: die Größe des Verhaltnisses Parameter/Datenpunkte ist ein guter Indikator für das Risiko von Overfitting. Je kleiner der Wert ist, desto besser!

Ein Beispiel Man kann das gut anhand von einem Beispiel erklären: nehmen wir an, wir sehen eine Elster im Park ein Nest bauen. Daraus kann man verschiedene Generalisierungen ziehen:

(12) a. Elstern bauen im Park Nester.
b. Elstern bauen Nester.
c. Vögel bauen Nester.

In diesem Fall wäre a. eine Form von Overfitting: wir haben einen Parameter zuviel, nämlich den Ort, der keine Rolle spielt, und daher entgeht uns die richtige Generalisierung b. Umgekehrt ist c. eine falsche Generalisierung, denn wir haben einen Parameter zuwenig, die Vogelart, die wir brauchen um eine korrekte Generalisierung zu treffen. Es geht also darum, die richtige Zahl von Parametern zu finden (in unserem Beispiel: Faktoren die relevant sind), um die richtigen Generalisierungen zu finden. Dafür gibt es allerdings kein Patentrezept, wie wir auch im obigen Beispiel sehen: oft hilft es einfach nur, wenn wir domänenspezifisches Wissen haben, dass wir anwenden (mehr dazu gleich).

Besser kann man das aber anhand von Funktionsgraphen erklären (grob quadratische Korrelation, lineare Funktion underfittet, hohes Polynom overfittet).

Störparameter (Ein weiteres Problem hierbei ist, dass es oft Störparameter gibt, also Parameter, die nicht relevant sind für das was wir suchen. das können typischerweise Meßfehler sein, aber auch durchaus systematische Dinge: nehmen wir an, wir suchen den Erwartungswert einer Normalverteilung. Dann ist die Varianz dieser Verteilung ein Störparameter, da sie durchaus unsere Beobachtungen beeinflusst, aber keine Relevanz hat für das was wir suchen. Da es oft diesen Störparameter gibt, ist es auch nicht immer wichtig, dass unsere Modelle die Daten exakt reproduzieren: das wäre nämlich oft bereits eine Form von overfitting. Viel wichtiger ist, dass die Abweichung nicht systematisch ist, und das wir eben nicht zuviele Parameter haben.)

Noch zwei kleine Punkte zur Aufteilung der Daten. Dabei gibt es ein paar Dinge zu beachten.

1. Wir müssen verhindern, dass diese Aufteilung sich ändert, während unser Algorithmus nicht auf 0 gesetzt wird – sonst passiert genau dass, was man verhindern will: der Algorithmus sieht die Testdaten und kann auf ihnen Musterlösungen erzielen. Dann sind natürlich die gesamten Ergebnisse wertlos. Man muss also darauf achten dass die Aufteilung konstant ist. Da die Aufteilung zufällig erfolgt muss man die Zufallsparameter festsetzen.

2. Falls der Datensatz klein ist, muss man sichergehen, dass der Testsatz (20%) *repräsentativ* ist. Das heißt: alle Merkmale, die relevant sind, müssen auch vertreten sein. In diesem Fall müssen wir sicher sein, dass wir alle Merkmale vertreten. Das ist insbesondere schwierig bei kontinuierlichen Merkmalen: hier müssen wir kategorisieren, (sonst können wir keine sinnvollen Aufteilungen treffen; siehe Geron S.53)

29 Lineare Regression

29.1 Der einfache lineare Fall

Ein sehr wichtiges und einfaches Verfahren des maschinellen Lernens ist die lineare Regression. Hier wird versucht, eine Funktion mittels einer linearen Funktion zu approximieren. Etwas allgemeiner verwendet man lineare Regression für die Approximation mittels Polynomialfunktionen beliebigen Grades. Nehmen wir erstmal eine einfache lineare Funktion $f : \mathbb{R} \to \mathbb{R}$, die die Form haben soll:

$$f(x) = ax + b \tag{393}$$

wobei $a, b \in \mathbb{R}$ die Parameter sind. In diesem Fall muss unser Datensatz $D \subseteq \mathbb{R} \times \mathbb{R}$ einfach eine Abhängigkeit zweier reellwertiger Parameter darstellen. Wir setzen als Konvention $D = \{(a_1, b_1), ..., (a_n, b_n)\}$, und für $x = a_i$ schreiben wir y_x für b_i, also der Wert den D x zuweist. Wir suchen nun diejenige lineare Funktion, die die Differenz minimiert, also

$$argmin_{a,b\in\mathbb{R}} \sum_{(x,y_x)\in D} (ax + b - y_x)^2 \tag{394}$$

Das Quadrat ist dafür da, dass Werte positiv werden – sonst würden sich negative und positive Abweichungen ausgleichen. Damit gewichten wir natürlich weitere Abweichungen stärker, was nicht unbedingt erwünscht ist; allerdings gibt es kaum andere Möglichkeiten: die Betragsfunktion $|-|$ ist nicht differenzierbar, wir brauchen allerdings die erste Ableitung der Funktion, wie wir unten sehen werden.

Wir machen nun einen Trick: eigentlich sind die Parameter a, b festgelegt, während x das variable Argument der Funktion ist. Weil wir aber nur an denjenigen x interessiert sind, die in unserem Datensatz auftauchen (d.h. endlich viele), während wir alle reellen Parameter berücksichten müssen. Daher ist die Funktion, die wir minimieren müssen, eigentlich folgende:

$$\sum_{(a,b)\in D} (ax + y - b)^2 = (a_1x + y - b_1)^2 + ... + (a_n + y - b_n)^2 \tag{395}$$

Hier haben wir einfach die Konstanten und Variablen vertauscht, und daraufhin eine arithmetische Umformung vorgenommen. Am Ende bekommen

wir die einfache Form (denn $a_1, ..., a_n, b_1, ...b_n$ sind einfache gegebene Konstanten)

$$f(x,y) = ax^2 + by^2 + cxy + dx + ey + f \tag{396}$$

Denn alle Summanden haben eine dieser Variablenformen als Koeffizient, und wir suchen einfach

$$argmin_{x,y\in\mathbb{R}} ax^2 + by^2 + cxy + dx + ey + f \tag{397}$$

Das berechnet man mit der gewohnten Methode: wir bilden (in diesem Fall partielle) Ableitungen und konstruieren damit den Gradienten. Das ist natürlich besonders einfach:

$$\nabla f(x,y) = ((2ax + cy + d), (2by + cx + e)) \tag{398}$$

Wir haben also 2 Gleichungen, die wir auf 0 setzen müssen:

$$2ax + cy + d = 0 \tag{399}$$

$$2by + cx + e = 0 \tag{400}$$

Wir haben 2 Gleichungen und 2 Variablen, also eine Lösung:

$$x = -\frac{cy + d}{2a} \tag{401}$$

also

$$2by - \frac{c^2y + cd}{2a} + e = 0 \leftrightarrow \tag{402}$$

$$y = \frac{cd - 2ae}{4ab - c^2} \tag{403}$$

Wir haben also die Nullstelle für x, y berechnet, und wissen somit, wie wir die Funktion minimieren können.

29.2 Höhere Dimensionen

Nehmen wir an, die Eingabe ist nicht eine Zahl $x \in \mathbb{R}$, sondern ein Vektor $\vec{x} \in \mathbb{R}^n$. In diesem Fall müssen wir redefinieren was ein lineares Modell ist. Wir definieren eine lineare Funktion $f : \mathbb{R}^n \to \mathbb{R}$ als eine Funktion

(404) $f(x_1, ..., x_i) = a_1 x_1 + ... + a_i x_i + b$

In der schreibweise als Vektoren wird das noch viel einfacher schreiben als Skalarprodukt:

(405) $f(\vec{x}) = \vec{a}^\top \vec{x} + b$

Dieses Skalarprodukt lässt sich auffassen als ein Spezialfall von Matrixmultiplikation, wobei ein Vektor ein Spezialfall einer Matrix darstellt, und $M^\top$ die Transposition des Vektors. Das Problem der optimalen Parameter $\vec{a}, b$ lässt sich mit denselben Methoden analytisch berechnen. Allerdings gibt es hier ein Problem: mit wachsender Anzahl von Parametern wird das finden der optimalen Lösung zunehmend komplex. Man kann übrigens nicht nur die Dimension der Eingabe beliebig erweitern bei linearen Funktionen, sondern auch die **Dimension der Ausgabe**.

Das kann der Fall sein, wenn wir in einem Datensatz (z.B. Boston Housing) nicht nur ein Zielmerkmal vorhersagen möchten, sondern zwei:

> Z.B. die anderen Parameter sollen zusammem vorhersagen Latitude und Longitude, also die Position eines Distriktes

In diesem Fall sehen wir, dass lineare Funktion eigentlich als **Matrixmultiplikation** aufgefasst werden kann. Nimm eine Funktion $f : \mathbb{R}^n \to \mathbb{R}^m$. f ist linear falls

(406) $f(\vec{x}) = \mathbf{A}\vec{x} + \vec{b}$

Wir haben also eine Matrixmultiplikation, was soviel bedeutet: wir applizieren m unabhängige aber lineare Modelle auf $\vec{x}$, und jeder Wert davon bestimmt eine Komponente des Ausgabevektors.

Tatsächlich ist Matrixmultiplikation eng verknüpft mit linearer Regression. Das zeigt sich noch an anderer Stelle:

⇒ die optimalen Parameter, die wir in einer linearen Regression suchen, lassen sich nämlich über reine Matrizenoperationen ermitteln!

nehmen wir einen Datensatz

$$D \subseteq \mathbb{R}^n.$$

Das bedeutet, für die Regression suchen wir $n+1$ Parameter. Da der letzte Parameter mit 1 multipliziert wird, erweitern wir D zu D' indem wir in jeden Vektor eine 1 hinzuscheiben:

$$(v_1, .., v_n) \mapsto (v_1, ..., v_n, 1)$$

Als nächstes können wir D' transformieren in eine Matrix $\mathbf{A}_D$, die wir bekommen, indem wir alle Einträge von D untereinander schreiben. Also, für i Datenpunkt,

$$\{\vec{v_1}, \vec{v}_2, ..., \vec{v}_i\} \text{ wird zu } \textit{einer} \text{ Matrix } \begin{pmatrix} \vec{v_1}^\top \\ ... \\ \vec{v_i}^\top \end{pmatrix} = \mathbf{A}_D$$

Die (optimalen) Parameter der linearen Regression bekommen wir nun durch folgende Gleichung:

(407) $\theta = (\mathbf{A}_D^\top \cdot \mathbf{A}_D)^{-1} \cdot \mathbf{A}_D^\top \cdot \vec{y}$

wobei

- θ der Parameter-Vektor ist, den wir suchen,
- $\vec{y}$ der Vektor der Zielwerte
- $\mathbf{A}_D$ die Matrix der Daten (die Datentabelle gelesen als Matrix)
- $[-]^\top$ die Transposition, $[-]^{-1}$ die Inversion.

Man nennt diese Gleichung die **Normalgleichung** der linearen Regression. Ihre Herleitung ist aber ziemlich kompliziert!

Diese Normalgleichung wird auch benutzt, um die Komplexität der Regression zu bestimmen. Im allgemeinen gilt:

$\Rightarrow$ um eine lineare Regression von n Parametern durchzuführen, braucht man $O(n^{2.4})$ Rechenschritte.

D.h. bei einer überschaubaren Anzahl von Parametern ist das durchaus machbar; bei einer sehr großen Zahl kann es aber zu Problemen kommen. Insbesondere in der polynomialen Regression kann die Anzahl der Parameter schnell explodieren.

29.3 Polynomiale Regression

Manchmal bezeichnet man auch die Regression mittels Polynomen als lineare Regression. Der Grund hierfür ist, dass die Methoden im Prinzip dieselben sind, nur dass das ganze etwas komplizierter ist (mehr Parameter, höhere Exponenten). Im polynomialen Fall nehmen wir an, dass

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0 \tag{408}$$

f ist also nun ein Polynom, keine lineare Funktion mehr. Warum ist das immer noch lineare Regression? Weil wir, bei der eigentlichen Regression, also der Suche nach den Parametern die die beste Funktion ausmachen, x als eine Konstante behandeln (wir setzen nämlich Datenpunkte ein, bekommen also einfach konstante Werte in $\mathbb{R}$), während die eigentlichen Variablen die Werte $a_0, ..., a_n$ sind. In diesen Werten ist die resultierende Funktion nach wie vor linear – wir haben also einen Fall der etwas komplexer ist als der vorhergehende, aber nach wie vor durch die Lösung linearer Gleichungssysteme lösbar ist. Denn die **Regressionsfunktion**, also die Funktion, deren Minimalstelle wir suchen, lautet:

$$f(x_0, ..., x_n) = \sum_{(a,b) \in D} (a^n x_n + a^{n-1} x_{n-1} + ... + x_0 - b)^2 \tag{409}$$

NB:

- Hier sind a^n etc. und b fixe reelle Zahlen,
- während die Variablen der Funktion nur linear auftreten!

Wir müssen nun

$$\nabla f : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1} \tag{410}$$

konstruieren, kriegen also ein lineares Gleichungssystem mit $n + 1$ Variablen und $n+1$ Gleichungen, das wir entsprechend lösen können. Polynomiale Regression lässt sich also ebenso mit elementaren mathematischen Methoden lösen, und das ist der große Vorteil dabei. Also merke:

$\Rightarrow$ Die höheren Exponenten wandern, bei der Transformation

$$\text{Grundfunktion} \mapsto \text{Regressionsfunktion}$$

von den Variablen in die Daten

$\Rightarrow$ Polynomiale Regression ist also **lineare Regression auf modifizierten Daten**!

Nochmal: polynomiale Regression *ist* lineare Regression, nur der Datensatz verändert sich:

$$(a, b) \in D \Longrightarrow (a, a^2, b) \in D'$$

Dann ergibt sich:

$$\text{Quadratische Regression auf } D \cong \text{Lineare Regression auf } D'$$

29.4 Probleme mit polynomialer Regression – Overfitting

Der Schritt von linearer zu polynomialer Regression kommt allerdings mit einem Problem:

- Für die lineare Regression ist overfitting meistens ausgeschlossen. Wir können also immer sicher sein, dass unsere Ergebnisse relevant sind, und eine Trennung in Test- und Trainingsset ist dabei relativ überflüssing. Manchmal nutzt man Regularisierung (s.u.).
- Polynomiale Regression ist ja nicht *eine* Regression, sondern eine **Familie**, nämlich für jeden Grad gibt es eine Regression. Dabei gilt folgendes:
  - ▷ eine Regression $n+1$ten Grades schneidet auf den Trainingsdaten immer besser ab als eine Regression $n$ten Grades.
  - ▷ und für *jeden* konsistenten endlichen Datensatz D gibt es einen Grad n so dass die polynomiale Regression $n$ten Grades einen Fehler von 0 hat. Damit ist natürlich klar dass die Gefahr der overfitting besteht, und man mit einem Testsatz nachkontrollieren muss.

29.5 Polynomiale Regression in höheren Dimensionen

Wir haben also gesehen:

- Der Schritt von linearer Regression zu polynomialer Regression ist eigentlich eher quantitativ, nicht qualitativ. Wir bleiben – in der eigentlichen Regression – bei linearen Funktionen die wir optimieren, wir haben nur mehr Parameter (siehe auch die Aufgabe im Notebook).
- Ebenso bei linearer Regression in höheren Dimensionen – alles bleibt im Prinzip wie gehabt, nur die Anzahl der Parameter ändert sich.

Polynomiale Regression in höheren Dimensionen ist also ebenfalls *im Prinzip* nur eine quantitativ erweiterte lineare Regression. Das Problem hierbei ist: die beiden Erweiterungen multiplizieren sich, und das sorgt wiederum dafr, dass wir eine Anzahl Parameter haben, die man nicht mehr ohne weiteres handhaben kann.

Betrachten wir z.B. ein Polynom zweiten Grades, und einen Eingabevektor der Länge 3. Dann bekommen wir:

$$f(x_1, x_2, x_3) = a_1 x_1^2 + a_2 x_2^2 + a_3 x_3^2 + a_4 x_1 + a_5 x_2 + a_6 x_3 + a_7 \tag{411}$$

Wir haben also

$$n \cdot m + 1 \tag{412}$$

Parameter, wobei

- n die Eingabedimension ist,
- m der Grad des Polynoms

Das multipliziert sich nochmal, wenn wir eine k-dimensionale Ausgabe haben: dann bekommen wir $(n \cdot m + 1) \cdot k$ Parameter!

29.6 Heuristische Optimierung: Gradientenbasierte Methoden

Falls wir eine große Anzahl Parameter haben, dann wir die analytische Optimierung evtl. schwierig bis unmöglich. Stattdessen nimmt man eine **heuristische Optimierung**, was soviel bedeutet: eine Methode, die uns gute und vielleicht sogar sehr gute Ergebnisse liefert, aber die keinesfalls garantiert, dass wir die optimale Lösung finden. Heuristische Optimierung ist übrigens für komplexe Modell wie neuronale Netze absolut Standard; analytische Optimierung funktioniert nur in relativ einfachen Modellen. Deswegen lohnt es sich in jeden Fall diese Methoden zu besprechen, auch wenn sie in linearer Regression eher eine Ausnahme bleiben (sehr viele Parameter).

Moderne Heuristiken zur numerischen Optimierung basieren sich meistens auf den Gradienten. Wir erinnern uns:

- Der Gradient ist die partielle Ableitung einer multivariaten Funktion nach sämtlichen Variablen.
- Damit liefert er uns einen Ausgabevektor für jede Eingabe, und dieser Vektor zeigt in die Richtung des steilsten Anstiegs an diesem Punkt der Eingabe, für die infinitesimale Änderung.
- Die Nullstelle des Gradienten liefert uns also (hoffentlich!) eine Minimalstelle der Funktion.

Oftmals ist es aber unmöglich eine Nullstelle zu berechnen. In diesem Fall nutzt man die Methode des *gradient descent*:

- wir initialisieren die Parameter der Zielfunktion (= Argumente der Regressionsfunktion) zufällig.
- Wir berechnen den Gradienten der Regressionsfunktion an diesem einen Punkt.
- Dieser Gradient zeigt uns, in welche Richtung wir die Parameter ändern müssen, damit die Kosten am steilsten ansteigen.
- Wir wollen natürlich das Gegenteil: Kosten senken. Also ändern wir die Parameter in der genau entgegengesetzten Richtung.

Das ist natürlich etwas grob skizziert. Insbesondere gibt das nicht an, *wie weit* wir den unsere Parameter ändern; der Vektor gibt ja erstmal nur eine Richtung an. Dieser Parameter des Verfahrens wird als **Lernrate** bezeichnet; nennen wir ihn l. Hiermit können wir das Verfahren schon einigermaßen präzise beschreiben:

(413) $\theta' = \theta - \nabla f(\theta) \cdot l$

Das bedeutet in Worten: die neuen Parameter θ' sind die Alten, plus ihren Gradienten in der Regressionsfunktion, gewichtet mit l. Man kann sich also l als den Parameter vorstellen, der die Schrittgröße festlegt, mit der wir Parameter ändern.

Das Problem ist:

- Wählen wir die Rate zu klein, dann brauchen wir sehr lange, um gute Parameter zu finden;
- wählen wir die Rate zu groß, dann stapfen wir über die besten Werte hinweg.

In jedem Fall gilt aber: gradient descent garantiert nicht, dass wir eine optimale Lösung finden; wir können immer in lokalen Minima stecken bleiben; und manchmal können wir sogar diese übersehen. Soweit das Prinzip.

Gradienten und Matrizen Übrigens lässt sich der Gradient einer Funktion auch mittels Matrizenoperationen berechnen: eine weitere seltsame Verknüpfung von linearer Algebra und Analysis.

- $RF(\theta)$ ist die Regressionsfunktion
- mit Parametern $\theta = (\theta_0, ..., \theta_n)$ als Argument;
- $\vec{y}$ ist der Vektor der Zielwerte
- $m = |D|$ ist die Größe des Datensatzes.
- $\mathbf{A}_D$ ist die Matrix mit den Eingabedaten (mit einer Spalte 1 am Ende, fr die Basis)

(414) $\nabla RF(\theta) = (\frac{df}{d\theta_0} RF(\theta), ..., \frac{df}{d\theta_n} RF(\theta)) = \frac{2}{m} \mathbf{A}_D^T \cdot (\mathbf{A}_D \cdot \theta - \vec{y})$

29.7 Lernrate – flexibel

Die Frage nach der richtigen Lernrate lässt sich schwer allgemeingültig beantworten; welche Rate am besten ist, hängt vom konkreten Problem ab. Was aber im allgemeinen sinnvoll ist, ist

- die Lernrate in jedem Schritt geringfügig zu verringern:
- am Anfang machen wir am besten große Schritte, damit wir möglichst in die Nähe des globalen Minimums kommen;
- später, wenn wir bereits in der Nähe sind, ist es sinnvoll kleinere Schritte zu machen, damit wir möglichst genau das Minimum treffen.

die Lernrate in jedem Schritt geringfügig zu verringern: am Anfang machen wir am besten große Schritte, damit wir möglichst in die Nähe des globalen Minimums kommen; später, wenn wir bereits in der Nähe sind, ist es sinnvoll kleinere Schritte zu machen, damit wir möglichst genau das Minimum treffen. Man kann also eine flexible Rate definieren, z.B.

$$l = \frac{1}{s}, \tag{415}$$

s die Anzahl der Schritte die wir gemacht haben. Diese Lernrate sinkt am Anfang sehr steil und konvergiert relativ schnell. Um die Kurve etwas flacher zu machen kann man folgendes machen: man nimmt zwei (Hyper)Parameter l_1, l_2, normalerweise $l_1 < l_2$, und setzt

$$l = \frac{l_1}{s + l_2} \tag{416}$$

Je größer l_1, l_2 gewählt sind, desto flacher die Kurve, wobei man jeweils den Ausgangspunkt nicht ändern muss.

29.8 Batch gradient descent

In der Regressionsfunktion

$$\sum_{(a,b)\in D} (ax + y - b)^2 \tag{417}$$

steckt nach Definition ja der gesamte Datensatz. Wir berechnen als, im normalen *gradient descent*, den Gradienten immer über dem gesamten Datensatz aus.

Eine **batch** ist soviel wie ein Ausschnitt des Datensatzes. **Batch gradient descent** bedeutet dementsprechend soviel wie: wir berechnen den Gradienten nicht der Regressionsfunktion über den gesamten Datensatz (417, sondern den Gradienten der Regressionsfunktion über die *batch* $B \subsetneq D$!

$$\sum_{(a,b)\in B} (ax + y - b)^2 \tag{418}$$

Das bedeutet: wir arbeiten nur mit einem Ausschnitt der Daten! Das hat folgende Vorteile:

- Die Berechnung ist einfacher (offensichtlich), insbesondere bei großen Datenmengen
- Die Auswahl der batch fügt ein randomisiertes Element hinzu – sowas kann uns über lokale Minima hinweghelfen. Wie wird das gemacht?
- Ganz einfach: in jedem Schritt nehmen wir eine neue batch (zufällig gewählt), die uns in eine etwas andere Richtung lenken wird!

Der letzte Punkt bringt uns zum nächsten, nämlich: stochastic gradient descent.

29.9 Stochastic gradient descent

In SGD wird die Regressionsfunktion über den gesamten Datensatz nicht verwendet. Stattdessen formt man die Zielfunktion f um in die Regressionsfunktion mittels **eines einzigen** zufällig ausgewählten Datenpunktes $(a, b) \in D$.

(419) $(ax + y - b)^2$

In jedem Schritt wird ein neuer Datenpunkt gewählt. Da dieser Datenpunkt zufällig ausgewählt wird, ist SGD nicht-deterministisch. Außerdem gilt: da jeder Datenpunkt die Parameter in eine mehr oder weniger leicht andere Richtung schickt, ist es eher unwahrscheinlich dass wir jemals konvergieren. Stattdessen hüpfen die Parameter mit jeder neuen Runde leicht umher. SGD hat zwei entscheidende Vorteile:

- Die Berechnung der einzelnen Schritte geht deutlich einfacher, denn wir setzen ja nur einen Datenpunkt ein.
- Sie stochastische Komponente hat auch einen Vorteil: die Gefahr bei Gradient descent ist dass wir in lokalen Minima verloren gehen. Dadurch dass bei SGD die Parameter immer leicht herumhüpfen, kann es sein dass wir leichter aus einem lokalen Minimum wieder herausfinden.

Übrigens sind Stochastic gradient descent und Batch gradient descent im Prinzip zusammengehörig: SGD ist BGD mit einer batch-size von 1, und BGD mit einer kleinen batch ist im Prinzip SGD. Die beiden Methoden gehören also im Prinzip zusammen.

29.10 Lineare Regression in R

In R gibt es ein einfaches Kommando zur (einfachen) lineaeren Regression, nämlich `lm`. Erstmal brauchen wir Daten; dazu nehmen wir eine Menge $D \subseteq \mathbb{R}^2$. In R geht das einfacher, wenn man zwei Vektoren nimmt:

$$v_1 = (x_1, ..., x_n),\ v_2 = (y_1, ..., y_n), \text{ wobei } D = \{(x_1, y_1), ..., (x_n, y_n)\}$$

Denn hiermit kann man eine einfache Funktion nutzen:

```
> x <- c(-4,-2.8,-1.5,0,0.7,1.9,2.3)
> y <- c(7,4.3,5.2,3.8,3.6,2,0.8)
> plot(x,y,xlim = c(-5,5),ylim=(-7,7))
```

Hier werden also die Werte von `x` gegen `y` geplottet, wobei die Zuordnung anhand der Stelle im Vektor erfolgt. Nun möchten wir diese Funktion linear approximieren, also mittels eines Modelles

(420) $f(x) = ax + b$

Das geht in R ganz einfach mit dem Befehl:

```
> lm1<- lm(y ~ x)
```

Wir haben hier das Modell lm1, und das soll y als linear abhängige Variable von x vorhersagen.

```
> lm1
Call:
lm(formula = y   x)

Coefficients:
(Intercept)         x
3.4308 -0.7896
```

Wir bekommen also den *intercept*, eine konstante die uns sagt wo die Funktion die y-Achse kreuzt, und den *slope*, also die Steigung, der die lineare Anhängigkeit von x liefert; wir haben also:

(421) $lm1(x) = -0.7896x + 3.4308$

Das ist also die beste lineare Approximation unserer Daten. Wie gut sie ist kann man mit der Funktion `residuals` sehen, die jeweils liefert:

(422) $residuals(f) = (f(x_1) - y_1, ..., f(x_n) - y_n)$

In unserem Fall also:

> residuals(lm1)
1 ...
0.41079392 ...

Man kann die Regressionsfunktion auch schön visualisieren, mittels:

> abline(lm1, col = "red")

Um Modelle höherer Ordnung in R zu bekommen, gibt es meines Wissens nach keinen direkten Befehl. Man kann das aber recht einfach machen. Zunächst muss man wissen, wie man eine Variable x in Abhängigkeit mehrerer Variablen setzt. Das geht ganz einfach:

> lm2 <- lm(y ~ x+z)

Jetzt müssen wir nur noch den passenden Vektor z erstellen:

> z <- 1:7
> for(i in 1:7)z[i]=(x[i])^2
> lm2 <- lm(y+ ~ x + z)

In diesem Fall bekommen wir eine Polynom zweiter Ordnung für unsere Funktion; wir können die Ordnung weiter erhöhen, bis unsere Residuen bei 0 liegen werden. Die große Frage ist: wird unser Modell dadurch besser?

30 Regularisierung

Regularisierung bedeutet: wir nehmen noch weitere Terme in die Optimierung auf, um die Parameter in eine gewisse Richtung zu lenken.

- Beispielsweise könnte es sein, dass wir Grund zu der Annahme haben, dass kleine Parameter besser passen, also besser generalisieren auf ein Problem.
- Im Falle einer linearen Regression

  (423) $f(x) = ax + b$

  liefert a ein Maß der **Korrelation**; das bedeutet:
- ein kleineres a wäre konservativer, vorsichtiger, weniger anfälliger für overfitting.
- Es macht also Sinn, ein großes a *per se* mit Kosten zu belegen!

Im Allgemeinen dient Regularisierung also dazu, overfitting zu vermeiden. Regularisierung kann auch zum Einsatz kommen, wenn wir zuviele Parameter und zuwenige Daten haben, so dass wir nicht nur die Gefahr von overfitting haben, sondern unsere Gleichungen gar keine eindeutige Lösung bestimmen.

Zu diesem Zweck fügen wir einen neuen Term in die Funktion ein, die wir minimieren möchten. Man unterscheidet verschiedene Methoden der Regularisierung je nach der Form, die dieser Term annimmt.

30.1 Tikhonov Regularisierung/Ridge Regression

Sei

$$\vec{\theta} = (\theta_0, ..., \theta_n)$$

der Vektor der Parameter. Also bei

(424) $f(x_1, ..., x_n) = a_n x_n + ... + a_1 x_1 + a_0$

wäre $\vec{\theta} = (a_0, ..., a_n)$. Wir schreiben dann $a_i = \theta_i$. Außerdem sei

$$FReg$$

die **Regressionsfunktion** dieser Parameter, also die Funktion, deren Minimalstelle wir suchen. Dann definieren wir:

(425) $TR(FReg)(\theta) = FReg(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$

Wir fügen also einen **Kostenterm** hinzu, der die Parameter θ_i *selbst* bestraft, unabhängig von der Vorhersage, also vom Fehler. α ist dabei ein Hyperparameter. Wichtig ist hier:

- es werden große Parameter grundsätzlich bestraft (als Kosten behandelt)
- nur der konstante Parameter θ_0 (intercept) wird nicht bestraft (enspricht keiner Korrelation)
- je größer der Hyperparameter α gewählt wird, desto mehr werden große Parameter bestraft, also, anders gesagt, desto flacher wird die Kurve der resultierenden Funktion.
- Die Rolle des Quadrates ist dreifach:
    1. Werte werden positiv
    2. Werte > 1 werden größer, also werden starke Korrelationen stark regularisiert
    3. Werte < 1 werden *kleiner*, also werden schwache Korrelation nur schwach regularisiert!

Regularisierung bringt uns also dazu, konservativ zu generalisieren (also vorsichtig).

Ein Vorteil von Tikhonov Regularisierung ist, dass sich die optimalen Parameter ebenfalls analytisch bestimmen lassen, also die mathematische Komplexität nicht ernsthaft zunimmt.

30.2 Lasso Regression

Lasso Regression heißt – glaube ich – tatsächlich so, weil das "Lasso" zu große Parameter wieder einfangen soll. Der Regularisierungsterm ist hier die Betragsfunktion:

$$LR(FReg)(\theta) = FReg(\theta) + \alpha \sum_{i=1}^{n} |\theta_i| \tag{426}$$

Lasso Regression hat eine wichtige Eigenschaft:

- es tendiert dazu, alle unwichtigen Merkmale (=Teilpolynome) komplett zu eliminieren.
- Denn der Betrag ist schärfer auf $\theta_i < 1$ als das Quadrat!

LR macht also eine Art von **Merkmalsauswahl**, die eben darin besteht, dass nur sehr wichtige Merkmale übrig bleiben. So kann man nach einer Lasso Regression z.B. sagen:

> Die Abhängigkeit ist ganz klar quadratisch, oder klar kubisch etc.

auch wenn eine polynomiale Regression kein derart eindeutiges Ergebnis liefert.

Wie man am Term sieht ist LR **nicht ableitbar**, also nicht analytisch optimierbar. Man braucht also in jedem Fall ein heuristisches Optimierungsverfahren, wie etwa **gradient descent**.

30.3 Frühes Aufhöhren

Insbesondere bei heuristischer Optimierung ist frühes Aufhören oft eine gute Methode, um overfitting zu vermeiden. Man möchte also vermeiden, dass unser Algorithmus *zu gut* auf die Trainingsdaten fittet. Um hier einen guten Punkt zu finden, muss man eine etwas komplexere Methode wählen:

- Wir wählen ein (nicht zu großes) n; insbesondere sollte es deutlich kleiner sein als die erwartete Anzahl von Iterationen. Wir können auch $n = 1$ setzen.
- Nach allen n Iterationen prüfen wir, wie gut unser Modell auf Trainings- und Testdaten abschneidet.
- Der Fit auf den Trainingsdaten wird am Anfang schnell, dann immer langsamer besser werden. Der Fit auf den Testdaten wird anfangs besser, aber oftmals ab einem gewissen Punkt $n*$ wieder langsam *schlechter*.
- Genau dieser Punkt markiert die optimale Anzahl der Iterationen; danach geht es tendenziell ins overfitting. Wir wählen also die Parameter nach $n*$ Iterationen als optimale Parameter aus.

Allerdings ist an dieser Stelle Vorsicht geboten: wir können uns nicht hinstellen und sagen: das beste abschneiden auf dem Testset ist die Performance unseres Systems auf dem Testset. Das wäre eindeutig Betrug, denn wir haben ja das Testset benutzt, um die Anzahl der Iterationen $n*$ zu bestimmen. Also haben wir die Daten im Hinblick auf das Testset optimiert. Wenn wir objektiv gültige Ergebnisse haben wollen, dann brauchen wir

1. Trainingsset (zur Optimierungsmethode)
2. Developmentset (zum iterierten testen, um $n*$ zu bestimmen)
3. Ein Testset, dass erst ganz zuletzt mit den Parametern nach $n*$ Iterationen gestestet wird.

31 Logistische Regression – Aktivierungsfunktionen

31.1 Definitionen

Logistische Regression ist im engeren Sinne keine Regression, sondern Klassifikation; es ist aber eine Klassifikation, die auf linearer Regression aufbaut. Logistische Regression funktioniert im einfachen Fall, wenn wir 2 Klassen haben, zwischen denen wir auswählen, z.B. A und B. Unser Problem ist also folgendes: wir möchten eine Eingabe in $\mathbb{R}$ nach A oder B klassifizieren. Wir machen das mittels der logistischen Sigmoid-Funktion

$$S(x) = \frac{1}{1 + e^{-x}} \tag{427}$$

Einige Beobachtungen: wir haben

- $lim_{x \to -\infty} S(x) = 0$
- $lim_{x \to \infty} S(x) = 1$
- $S(x) \in (0, 1)$ f.a. $x \in \mathbb{R}$
- $S(x)$ ist streng monoton steigend in x

Allgemeiner nennt man solche Funktionen **Sigmoidfunktionen**. Andere Beispiele für solche Funktionen sind:

$$T(x) = \frac{x}{1 + |x|} \tag{428}$$

$$U(x) = \frac{x}{1 + x^2} \tag{429}$$

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 1 - \frac{2}{e^{2x} + 1} \tag{430}$$

Was allen diesen Funktionen gemeinsam ist, ist dass sie im Bereich um 0 relativ schnell von 0 Richtung 1 wechseln, sonst für große positive/negative Zahlen langsam Richtung 1/0 konvergieren. Es gibt also nur einen kleinen Bereich, in dem eine nicht extreme Änderung im Argument eine signifikante Änderung im Wert verursacht; ansonsten gibt es relativ schnell eine Sättigung.

solche Prozesse findet man oft in der Natur, z.B. bei Populationen. Solche logistischen Funktionen haben allerdings die Einschränkung dass sie nur $\mathbb{R} \to \mathbb{R}$ definiert sind. Aktivierungsfunktionen spielen in neuronalen Netzen eine große Rolle, und werden normalerweise auf die einzelnen Komponenten von Vektoren angewendet.

Interessant zu wissen ist auch folgendes: das unbestimmte Integral der **Normalverteilung** ist ebenfalls eine Aktivierungsfunktion, also (sei $dnorm(-|0,\sigma)$ die Normalverteilung mit Erwartungswert 0 und Standardabweichung σ):

$$\int_{-\infty}^{x} dnorm(x|0,\sigma)dx \tag{431}$$

hat eine vergleichbare Kurve – wobei je grßer σ (Streuung), desto flacher ist sie. Das bedeutet natürlich im Umkehrschluss auch, dass die Ableitung einer (positiven) Aktivierungsfunktion eine Glockenkurve bildet!

Wenn wir nun 2 Klassen haben, dann können wir einfach sagen: wir **interpretieren** den Wert als eine **Wahrscheinlichkeit**, also:

$$P(X = A|x) = S(f(x)) \tag{432}$$

wobei f eine lineare Funktion ist, die wir mittels linearer Regression induzieren. Wir transformieren also eine Funktion in die reellen Zahlen zu einer Funktion nach $[0,1]$:

$$f : \mathbb{R} \to \mathbb{R} \Longrightarrow S \circ f : \mathbb{R} \to [0,1]$$

Das Ergebnis können wir dann als Wahrscheinlichkeit interpretieren. Das gibt uns nun einen **Klassifikator**:

$$C(x) = \begin{cases} A, \text{ falls } P(X = A|x) > 0.5 \\ B \text{ andernfalls} \end{cases} \tag{433}$$

Das funktioniert natürlich nur, falls wir nur zwei Klassen (hier A, B) zur Verfügung haben. Intuitiv bilden wir hier die Eingabe auf eine Wahrscheinlichkeit ab, und Klassifizieren nach Maximum Likelihood. Natürlich kann auch das sehr leicht generalisiert werden auf beliebige Eingaben

$$\mathbf{x} \in \mathbb{R}^n.$$

mit

$$f : \mathbb{R}^n \to \mathbb{R} \Longrightarrow S \circ f : \mathbb{R}^n \to [0,1]$$

31.2 Bedeutung

Die Bedeutung der logistischen Regression ist erstmal folgende: wir suchen

a) eine Klassifikationsfunktion

$$C : \mathbb{R}^n \to \{0, 1\}$$

b) eine Wahrscheinlichkeitsverteilung

$$P : \mathbb{R}^n \to [0, 1]$$

Wir tun das aber mittels zweier Zwischenschritte:

$$\mathbb{R}^n \underset{f}{\Longrightarrow} \mathbb{R} \underset{S}{\Longrightarrow} [0, 1] \underset{C}{\Longrightarrow} \{0, 1\}$$

Die zugrundeliegende Annahme ist hierbei, dass es einen kritischen Bereich gibt, auf dem die Klassifikation von 0 zu 1 springt und umgekehrt ist; aber damit man das sieht, muss man zunächst eine **numerische Transformation** durchführen, nämlich die lineare Regression. Genauer gesagt, die lineare Funktion f kann die numerischen Werte so transformieren, dass gilt (im eindimensioneln Fall!):

$$\text{Falls } f(x) > s, \text{ dann } C(x) = 1, \text{ und falls } f(x) \leq s, \text{ dann } C(x) = 0$$

Das bedeutet, wir haben eine *quasi-lineare Abhängigkeit* von unseren Eingabedaten und den Klassen. Das kann man noch genauer fassen, es gibt nämlich eine genaue Beschreibung wenn wir den **Hyperparameter** des **Grades der Funktion** mitberücksichtigen. Nehmen wir an, wir suchen eine Funktion $f : \mathbb{R} \to \{0, 1\}$. Wir wissen, dass

- Eine lineare Funktion ist hat eine Gerade als Graphen, kann also einen gewissen Wert nur einmal annehmen
- Ein Polynom zweiten Grades hat einen Wendepunkt, kann einen gewissen Wert höchstens zweimal annehmen.
- ...
- Ein Polynom $n$ten Grades hat $n-1$ Wendepunkte, kann einen gewissen Wert höchstens n-mal annehmen.

Dann heißt das soviel wie:

- Falls wir eine lineare Funktion haben, dann gibt ein $\alpha \in \mathbb{R}$, so dass falls $x < \alpha$, dann $f(x) = 1$ (bzw. 0), ansonsten $f(x) = 0$ (bzw. 1). Wir spalten also den Eingaberaum $\mathbb{R}$ in zwei Teile.
- Falls wir ein Polynom zweiten Grades haben, dann gibt $\alpha_1, \alpha_2 \in \mathbb{R}$, so dass falls $\alpha_1 < x\alpha_2$, dann $f(x) = 1$ (bzw. 0), ansonsten $f(x) = 0$ (bzw. 1). Wir spalten also den Eingaberaum $\mathbb{R}$ in drei Teile.
- ...
- Falls wir ein Polynom n-ten Grades haben, haben wir $\alpha_1, ..., \alpha_n$, die $\mathbb{R}$ in $n + 1$ Teile spalten, und wir klassifizieren auf dieser Basis.

Der Fall in höheren Dimensionen Das ganze lässt sich gut in höhere Dimensionen verallgemeinern. **Hyperebenen** stellen eine Generalisierung von Ebenen (im 3-dimensionalen Raum) auf beliebige Dimensionen dar. Eine normale Ebene ist spezifiert durch einen **Stützvektor** $\vec{s}$ und zwei linear unabhängige **Richtungsvektoren** $\vec{r}_1, \vec{r}_2$. Wichtig ist: die Richtungsvektoren müssen

1. Ungleich 0_V sein (sonst haben sie keine Richtung!)
2. Sie müssen linear unabhängig sein – sonst ist die Ebene unterdeterminiert!

Ein Punkt p liegt auf der Ebene, falls er sich darstellen lässt als

$$p = \lambda_1\vec{r}_1 + \lambda_2\vec{r}_2 + \vec{s} \text{ für } \lambda_1, \lambda_2 \in \mathbb{R} \tag{434}$$

Diese Definition lässt sich leicht verallgemeinern: man nimmt, für einen Raum $\mathbb{R}^n$, einfach den Stützvektor $\vec{s} \in \mathbb{R}^n$ und ebenso $n - 1$ linear unabhängige Richtungsvektoren $\vec{r}_1, ..., \vec{r}_{n-1}$ (ungleich 0). Technisch gesehen ist die Hyperebene eine Menge von Punkten:

$$H = \{\vec{s} + \lambda_1\vec{r}_1 + ... + \lambda_{n-1}\vec{r}_{n-1} : \lambda_1, ..., \lambda_{n-1} \in \mathbb{R}\} \tag{435}$$

- Im Falle einer einfachen logistischen Regression auf $\mathbb{R}^n$ sind die Menge

  (436) $M_1 = \{\vec{x} \in \mathbb{R}^n : C(\vec{x}) = 1\}$

  und

  (437) $M_0 = \{\vec{x} \in \mathbb{R}^n : C(\vec{x}) = 0\}$

  durch eine Hyperebene getrennt.

- ...

- Im Falle einer logistischen Regression auf $\mathbb{R}^n$, basierend auf einer polynomiellen Regression $m$ten Grades, sind die Mengen M_1, M_0 durch m Hyperebenen separiert.

Man nennt diese Eigentschaft auch **lineare Separierbarkeit**.

Merke

- Eine logistische Regression ist eine Klassifikation auf Basis einer linearen Regression (polynomialle Regression ist ja auch lineare Regression).

- Die Aktivierungsfunktion spielt dabei für die eigentliche Klassifikation *keine nennenswerte* Rolle, da

$$S(f(\vec{x})) > 0.5 \text{ gdw. } f(\vec{x}) > 0.$$

- Die Aktivierungsfunktion ist aber entscheidend für die Optimierung der Funktion, und bei Bedarf für die Interpretation als Wahrscheinlichkeit.

31.3 Kostenfunktion für LogReg

Wie berechnen wir die Kosten von einer gegebenen Klassifizierungsfunktion, so dass wir die Parameter des linearen Teils optimieren können? Hier nimmt man die sog. log-loss Funktion, die jeweils die logarithmische Wahrscheinlichkeit des falsch liegens misst: Zur Erinnerung:

- Y ist eine Zufallsvariable, die den Wert 0 oder 1 annehmen kann, jeweils mit einer gewissen Wahrscheinlichkeit. Dabei ist diese Wahrscheinlichkeit durch die Funktion gegeben:

  $$P_\theta(Y = 1|\vec{x}) = S(f_\theta(\vec{x})) \quad (438)$$
  $$P_\theta(Y = 0|\vec{x}) = 1 - S(f_\theta(\vec{x})) \quad (439)$$

  f_θ die lineare Funktion mit θ als Parametern.

- Mit y, als "korrekter Datenpunkt", ist in $\{0, 1\}$.

- $m = |D|$ ist die Größe des Datensatzes.

Dann bekommen wir folgende Funktion:

Log-loss

$$K(\theta) = -\frac{1}{m} \sum_{(\vec{x},y)\in D} [y \cdot log P_\theta(Y = y|\vec{x}) + (1 - y) \cdot log(1 - P_\theta(Y = y))] \quad (440)$$

Wir nehmen also die Summe der beiden Abweichungen vom richtigen Wert, jeweils in beide Richtungen. Wenn wir genau richtig liegen würden, wäre der Term 0 bzw. undefiniert ($log\ 0 = -\infty$); da $S(x) \in (0, 1)$ kann das aber nicht passieren. Allgemein gilt:
Merke:

> Je näher $P_\theta(Y = y|\vec{x})$ an y liegt (über alle Datenpunkte), desto kleiner ist der Term $K(\theta)$.

Wir sehen also hier, dass probabilistische Konzepte eine entscheidende Rolle für das Training spielen. Ohne sie gibt es normalerweise keine Möglichkeit, Klassifizierung effektiv zu optimieren.

32 Softmax-Regression

Die logistische Regression liefert uns eine Funktion $f : \mathbb{R}^n \to [0, 1]$; wir haben gesagt, wir können diese Funktion auffassen als

1. Wahrscheinlichkeit – mit der Beschränkung dass wir nur die Wahrscheinlichkeit eines *einzigen* Ereignisses A bekommen.

2. Klassifizierer – mit der Beschränkung, dass wir nur binär klassifizieren können: nämlich entweder A (entspricht $P(A|x) > 0.5$) oder andernfalls $\overline{A}$.

Wir bleiben mit der Frage zurück:

> *Wie kann man eine Regression zum klassifizieren über eine beliebige endliche Menge C nutzen?*

Die **Softmax-Funktion** ist eine sehr beliebte Funktion, mit der man eine n-dimensionale Eingabe in eine diskrete Wahrscheinlichkeitsverteilung über n Ergebnisse transformieren kann. Eine solche Funktion ist insbesondere wichtig wenn wir eine logistische Regression zur Klassifizierung einsetzen möchten. Nehmen wir einmal folgendes Beispiel.

Beispiel Wir möchten eine Eingabe $\vec{x} \in \mathbb{R}^n$ als eine Klasse $c \in C$ klassifizieren, wobei $|C|$ endlich ist. $\vec{x}$ kann ein Bild sein (Pixeldaten) und C eine Menge mit Objekten, oder $\vec{x}$ kann (wie im Boston housing) ein Vektor mit verschiedenen Daten sein und C eine Klassifizierung wie "gute Lange", "mittlere Lage" etc. Im Deep Learning wird man sehen dass $\vec{x}$ auch die Repräsentation eines Wortes sein kann, C eine Menge von POS-tags; im modernen machine learning werden praktische alle Eingaben erstmal zu Vektoren transformiert.

Wir kann man das mit logistischer Regression modellieren? Den Fall $|C| = 2$ haben wir bereits gesehen; aber der lässt sich hier nicht verallgemeinern. Was man stattdessen macht ist folgendes: man trainiert ein lineares Modell

(441) $f : \mathbb{R}^n \to \mathbb{R}^{|C|}$

so dass Komponente der Ausgabe eine Kategorie C repräsentiert. Am konzeptuell einfachsten wäre es dann, einfach eine *argmax*-Funktion nachzuschalten:

(442) $max(x_1, ..., x_m) = max\{x_1, ..., x_m\}$

Wenn wir also unsere Klassen nummerieren, dann bekommen wir,

(443) $C(\vec{x}) = c_i$, wobei $i = argmax_i(\mathbf{A}(\vec{x}))_i$

Anders gesagt: i ist die größte Komponente.

Das Problem ist folgendes: die *max*-Funktion ist nicht differenzierbar. Weiterhin gilt: selbst wenn wir an einem Punkt einen Gradienten bekommen, dann haben wir wahrscheinlich eine Ebene – ein minimale Änderung in der Eingabe verursacht keine Änderung in der Ausgabe. Solche Plateaus können es schwierig bis unmöglich machen, ein Modell zu trainieren. Die Werte ändern sich ja sprunghaft, also wie sollen wir numerische Parameter trainieren?

Daher gibt es die sog. *softmax*-Funktion, die eine ähnliche Funktion übernimmt, aber differenzierbar ist (daher *soft*, weil sie keine Knicke hat). Diese Funktion spielt auch im Deep Learning eine sehr wichtige Rolle, und eigentlich immer, wenn (nicht-binäre) Klassifizierung mit numerischen Berechnungen durchgeführt werden soll. Es gilt:

$$softmax : \mathbb{R}^n \to \mathbb{R}^n$$

wobei

(444) $$softmax(x_1, ..., x_n)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Diese Notation ist gewählt um die Definition übersichtlicher zu machen; man kann auch äquivalent schreiben:

(445) $$softmax(x_1, ..., x_n) = (\frac{e^{x_1}}{\sum_{j=1}^{n} e^{x_j}}, ..., \frac{e^{x_n}}{\sum_{j=1}^{n} e^{x_j}})$$

Es gilt also, da wir durch die Summe der Komponenten dividieren:

(446) $$\sum(softmax(\vec{x})) = 1$$

Wir können den Vektor also als Wahrscheinlichkeitsfunktion lesen!

Wie gesagt: falls $\mathbf{x} \in \mathbb{R}^n$, dann ist auch $softmax(\mathbf{x}) \in \mathbb{R}^n$. Die softmax-Funktion ist eine Art "weiche" argmax-Funktion:

⇒ dadurch dass wir exponentieren bekommen alle Komponenten, die nicht unbedeutend kleiner sind als der Größte, eine vernachlässigbare Größe

Der große Vortei von softmax ist: es ist eine stetig differenzierbare Funktion. Das ist sehr wichtig für die Optimierung, für die wir Gradienten brauchen.

Beispiel !!!!!!!!!!!!

Optimierung von Softmax: Kreuzentropie Softmax-Funktionen werden üblicherweise als Ausgabefunktionen verwendet: sie liefern tatsächlich eine Art **konditionale Wahrscheinlichkeitsverteilung**:

$$P(c_i|\vec{x}) = softmax(\mathbf{A}(\vec{x}))_i \tag{447}$$

Am Ende nehmen wir natürlich wiederum die Komponente mit dem maximalen Wert. Wichtig ist dass die Funktion stetig und differenzierbar ist – andernfalls könnten wir nicht weiter optimieren, der Gradient gibt uns keinerlei Information.

Die Optimierung läuft (wie bei logistischer Regression) nicht auf die Ausgabe (diskret), sondern über die gesamte Wahrscheinlichkeitsverteilung. Hier nutzt man eine Kostenfunktion wie **categorial crossentropy** (Kreuzentropie), was in einer stetigen Kostenfunktion resultiert:

$$H(X, P, Q) = H(X_P) + D(P\|Q) = -\sum_{x\in\Omega} P(X = x) log Q(X = x) \tag{448}$$

wobei P die Verteilung unseres Systems ist, Q die "wahre Verteilung" (ergibt sich aus den "korrekten Vorhersagen". Das resultiert also in einer differenzierbaren Funktion die heuristisch (analytisch?) optimiert werden kann.

Wenn wir die obige Formel der Kreuzentropie auf eine Vorhersage und einen Datenpunkt applizieren, ist das viel einfacher als es aussieht:

- Sei $M : \mathbb{R}^n \to \mathbb{R}^i$ unser Machine Learning Modell (softmax regression).
- $M(\vec{x}) = \vec{y} \in \mathbb{R}^i$ ist ein Vektor, der eine Wahrscheinlichkeitsverteilung über i Klassen darstellt.
- Der "target"-Vektor bekommt dann die Form $\hat{y} \in \{0, 1\}^i$, wobei $|\hat{y}| = 1$ (*one-hot Vektor* sagt man auch dazu, entspricht einer Verteilung ohne Entropie).

Die Kreuzentropie der beiden Verteilungen reduziert sich dann auf

$$-log\ y^\top \hat{y} \tag{449}$$

Sei $\hat{y}_i = 1$. Wir bekommen dann (noch einfacher):

(450) $-log\ y^\top \hat{y} = -log(\vec{y}_i)$

als unsere Kosten auf einem Datenpunkt, nämlich:

$\Longrightarrow$ den negativen Logarithmus der vorhergesagten Wahrscheinlichkeit des korrekten labels.

Wir optimieren also ausschliesslich, indem wir die Wahrscheinlichkeit unserer Vorhersage für das korrekte Ziellabel erhöhen – alles andere spielt keine Rolle!

Auf dem gesamten Datensatz betrachtet:

(451) $-\sum_{(\vec{x},\hat{y})\in D} log\ M(\vec{x})^\top \hat{y}$

Softmax-Regression: Überblick !!!

Nachtrag: softmax und binäre Klassifikation Man kann sich nun fragen, inwieweit sich eine *softmax*-Regression von einer logistischen Regression unterscheidet im Fall dass wir binär klassifizieren. Die Antwort lautet: es gibt einen Unterschied, allerdings nur im Detail: wenn wir

(452) $softmax(x_1, x_2)$

berechnen, dann entspricht dass einer Aktivierungsfunktion, man muss nur die Argumente umstellen. Es gilt im Allgemeinen:

(453) $exp(x + a) = a \cdot exp(x)$

also

(454) $exp(x + a)/exp(y + a) = exp(x)/exp(y)$

Hieraus und aus der Definition von Softmax folgt der allgemeine Verschiebungssatz:

(455) $softmax(x_1, x_2) = softmax(x_1 + a, x_2 + a)$

für $a \in \mathbb{R}$. Das bedeutet also, dass die folgenden Funktionen A eindeutig definiert sind:

$$A(x_1 - x_2) := softmax(x_1, x_2)_1 \tag{456}$$

$$A(x_2 - x_1) := softmax(x_1, x_2)_2 = \tag{457}$$

Es ist nun leicht zu prüfen, dass A eine Aktivierungsfunktion mit den oben beschriebenen Charakteristika ist. Wir haben also folgende Korrespondenz:

Binärer Softmax $\cong$ Aktivierungsfunktion über Differenz der Argumente

32.1 Die Interpretation der logistischen Regression – ein Beispiel

Wir betrachten folgendes Beispiel: wir möchten, als abhängige, diskrete Variable vorhersagen, ob ein Student die Prüfung besteht (0 oder 1; abhängig bedeutet nur: wir möchten das vorhersagen). Als Prädiktor nutzen wir die Anzahl der Stunden, die der Student gelernt hat. Unsere Regressionsfunktion soll eine einfache lineare Funktion sein. Unser Datensatz sieht nun wie folgt aus (nennen wir den Datensatz $D1$):

Stunden gelernt	4	5	6	7	8	9	10
Bestanden	0	0	1	0	1	1	1

Eine lineare Regression liefert hier folgendes Ergebnis (danke R):

(458) $f(x) = 0.1786x - 0.6786$

Wir haben also eine positive Abhängigkeit von Lern-Stunden und Klausur-Bestehen (zum Glück); aber dieses Ergebnis ist noch unbefriedigend: wir können das nicht sinnvoll als Wahrscheinlichkeit interpretieren. Wir setzen aber nun diesen Term ein in unsere Aktivierungsfunktion, und defineren:

(459) $P(\text{bestehen}|n \text{ Stunden lernen}) = \frac{1}{1+e^{-f(x)}} = \frac{1}{1+e^{-(0.1786x-0.6786)}}$

Das liefert uns nun ordentlich Werte:

Stunden	4	5	6	7	8	9	10
P(bestehen)	0.508	0.553	0.597	0.639	0.679	0.717	0.752

Das ist schon besser, aber nicht wirklich optimal, insbesondere stört die Asymmetrie (bei 4 haben wir bereits eine ziemlich hohe Wahrscheinlichkeit!). Die logistische Funktion interagiert mit der linearen auf eine Art und Weise, die nicht optimal ist. Wir sollten also stattdessen folgendes suchen:

Logistische Regressionsfunktion

(460) $argmin_{a,b\in\mathbb{R}} \quad K\left(\sum_{(x,y)\in D} \frac{1}{1+e^{-(ax+b)}}\right)$

Wobei K die log-loss Funktion ist. Wir nutzen also die Aktivierungsfunktion, um den Unterschied zwischen 0 und 1 sichtbar zu machen, da eine lineare Funktion hierzu nicht in der Lage ist. Um 460 auszurechnen reichen keine elementaren Methoden, man muss also etwas komplexere numerische Optimierung anwenden (der Computer kann das). R macht das mit dem Kommando:

> glm(x ~ y, family = binomial())

Nun definieren wir:

> g <- function(x){1/(1+exp(-(1.251*x-8.113)))}

Das sollte nun stimmen; wir bekommen dementsprechend:

Stunden	4	5	6	7	8	9	10
P(bestehen)	0.0427	0.135	0.353	0.656	0.869	0.959	0.988

Tadaa! Wir bekommen also eine Wahrscheinlichkeit.

Logistische Regression und statistische Abhängigkeit Was aber interessanter ist, ist folgende Frage: gegeben diese Ergebnisse, wie ist die Wahrscheinlichkeit, dass die Abhängigkeit des Bestehens von der Lernzeit reiner Zufall ist? Das ist eine klassische statistische Frage, und im Zusammenhang mit logistischer Regression gibt es den sog. **Wald-test**:

$$\frac{(\theta_{ML} - \theta)^2}{var(\theta_{ML})} \tag{461}$$

Er gibt die Wahrscheinlichkeit, dass die gegebene Verteilung zufällig ist, also dass es keine Abhängigkeit der beiden Variablen (Lernzeit / Bestehen) gibt. Das ist verwandt mit dem Test des Likelihood-Verhältnisses, was ein wenig einfacher zu verstehen ist. Seien

- ω unsere Beobachtungen, also die Daten in D1;
- H_0 die Nullhypothese, also Unabhängigkeit der Variablen: Lernzeit hat keinen Einfluss auf das Bestehen, Wahrscheinlichkeit von Bestehen oder Nicht-bestehen wird durch Maximum Likelihood geschätzt.

- H_1 die Alternativhypothese, also unsere durch logistische Regression berechnete bedingte Verteilung.

Dann haben wir den Likelihood-Quotienten:

$$R(\omega) := \frac{P(\omega|H_0)}{P(\omega|H_1)} \qquad \text{(Sonderfall für } P(\omega|H_1) = 0\text{)} \tag{462}$$

Hierauf kann man nun den Schwellentest S_t anwenden, $t > 0$, und üblicherweise $t = 0.05$. Zur Erinnerung: das ist der Test, der sich für H_0 entscheidet falls $R(\omega) > t$, und für H_1 andernfalls, also:

$$S_t(\omega) = \begin{cases} H_0, \text{ falls } R(\omega) > t \\ H_1 \text{ andernfalls.} \end{cases} \tag{463}$$

Das können wir/Sie nun ausrechnen:

Übung

1. Berechnen Sie für das Beispiel in diesem Abschnitt (die Daten $D1$, H_0 und H_1 wie oben angegeben) den Quotienten $R(\omega)$.

2. Ist das Ergebnis signifikant, das heißt, würde der Schwellentest H_0 zurückweisen (wir setzen $t = 0.05$)?

33 Schema F der Regression/Klassifikation

Folgendes gilt für Regression im Allgemeinen, sowohl lineare als logistische. Es gilt aber auch darüber hinaus für die meisten Fälle von machine learning, deswegen nenne ich es Schema F – es taucht eigentlich in jedem Modell auf. Folgende Zutaten:

1. Ein Modell M. M nimmt zwei Argumente, nämlich $\vec{\theta}$, die Parameter, die es erst spezifizieren, und $\vec{x}$, die Eingabe. Es liefert eine Ausgabe y. Wir schreiben also z.B. $M(\vec{\theta})(\vec{x}) = y$.
2. Eine Kostenfunktion $K : M \times M \to \mathbb{R}_0^+$.
3. Einen Datensatz D

Wir suchen die optimalen Parameter θ gegeben D. Das macht man, indem man die Regressionsfunktion zu minimieren versucht. Erstellen wir also zunächst die **Regressionsfunktion**. Das geht wie folgt:

$$K(M(\theta)(\vec{x}), y_x) : \ (\vec{x}, y_x) \in D \tag{464}$$

sind sie Kosten auf einem Datenpunkt $(\vec{x}, y_x) \in D$. Wir setzen also die Modellfunktion auf der einen und den richtigen Wert auf der anderen Seite in die Kostenfunktion ein. Was wir nun machen ist:

Wir summieren die Werte für alle $(\vec{y}, x_y) \in D$. Wir bekommen also:

$$\sum_{(\vec{x}, y_x) \in D} K(M(\theta)(\vec{x}), y_x) \tag{465}$$

Das ist bereits die Regressionsfunktion. Wichtig ist: ihr Argument ist θ; wir suchen

$$\underset{\theta}{argmin} \sum_{(\vec{x}, y_x) \in D} K(M(\theta)(\vec{x}), y_x) \tag{466}$$

Das ist das Problem, ein einfaches Optimierungsproblem. Dieses Schema funktioniert für alle Arten von Regression und z.B. auch für neuronale Netze. Für SVM ist das Problem aber etwas anders gelagert.

Das eigentliche Problem ist dann übrigens, die Lösung von 466 tatsächlich zu finden; hier gibt es viele Lösungen, einige davon haben wir bereits besprochen.

Hausaufgabe zum 25.11.2019

Wir haben den Unterschied zwischen einem Modell und seiner Regressionsfunktion besprochen. Z.B. für die lineare Regression lautet das Modell

$$f(x) = ax + b \tag{467}$$

im einfachsten Fall; im komplexeren Fall (Vektor als Eingabe)

$$f(\vec{x}) = \vec{a}^\top \vec{x} + b \tag{468}$$

Das ist die Art von Funktion, die wir am Ende haben wollen als Modell. Um aber die passenden Parameter $\vec{a}, b$ zu finden, müssen wir eine andere Funktion optimieren, nämlich die Regressionsfunktion. Diese ergibt sich aus der Modellfunktion und der Kostenfunktion. Die Kostenfunktion ist in unserem Fall

$$K(x, y) = (x - y)^2 \tag{469}$$

Da wir die Kosten für jeweils die Ausgabe des Modells $f(\vec{x})$ und die "korrekte" Ausgabe in den Daten messen wollen, interessiert uns

$$\sum_{(\vec{x},y)\in D} K(f(\vec{x}), y) \tag{470}$$

Wenn wir nun für $f(\vec{x})$ unser Modell einsetzen, für K unsere Kostenfunktion, dann bekommen wir

$$\sum_{(\vec{x},y)\in D} K(f(\vec{x}), y) = \sum_{\vec{x},y\in D} (\vec{a}^\top \vec{x} + b - y)^2 \tag{471}$$

Das ist nun die Regressionsfunktion; wir suchen als optimales Modell

$$\underset{\vec{a},b}{argmin} \sum_{\vec{x},y\in D} (\vec{a}^\top \vec{x} + b - y)^2 \tag{472}$$

Soweit der Schritt vom Modell, das optimiert werden soll, zur Regressionsfunktion, deren Minimalstelle wir suchen.

Wir haben nun genau dieselbe Situation für die logistische Regression, nur mit etwas anderer Modell- und Kostenfunktion. Das **Modell** sieht wie folgt aus:

$$g(x) = S(f(\vec{x})) = S(\vec{a}^\top \vec{x} + b) \tag{473}$$

wobei S die logistische Sigmoid-Funktion ist. Wir suchen die optimalen Parameter $\vec{a}, b$ zur Klassifizierung nach 0,1. Die Kostenfunktion ist folgende (für $y \in \{0, 1\}$ das korrekte label):

$$(474)\quad K(P_f, y) = -\frac{1}{m} \sum_{(\vec{x},y)\in D} [y \cdot log P_f(Y = y|\vec{x}) + (1-y) \cdot log(1 - P_f(Y = y|\vec{x}))]$$

Hier ist

$$(475)\quad P_f(Y = y|\vec{x}) = S(f(\vec{x}))$$

Wir haben also das Modell und die Kosten. Die Regressionsfunktion für die logistische Regression ergibt sich, wenn wir das Modell in die Kostenfunktion einsetzen wie in (470). Das ist also diejenige Funktion, deren Minimalstelle die optimalen Parameter liefert.

1. Schreiben Sie diese Funktion aus, mit den Argumenten, für die es die Minimalstellen zu finden gilt.
2. Schreiben Sie dieselbe Funktion in Python-nahen Pseudocode, so dass die Implementierung geradeaus ist.

Lösung auskommentiert!

34 Nearest neighbour Regression

34.1 Lineare Algebra (reine Wiederholung)

Nearest neighbour regression ist eine denkbar simple Methode: wir haben eine Funktion $f : V \to C$, wobei V ein Vektorraum ist, und C eine (endliche) Menge von Klassen; wir haben eine Menge $D \subseteq V \times C$ von Datenpunkten, mittels derer wir die Funktion f "lernen" sollen. Was wir hierfür erstmal brauchen ist der Begriff der **Norm**:

Eine Norm, definiert auf einem Vektorraum V ist das eine Funktion

$$\| - \| : V \to \mathbb{R}_0^+$$

es werden also beliebige Vektoren auf einen nicht-negativen Wert abgebildet. Zusätzlich muss $\| - \|$ noch folgende Bedingungen erfüllen f.a. $\vec{v} \in V$, $\lambda \in \mathbb{R}$.

1. $\|\vec{v}\| = 0 \Rightarrow \vec{v} = 0_V$ (die 0 des Vektorraums, neutral für Addition)
2. $\|\lambda \cdot \vec{v}\| = |\lambda| \cdot \|\vec{v}\|$, wobei $|\lambda|$ der Betrag ist (respektiert Skalarmultipikation)
3. $\|\vec{v} + \vec{w}\| \leq \|\vec{v}\| + \|\vec{w}\|$ (allgemeine Dreiecksungleichung)

Die intuitivste Norm ist die *euklidische*, die jedem Vektor seine **Länge** zuweist (wenn wir einen Vektor als eine Linie vom Ursprung auf seine Koordinaten (im n-dimensionalen Raum) auffassen. Diese Norm basiert auf einer Verallgemeinerung des Satz des Pythagoras:

$$\|(v_1, ..., v_n)\| = \sqrt{v_1^2 + ... + v_n^2} \tag{476}$$

In dieser geometrischen Interpretation wird Bedingung 3 zur **Dreiecksungleichung**: in jedem rechwinkligen Dreieck ist die Länge der Hypothenuse geringer als die Summe der Länge der Katheten. Es gibt aber noch viele weitere Normen, z.B. die sog. p-Norm, wobei $p \geq 1$ eine reelle Zahl ist:

$$\|(v_1, ..., v_n)\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{\frac{1}{p}} \tag{477}$$

Für $p = 1$ vereinfacht sich das zu

$$\|(v_1, ..., v_n)\|_1 = \sum_{i=1}^{n} |v_i| \tag{478}$$

Das ist die sog. Manhattan-Norm, weil man immer rechtwinklig um die Blocks fahren muss – das gibt im 2-dimensionalen Fall also die kürzeste Strecke in Manhattan an.

Darauf basiert der Begriff der Metrik; jede Norm induziert eine Metrik d mittels

(479) $d(\vec{v}, \vec{w}) = \|\vec{v} - \vec{w}\|$

wobei natürlich

(480) $\vec{v} - \vec{w} = (v_1 - w_1, ..., v_i - w_i)$

Es ist nicht schwer zu sehen dass gilt:

- $d(x, y) \geq 0$ (positiv)
- $d(x, y) = d(y, x)$ (symmetrisch)
- $d(x, y) \leq d(x, z) + d(z, y)$ (Dreiecksungleichung)

Die **euklidische Distanz** ist definiert durch die euklidische Norm, mit

(481) $d_2(\vec{v}, \vec{w}) = \|\vec{v} - \vec{w}\|_2$

34.2 Die eigentliche Klassifikation (einfach)

Nun machen wir einfach folgendes: gegeben einen beliebigen Vektor $\vec{v}$, eine Norm $\| - \|$ mit Metrik d, definieren wir

(482) $nn_D(\vec{v}) = argmin_{(\vec{w},c)\in D} d(\vec{v}, \vec{w})$

Wir suchen uns also den **nächsten Nachbarn** nach unserer Metrik. Als nächstes machen wir das denkbar naheliegendste: wir machen genau das, was der nächste Nachbar macht (Einfachheit halber fassen wir unseren Datensatz D als partielle Funktion $D : V \to C$ auf)

(483) $NNR_D(\vec{v}) = D(nn_D(\vec{v}))$

In Wort: $NNR_D : V \to C$ ist die nearest neighbour regression über D, und diese (vollständige) Funktion funktioniert, indem sie für jeden Eingabevektor $\vec{w}$ zunächst den (nach Metrik d nächstliegenden Vektor $\vec{v}$ im Datensatz findet (ein endliches Suchproblem), um ihm dann dieselbe Klasse zuzuordnen, die auch $\vec{v}$ zugeordnet wird.

Ein einfaches Beispiel wäre z.B. eine Sprach- oder Bilderkennung: bei Bildern wäre das Pixelbild ein Vektor, in dem jeder Pixel eine Komponente ist, und der Wert ist die Farbe der Pixel. Bei Spracherkennung kann man die verschiedenen Frequenzbereiche F1-F3 auf einen Vektor verteilen, mit dem Wert als Frequenz; die Klassifikation wäre dann die Frage: welcher Gegenstand ist auf dem Bild abgebildet (aus einer endlichen Menge), bzw. welcher laut wird geformt? NNR ist einfach folgende Methode: finde den ähnlichsten Punkt in unserem Datensatz und klassifiziere entsprechend.

34.3 Generalisierung: k nearest neighbour

Es gibt eine einfache Generalisierung von nearest neighbour, die nicht die Klasse des nächsten Nachbarn betrachtet, sondern die Klassen der k nächsten Nachbarn, wobei k ein Hyperparameter ist. Die Klassifikation erfolgt dann soz. per **Mehrheitsbeschluss**. Etwas formaler: sei

$$nn_D^k(\vec{v}) \text{ die Menge der } k \text{ nächsten Nachbarn von } \vec{v} \text{ in } D$$

Die Mehrheitsklasse einer Menge $X \subseteq D_{data}$ wäre definiert als

(484) $maj(X) := argmax_c\{\vec{x} : \vec{x} \in X, D(\vec{x}) = c\}$

Also wird die Klassifikation ganz einfach zu:

(485) $NNR_D(\vec{v}) := maj(nn_D^k(\vec{v}))$

Also ganz einfach! Wichtig für die Auswahl von k ist:

- ein zu kleines k (insbesondere $k = 1$) kann zur Folge haben, dass einzelne Aussreisser die Klassifikation bestimmen
- ein zu großes k kann zur Folge haben, dass kleine Klassen/kleine Cluster einfach überstimmt werden.

Wie immer hängt alles von der Datenlage und Datenmenge ab: gibt es kleine Klassen (also solche mit wenig Datenpunkten)? Dann sollte man k klein wählen. Sind die Klassen klar in Cluster aufgeteilt, oder verstreut und nicht gut separierbar? Im letzteren Fall sollte man k nicht zu groß wählen. Letzten Endes ist es aber schwierig allgemeine Regeln aufzustellen. Hier hilft nur: **Hyperparameter optimieren**, mit den gebotenen Vorsichtsmassnahmen.

34.4 Nearest Neighbour Regression und Logistische Regression

Was ganz interessant ist: NNR kann nicht durch eine logistische Regression beliebiger Ordnung modelliert werden. Das sieht man sehr leicht an einem Beispiel: man nehme einen Datensatz

$$D = \{((0,0),a),((0,1),b),((0,2),a),((0,3),b),...\}$$

NNR wird hier beliebig oft wechseln können zwischen a und b; jedes lineare Modell, wie z.B. bei logistischer Regression, wird nur eine konstante Anzahl von wechseln ermöglichen können. Allerdings gilt das nur *apriori* und ohne Trainingsdaten: denn wenn wir eine NNR haben zusammen mit einem Datensatz D, dann wird es auch nur eine konstante Zahl an wechseln geben, einfach weil die Datenmenge endlich ist!

Wir sehen hier aber, was ein Modell des maschinellen Lernens ausmacht: es ist eine Funktion von (Trainings)Daten zu einer (Klassifikations- oder Regressions-)Funktion.

$$\text{(Abstraktes) ML-Modell} \underset{Daten}{\Longrightarrow} \text{(Konkrete) Funktion} \underset{Eingabe}{\Longrightarrow} \text{Ausgabe}$$

Übrigens gilt auch bei dieser Methode: man sollte die Daten aufspalten in Trainings- und Testdaten, um festzustellen ob die Methodik angemessen ist. Das kann in manchen Fällen der Fall sein, in manchen nicht.

Stunden gelernt	4	5	6	7	8	9	10
Bestanden	0	0	1	0	1	1	1

Wenn wir wiederum diese Tabelle betrachten, dann sehen wir wir $NNR(5.6) = 1$ haben, $NNR(7.4) = 0$. Hier sehen wir das zentrale Merkmal: *NNR generalisiert sehr wenig.* Der Vorteil, dass wir keine weiteren Annahmen in die Daten stecken, ist also auch zugleich ein Nachteil, denn das Modell ist sehr anfällig für Unregelmäßigkeiten in den Daten, da es sie 1 zu 1 übernimmt. Das kann in manchen Fällen gut sein, insbesondere wenn es viel Varianz in den Daten gibt; das lässt sich aber schwer apriori sagen.

34.5 Probleme und Perspektiven

Skalierung NNR ist natürlich sehr **sensibel für Skalierung**: falls ein Parameter sehr weit gestreut ist, wird er die Klassifikation dominieren. Man sollte NNR in jedem Fall die Daten standardisieren.

Rauschen NNR ist dann aber ebenfalls sehr sensibel für **Rauschen** in den Daten: wenn wir ein Merkmal haben, dass eigentlich keine Bedeutung hat, kann es hier dennoch die gesamte Klassifikation verzerren (zur Erinnerung: lineare/logistische Regression filtern so ein Merkmal einfach automatisch raus – der entsprechende Parameter wird gering gewichtet).

Eine Lösung: Gewichtete Merkmale Was man dagegen zun kann ist: wir gewichten die Merkmale. Hierfür muss man natürlich wieder trainieren: man kann das machen, indem man die Daten in Trainings- und Testdaten teilt. Man klassifiziert die Testdaten auf Basis der gewichteten Testdaten, und schaut welche Gewichte das am besten machen. Allerdings werden die (realwertigen) Gewichte hierdurch nicht ausreichend spezifizert; man braucht weitere Beschränkungen.

Hausaufgabe

Skizzieren Sie einen NNR-Klassifizierer mit Pseudocode!

Skizzieren Sie einen Algorithmus, der für einen beliebigen (passenden) Datensatz einen NNR-Klassifizieren erstellt.

35 Kurze Zusammenfassung für Regression und ML

- Warum ist es gut einfache Methoden zu nutzen, wenn es komplexe gibt?
- Kann man sagen: ein ML-Algorithmus ist besser als ein anderer?
- Was könnte PAC-Lernbarkeit in Bezug auf Regression bedeuten? (und allgemein auf unsere konkreten Probleme bezogen?)
- Wann braucht man Test- und Trainingsdaten, wann nicht?
- Wann braucht man zusätzlich ein Development-set?
- Was bedeutet Regularisierung im engen und weiten Sinn?
- Was ist eine Kostenfunktion im ML, warum ist sie so wichtig?
- Warum benutzt man zur Klassifikation eine *probabilistische* Kostenfunktion, nicht einfach richtig/falsch Kosten?

36 Support Vector Machines für Klassifikation

36.1 Hyperebenen und lineare Separierung

Hier ist der Begriff der **Hyperebene** zentral. Eine Ebene ist ein $(n-1)$-dimensionaler Teilraum eines n-dimensionalen Raums. Eine Hyperebene in der Ebene (2D) ist eine Gerade, eine Hyperebene im 3D-Raum ist eine Ebene. Hyperebenen stellen also eine Generalisierung von Ebenen (im 3-dimensionalen Raum) auf beliebige Dimensionen dar. Eine normale Ebene ist spezifiert durch

- einen **Stützvektor** $\vec{s}$
- und zwei linear unabhängige **Richtungsvektoren** $\vec{r}_1, \vec{r}_2$.

Ein Punkt p liegt auf der Ebene, falls er sich darstellen lässt als

$$p = \lambda_1 \vec{r}_1 + \lambda_2 \vec{r}_2 + \vec{s}.$$

Diese Definition lässt sich leicht verallgemeinern: man nimmt, für einen Raum $\mathbb{R}^n$, einfach den Stützvektor

$$\vec{s} \in \mathbb{R}^n$$

und ebenso $n-1$ linear unabhängige Richtungsvektoren

$$\vec{r}_1, ..., \vec{r}_{n-1}$$

Technisch gesehen ist die Hyperebene eine Menge von Punkten:

$$H = \{\vec{s} + \lambda_1 \vec{r}_1 + ... + \lambda_{n-1} \vec{r}_{n-1} : \lambda_1, ..., \lambda_{n-1} \in \mathbb{R}\} \tag{486}$$

Das ist wie folgt zu verstehen: $\vec{s}$ ist der Stützvektor, $\vec{r}_1, ..., \vec{r}_{i-1}$ sind die Richtungsvektoren. $\vec{s}$ bezeichnet einen beliebigen Punkt auf der Ebene; die Richtungsvektoren geben die Richtungen an, in die wir von diesem Punkt gehen können. Wichtig ist dass alle diese Vektoren voneinander unabhängig sind.

Linear unabhängig bedeutet: $\vec{x}_1, ..., \vec{x}_i$ sind linear unabhängig, falls gilt: wenn

$$\lambda_1 \vec{x}_1 + ... + \lambda_i \vec{x}_i = 0,$$

dann $\lambda_1 = ... = \lambda_i = 0$.

Hyperebenen haben die Eigenschaft, dass Sie den n dimensionalen Raum ein zwei Teile teilen – je nachdem man sich in der n-ten Dimension in eine positive oder negative Richtung bewegt.

Koordinatenform Es gibt noch eine weitere Darstellung von Hyperebenen: wir können eine Hyperebene im n-dimensionalen Raum darstellen mit einer (linearen) Gleichung

(487) $x_1 w_1 + ... + x_n w_n + s = 0$

NB: hier sind alle $x_i, w_i \in \mathbb{R}$. Man kann die Gleichung auch als eine einfache Vektormultiplitation schreiben: für

$$\vec{w} = (w_1, ..., w_n),\ \vec{x} = (x_1, ..., x_n)$$

haben wir dann

(488) $\vec{w}^\top \vec{x} + s = 0$

Die Punkte der Hyperebene sind dann einfach die Menge der Lösungen:

(489) $H = \{(x_1, ..., x_n) : x_1 w_1 + ... + x_n w_n + s = 0\} = \{\vec{x} : \vec{w}^\top \vec{x} + s = 0\}$

Also die Menge der Lösungen der Gleichung bildet einen Teilraum!

Kleiner Einschub Im n-dimensionalen Raum reduziert jede Gleichung die Dimension um 1. Sprich: die Menge der Lösungen zweier Gleichungen liefern einen $(n-2)$-dimensionalen Teilraum etc. Wenn wir n (linear unabhängige) Gleichungen haben, ist die Lösung eindeutig, also ein Punkt - ein 0-dimensionaler Raum!

Und weiter! Diese Darstellung hat einen wichtigen Vorteil: sie erlaubt eine unmittelbare Definition der Punkte, die oberhalb bzw. unterhalb der Ebene liegen:

$$\uparrow H = \{\vec{x} : \vec{w}^\top \vec{x} + s \geq 0\} \tag{490}$$

$$\downarrow H = \{\vec{x} : \vec{w}^\top \vec{x} + s < 0\} \tag{491}$$

SVM-Klassifizierung nach zwei Klassen y_1, y_2 sucht nun eine Hyperebene, die genau folgendes leistet:

$$y_1 \cong \uparrow H \tag{492}$$

$$y_2 \cong \downarrow H \tag{493}$$

Einfachheit halber nimmt man bei binärer Klassifizierung durch SVM an, dass unsere Zielklassen

$$y_i \in \{-1, 1\}$$

sind. Deswegen suchen wir also diejenige Hyperebene H, definiert durch $\vec{w}, s$ so dass für alle $(\vec{x}_i, y_i) \in D$.

$$y_i(\vec{w}^\top \vec{x}_i + s) \geq 0 \tag{494}$$

Wenn es eine solche Hyperebene gibt, dann sind die Klassen **linear separierbar**. Soweit der einfachste Fall: die Hyperebene separiert die Klassen. Wir haben allerdings hier vorausgesetzt, dass die Klassen linear separierbar sind; außerdem haben wir selbst in diesem Fall ein unterdeterminiertes Problem:

- **Jede** Hyperebene, welche die Klassen separiert, erfüllt die letzte Gleichung.

36.2 Normalitätserwägungen

Zunächst gilt es, Hyperebenen kanonisch zu repräsentieren: Es ist nicht schwer zu sehen, dass falls $\vec{w}, s$ eine Hyperebene darstellt, dann stellt $\lambda\vec{w}, \lambda s$ *dieselbe* Ebene dar.

$$x_1 w_1 + ... + x_n w_n + s = 0 \iff \lambda(x_1 w_1 + ... + x_n w_n + s) = 0 \tag{495}$$

Deswegen nimmt man normalerweise folgende **Normalform**: wähle $\vec{w}, s$ so dass

$$\vec{w} = \frac{\vec{w}}{\|\vec{w}\|} \tag{496}$$

$$s = \frac{|s|}{\|\vec{w}\|} \tag{497}$$

Dadurch haben wir sichergestellt:

- $\vec{w}$ ist ein Vektor der Länge 1 ($\|\vec{w}\| = 1$), und
- s bezeichnet die perpendikulare Distanz der Hyperebene vom Ursprung.

Wir brauchen also einfach eine (euklidische) Norm von 1 für alle Vektoren.

▶ Durch skalare Multiplikation kann man jeden Vektor dergestalt skalieren (multipliziere mit $1/\|\vec{w}\|$).

Es gibt nur eine Repräsentation der Hyperebene, die derart aussieht; wenn wir nun verschiedene Lösungen haben, dann liegt das daran, dass es viele Hyperebenen gibt, welche die Daten separieren! Wir suchen nun "die Beste".

36.3 Das Problem der optimalen Separierung

SVM versuchen also ein klassisches Optimierungsproblem zu lösen: wir versuchen die Abstände der Punkte zu einer Hyperebene zu **maximieren**.

Wir formulieren das mathematische Problem, unter der Annahme dass die Daten linear separierbar sind. Nehmen wir weiterhin an, unsere Daten haben die Form

$$D \subseteq \mathbb{R}^2 \times \{-1, 1\}.$$

Die Distanz eines Punkte $\vec{x}$ zur Hyperebene $H(\vec{w}, s)$ ist, wie oben beschrieben, definiert durch

(498) $d(\vec{x}, H(\vec{w}, s)) = |\vec{w}^\top \vec{x} + s|$

Also der Betrag der Lösung, wenn wir $\vec{x}$ in die Gleichung einsetzen.

Das Optimierungsproblem von SVM ist nun folgendes:

- Wir suchen *nicht* die maximale Distanz über alle Punkte summiert; sondern wir suchen die *maximale minimale* Distanz für jeweils beide Klassen. Also keine Summe über den Datensatz!
- Wir möchten, dass die minimale Distanz für $\{(\vec{x}, 1) \in D\}$ gleich ist der minimalen Distanz für $\{(\vec{x}', -1) \in D\}$,
- und beide maximal sind! (also maximale Minimaldistanzen)

Das resultiert in einem sog. **Optimierungsproblem mit Seitenbedingungen**, wie sie sehr häufig auftreten. Wir suchen eine optimale Lösung (Minimalstelle einer Funktion); aber diese Lösung muss gleichzeitig weitere, unabhängige Bedingungen erfüllen. Das Problem für SVM sieht wie folgt aus.

Optimierungsproblem:

(499) $argmin_{\vec{w}} \frac{1}{2} \vec{w}^\top \vec{w}$

Seitenbedingung:

(500) $y_i(\vec{w}^\top \vec{x}_i + s) \geq 1,$

für alle $(\vec{x}_i, y_i) \in D$.

Intuitiv soll also

1. alle Objekte der Klasse 1 mit $x \geq 1$,
2. alle Objekte der Klasse -1 mit $x \leq -1$ klassifiziert werden,
3. die erste Gleichung stellt sicher, dass die Parameter minimal sind (erstaunlicherweise, aber nur zusammen mit der Normalitätsbedingung!).

Die Symmetrie 1/-1 ist hier nötig, da wir die Linie genau in der Mitte der Klasse ziehen möchten. Das garantiert, zusammen mit Bedingung 2, erstaunlicherweise die maximalen Minimalabstände!

Man beachte den **freien Stützvektor** s: der ist *nicht* Teil der Optimierung, aber entscheidend für die korrekte Klassifikation.

1. Die erste Bedingungen ist ein **konvexes Optimierungsproblem** und daher effizient lösbar mit Methoden wie gradient descent (es handelt sich ja um einen quadratischen Term).
2. Die Seitenbedingungen lassen sich in dieses Problem inkludieren mittels sog. **Lagrange-Multiplikatoren** – das sind Faktoren, die dem Term zugefügt werden und sicherstellen, dass die Seitenbedingungen eingehalten werden durch jede Lösung.

Es gibt noch eine wichtige Beobachtung: alle Vektoren, sowohl Parameter als auch Datenpunkte, kommen nur "gebunden" vor in Produkten der Form

(501) $\vec{x}^\top \vec{y}$

Das ist von entscheidender Bedeutung für den Kernel-Trick! (s.u.)

Vorteile von SVM

1. Was wir hier besprochen haben ist das einfache SVM-Modell; also ein linearer bzw. gar kein Kernel. Der Hauptvorteil von einfachen SVM, bei linear separierbaren Daten, ist, dass ein overfitting praktisch ausgeschlossen ist. Wir bekommen einen einfachen Klassifikator, der sehr gut auf neue Daten generalisiert.
2. SVM interessieren sich nur für Grenzfälle. Ausreisser "in die richtige Richtung" (wenn man so sagen kann) haben keine Einfluss auf die Klassifikation. Das "Hinterland" spielt also keine Rolle. Das ist sehr wichtig und anders als z.B. bei logstischer Regression.

3. Bei logistsicher Regression spielt *jeder* Datenpunkt eine Rolle, da jeder Kosten verursacht. Falls nun z.B. die Daten unasgewogen sind (Klasse 1 viel häufiger als Klasse 0, dann rückt die Entscheidungsgrenze automatisch an Klasse 1 heran - damit lassen sich Kosten reduzieren!

Problem von SVM Es gibt einige Probleme von SVM:

1. SVM sind sensitiv für Skalierung. Ein SVM-basierter Klassifikator, trainiert auf $stand(D)$, wobei D ein Datensatz ist, $stand$ die Standardisierungsfunktion, wird (möglicherweise) einen neuen Datenpunkt $stand(x)$ anders klassifizieren, als derselbe Klassifikator trainiert auf D den Datenpunkt x. Deswegen sollte man im Zweifelsfall standardisieren.

2. SVM sind äußerst sensibel für Ausreißer, also Datenpunkte, die evtl. auf noise bzw. falscher Klassifizierung basieren. Ein einziger solcher Datenpunkt kann die gesamte Klassifizerung unsinnig machen, da die Bedingung in 513 **hart** ist; wenn ein Datenpunkt sie nicht erfüllt, ist der Klassifikator falsch.

3. Ein einzelner Ausreißer (in die falsche Richtung) kann sogar dafür sorgen, dass ein ganzer Datensatz nicht mehr linear separierbar ist;

Insbesondere Punkt 2 und 3 sind ein ernstes Problem, da in großen Datensätzen praktisch immer Ausreißer vorliegen. Aus diesem Grund definiert man eine neue Klassen von SVM-Klassifikatoren, nämlich solche mit **weichen Rändern**.

36.4 Zwischenstück: Optimierung mit Lagrange Multiplikatoren

Lagrange Multiplikatoren werden genutzt, um Optimierungsprobleme mit Nebenbedingungen zu lösen. Also wie bei uns hier. Der Einfachheit halber nehmen wir aber ein etwas simpleres Problem. Nimm an, wir suchen

(502) $\underset{x,y}{argmax} f(x,y)$

unter der Bedingung das

(503) $g(x,y) = c$

wobei c eine Konstante ist; letzteres nennen wir die Nebenbedingung. Nimm nun an, g ist eine lineare Funktion, sprich:

(504) $\{x,y,z : g(x,y) = z\}$

ist eine (schiefe) Ebene im Raum, und die Menge der Punkte

(505) $G := \{x,y,c : g(x,y) = c\}$

(wobei c die Konstante der Nebenbedingung!) ist eine *Gerade*. Die 2D-Gerade:

(506) $\{(a,b) : g(a,b) = c\}$

kann am nun durch den 3D-Funktionsgraphen von f ziehen: das Ergebnis wird natürlich ein 2D-Ausschnitt aus der "Landschaft". Was wir suchen das *Maximum* dieses 2D Funktionsgraphen, also die Koordinaten seiner Maximalstelle.

Nun hilft uns folgende Visualierung: wo immer wir eine Maximalstelle haben, wird die Gerade G *parallel* zu den Höhenlinien (wie auf der Landkarte) von f verlaufen (sonst würden wir ja in einer Richtung an Höhe gewinnen. Das bedeutet wiederum: der Gradient

(507) $\nabla_{x,y} f(x,y)$

steht an diesem Punkt a^*, b^* orthogonal zu G (Gradient ist immer orthogonal zur Tangente der Höhenlinie). Aber Moment: G selber ist ja auch nur eine Höhenlinie von g, nämlich die Höhe c. Das bedeutet: der Gradient

(508) $\nabla_{x,y} g(x,y)$

steht ebenfalls orthogonal auf G, also, also sind die beiden Gradienten (als Vektoren) an diesem Punkt **kollinear**, d.h.

(509) $\nabla_{x,y} f(a^*, b^*) = -\lambda \nabla_{x,y} g(x, y)$

für $\lambda \in \mathbb{R}$ ein Skalar – unter Bedingung dass $g(x, y) = c$. Das $-\lambda$ ist natürlich überflüssig, aber im folgenden nützlich.

Wir versuchen, diese Bedingungen in eine einzige Funktion unterzubringen. Das macht man mit der **Lagrange-Funktion**:

(510) $\Lambda(x, y, \lambda) := f(x, y) + \lambda(g(x, y) - c)$

Die Lösung des oben beschriebenen Optimierungsproblems mit Seitenbedingung ist jetzt äquivalent zu einem *lokalen* Maximum von $\Lambda(x, y, \lambda)$ (mit Parameter λ!).

- Fall $g(x, y) = c$, dann ist $\lambda(g(x, y) - c) = 0$ f.a. λ
- Wenn $f(x, y)$ hier maximal ist, führt jede Änderung zu einem niedrigeren Wert
- Gleichzeitig gibt es ein λ (entweder positiv oder negativ), so dass mit wachsendem/sinkenden x, y der Gesamtwert der Funktion Λ erstmal sinkt!
- λ selbst interessiert uns aber nicht, wir brauchen am Ende nur x, y

Das lokale Minimum finden wir mittels Lösung der Gleichung

(511) $\nabla_{x,y,\lambda} \Lambda(x, y, \lambda) = 0$

Das ist aber nur ein Kandidat, also eine notwendige Bedingung, keine hinreichende!

Man beachte nun: die Bedingung, dass g linear ist, ist keinesfalls notwendig – die einzige tatsächliche Beschränkung ist eben dass wir die Nullstellen von $\nabla_{x,y,\lambda} \Lambda(x, y, \lambda)$ finden müssen!

36.5 Regularisierung von SVM – Weiche Ränder

Weiche Ränder von SVM bedeuten so viel wie: wir erlauben, dass unser Klassifikator auch mal falsch klassifiziert. Der Fehler ist mit Kosten verbunden, aber wenn wir dadurch Vorteile erhalten, die diese Kosten übertreffen, dann klassifizieren wir lieber einmal falsch, als dass wir sehr enge Ränder haben für den ganzen Datensatz. Wir ändern also den Ansatz wie folgt: wir führen eine

$$\text{Famile von } \textit{slack}\text{-Variablen } (\xi^i) : i \leq |D|$$

ein, wobei es für jeden Datenpunkt eine Instanz gibt, und allgemein gilt dass $\xi^i \geq 0$. Unser Klassifikator wird nun wie folgt definiert:

Optimierungsproblem

$$\underset{\vec{w},\xi^i}{argmin}\ \frac{1}{2}\vec{w}^\top\vec{w} + C\sum_{i=1}^{|D|}\xi^i \tag{512}$$

unter

Seitenbedingung

$$y^i(\vec{w}^\top\vec{x}^i + b) \geq 1 - \xi^i, \tag{513}$$

für alle $i \in \{1, ..., |D|\}$.

Hier haben wir also $|D|$ zusätzliche Parameter, die wir allesamt optimieren müssen. Die slack-Variablen ξ^i haben also folgende Bedeutung:

- Falls ξ^i groß gewählt wird, dann wird der entsprechende Datenpunkt soz. "verschoben", der Ausreißer kommt also auf die "richtige Seite" zurück.
- Allerdings verursacht *jedes einzelne* $\xi^i > 0$ Kosten. Es lohnt sich also nur, wenn wir dadurch die Parameter entsprechend verbessern können!
- C ist ein Hyperparamter der weichen SVM: er besagt intuitiv, wie weich die Ränder sind. Je größer C, desto teurer wird eine Verschiebung eines evtl. Ausreißers, also desto härter werden die Ränder. Die "einfache" SVM hat also $C = \infty$.

Wichtig ist hier: ein kleines C (also "weiche Ränder") sind dann nützlich, wenn unsere Daten viel *noise* enthalten, und wenn wir verhindern, dass einzelne Datenpunkte unsere gesamte Klassifikation verschieben.

SVM mit weichen Rändern sind übrigens, wie man leicht sehen kann, immer noch lineare Modelle – nur dass sie *viel* mehr Parameter haben, nämlich einen für jeden Datenpunkt. Dementsprechend kann, bei großen Datensätzen, das Training durchaus aufwändig sein.

37 SVM: Merkmale und Kernel

Oftmals lassen sich Daten nicht sinnvoll linear separieren, auch nicht mit weichen Rändern. Das ist z.B. der Fall wenn die Datenpunkte der Kasse 1 um die Klasse 0 herum verteilt liegen, in einem Kreis. Was man hier machen kann ist folgendes: wir gehen ähnlich vor wie in der linearen Regression wenn diese unzulänglich ist: wir fügen ein neues Merkmal hinzu, z.B. quadratisch, kubisch etc., und führen die (immer noch lineare) Regression auf den neuen Daten aus – mit dem Ergebnis einer polynomialen Regression. Dasselbe kann man nun auch mit den Daten machen, auf denen man eine SVM appliziert, mit einem kleinen Unterschied:

- Wir erweitern nicht nur die bisherigen Komponenten; oftmals werden sie auch einfach ersetzt.

Man nennt diese Funktion, die die Daten transformiert, die

Merkmalsfunktion ϕ

Wir trainieren also den SVM-Klassifikator nicht auf den Daten D, sondern auf deren Transformation $\phi[D]$. Wenn man eine gute Merkmalsfunktion gefunden hat, den Klassifikator C auf $\phi[D]$ trainiert, dann bekommt man einen Klassifikator

$$C \circ \phi : \mathbb{R}^n \to \{-1, 1\}$$

Das bedeutet: um einen neuen Datenpunkt zu klassifizieren, muss man zunächst die Merkmalsfunktion ϕ nutzen, um dann das Ergebnis $\phi(\vec{x})$ zu klassifizieren. Hier kommt das Aber:

- SVM ist vollautomatisch, wie das meiste
- Eine gute Merkmalsfunktion ϕ zu finden ist Handarbeit, das sog. *feature engineering*.

37.1 Matrizen und lineare Abbildungen – eine Randbemerkung

Ein möglicher Ansatz ist folgender: wir nehmen als Merkmalsfunktion eine Matrix $\mathbf{A} : \mathbb{R}^n \to \mathbb{R}^m$, wobei $m > n$. Wir betten also ein einen **höherdimensionalen** Raum ein. Wir haben also

(514) $\phi(\vec{x}) = \mathbf{A}(\vec{x}) + \vec{b}$

Unser Klassifikator soll also folgendes tun:

(515) $y^i(\vec{w} \cdot (\mathbf{A} \cdot \vec{x}^i + \vec{b}) + b \geq 1$

für alle $i \in \{1, ..., |D|\}$. Nachdem aber die Multiplikation von Matrizen assoziativ ist, bedeutet das: das Ergebnis ist wiederum eine lineare Abbildung, also ist damit nichts gewonnen! Das wäre:

(516) $y^i((\vec{w} \cdot \mathbf{A}) \cdot \vec{x}^i + b \geq 1$

Der zugrundeliegender Punkt ist der: **lineare Modelle sind abgeschlossen unter Komposition**. Wenn wir also ein lineares Modell applizieren, und daraufhin noch eines (und noch eines etc.), dann ist es, als ob wir ein einziges appliziert hätten.

37.2 Feature-maps

Was man normalerweise zuerst macht, sind **polynomiale features**. D.h. man nimmt z.B. eine Abbildung

(517) $\phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$

Wir haben also hier quadratische Merkmale, und eine Einbettung $\phi : \mathbb{R}^2 \to \mathbb{R}^3$. Daraus folgt natürlich dass unser Optimierungsproblem etwas verlagert wird:

(518) $\vec{w}^T\phi(\vec{x}) = w_1x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2$

Wir suchen also drei Parameter statt zwei, und das Problem wird deutlich komplizierter (mit wachsender Komplexität des Polynoms). Es funktioniert dennoch Dank des Kernel Trick!

37.3 Der Kernel-Trick

Der Kernel-Trick besteht darin, dass wir hochdimensionale Merkmalsfunktionen nutzen können, ohne sie jemals zu berechnen bzw. zu "besuchen", wie man sagt. Es gibt nämlich eine Korrespondenz von **Merkmalsfunktionen** und **Kerneln**, einer gewissen Klasse von binären Funktionen. Nimm an,

- wir haben eine Merkmalsfunktion ϕ, die unsere Daten einbettet in einen hochdimensionalen Raum (möglicherweise sogar ∞-dimensional!).
- Wir ersetzen diese Merkmalsfunktionen durch eine **Kernelfunktion** K

wobei

$$\phi(\vec{x})^\top \phi(\vec{y}) \quad = \quad K(\vec{x}, \vec{y}) \tag{519}$$

K, der Kernel, ist dabei eine spezielle Funktion

$$K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$

die folgende zwei Bedingungen erfüllt:

1. es gibt einen Skalarproduktraum $(F, \top)$, wobei $F \subseteq \mathbb{R}^n$, $\top : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ und
2. eine Abbildung ϕ: $X \to F$ in diesen Raum gibt,

so dass

$$K(x, y) = \phi(x)^\top \phi(y) \tag{520}$$

für $x, y \in X$.

- F ist hier der erweiterte Merkmalsraum (höherdimensional),
- X der Ausgangsraum (niedrigdimensional)
- ϕ die Merkmalsabbildung.

Der Satz von Mercer garantiert:

Satz von Mercer (stark vereinfacht) Für jede Merkmalsfunktion ϕ gibt es eine Kernelfunktion. Qua Definition gilt natürlich auch das Gegenstück: für jede Kernelfunktion gibt es eine entsprechende Merkmalsfunktion.

Das erstaunliche ist: der Merkmalsraum, in welchen abgebildet wird, muss in der Praxis nicht bekannt sein, da jede Merkmalsfunktion durch den Satz von Mercer eine einfache Charakterisierung mittels Kernel besitzt.

Der Kernel-Trick ist nun folgender: die Merkmalsfunktion kann durchaus komplex sein – zur Optimierung (bzw. zum Training) brauchen wir nicht ϕ selbst, sondern nur den Term

$$\phi(\vec{x})^\top \phi(\vec{z}) = K(\vec{x}, \vec{z})$$

Soweit sogut –

- aber warum sollte uns $K(\vec{x}, \vec{z})$, für $\vec{x}, \vec{z} \in D$, interessieren?
- In der bisherigen Optimierung kommt das Skalarprodukt von Datenpunkten nicht vor!

Hier kommt soz. der eigentlich Trick: wir verändern unseren Datensatz mit der Kernel-Matrix. Um das Prinzip zu verstehen, macht es Sinn sich erstmal die (euklidische) Distanzmatrix zu betrachten.

Die Distanzmatrix, euklidisch Vorher aber müssen wir unsere Daten noch etwas modifizieren: nimm einen Datensatz

$$\mathbf{D} = \{d_1, ..., d_n\} \subseteq R^i, \text{ wobei } |\mathbf{D}| = n$$

Also n Datenpunkte mit je i Dimensionen. Diesen Datensatz können wir natürlich repräsentieren als eine Matrix

$$\mathbf{M} \in \mathbb{R}^{n \times i}$$

Diese Matrix kann man transformieren wir nun in eine Distanzmatrix:

$\mathbf{N} \subseteq \mathbb{R}^{n \times n}$, wobei $\mathbf{N}_{ij} = d_2(d_i, d_j)$

(also die euklidische Distanz zweier Datenpunkte)

Diese (euklidische) Distanzmatrix kodiert unseren ursprünglichen Datensatz eindeutig bis auf

Isometrie, d.i. Rotation, Spiegelung, Verschiebung

Diese Operationen präservieren Separierbarkeit und Hyperebenen, und erlauben also den Datensatz eindeutig zu rekonstruieren. Wir wollen aber etwas anderes:

Die Distanzmatrix, nicht euklidisch Wir bilden eine Distanzmatrix, aber nicht mit der euklidischen Distanz, sondern mit der Kernelfunktion:

$K\mathbf{N} \subseteq \mathbb{R}^{n \times n}$, wobei $\mathbf{N}_{ij} = K(d_i, d_j)$ (also die Distanz zweier Datenpunkte)

Diese Matrix $K\mathbf{N}$ kann man wiederum als unseren (transformierten) Datensatz auffassen (mit n Datenpunkten), die $j$te Zeile repräsentiert d_j.

Der Kernel-Trick, kompakt Eine SVM-Separierung auf $K\mathbf{M}$ entspricht einer SVM Separierung auf $\phi[\mathbf{D}] = \{\phi(d) : d \in \mathbf{D}\}$. Ohne dass man ϕ jemals berechnet!

Wir betrachten nun einige Merkmalsfunktionen und den azugehörigen Kernel.

Merkmal	Kernel	
$\phi(x_1, x_2) = (x_1, x_2, x_1 x_2)$	$K(\vec{x}, \vec{y}) = x_1 y_1 + x_2 y_2 + x_1 x_2 y_1 y_2$	
$\phi(x_1, x_2) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$	$K(\vec{x}, \vec{y}) = (\vec{x}^\top \vec{y})^2$	
...	$K(\vec{x}, \vec{y}) = (\vec{x}^\top \vec{y} + \theta)^k$	(Polynomialer Kernel)
...	$K_G(\vec{x}, \vec{y}) = exp(-\frac{1}{2\sigma^2} \|\vec{x} - \vec{y}\|_2)$	(Gaussischer Kernel)

Zur Erinnerung: die Gausssche Normalverteilunug sieht wie folgt aus:

$$f(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2\right) \tag{521}$$

Wir lassen also den (konstanten, normalisierenden) Term

$$\frac{1}{\sigma\sqrt{2\pi}} \tag{522}$$

weg, ansonsten ist der Kernel eine Normalverteilung über die **euklidische Distanz**: je größer die Distanz $\|\vec{x} - \vec{y}\|_2$, desto kleiner $K_G(\vec{x}, \vec{y})$ (die Normalverteilung hat ihr Maximum bei $x = \mu$!).

Polynomiale und Gaussische Kernel sind eigentlich die geläufigsten.

37.4 Hinge-loss

Hinge ist eine spezielle Kostenfunktion, die man zur Optimierung von SVM einsetzt. Es ist definiert als

(523) $hinge(y) = max(0, 1 - l \cdot y)$

wobei y die "**Rohausgabe**" des Klassifikators ist, also das Ergebnis des Applikation der linearen Funktion auf die Eingabe, nicht das "verarbeitete" Ergebnis -1,1, also

(524) $y = \vec{w}^\top \vec{x} + s$

wobei $\vec{w}, s$ die Hyperebene bestimmt.

$$l \in \{-1, 1\}$$

dagegen ist das **korrekte Ziellabel**. Die Funktion gibt also einen niedrigen Wert, falls l und y dasselbe Vorzeichen haben ($\cong$ richtige Klassifikation), und 0 falls $l \cdot y \geq 1$ (ausreichender Abstand zur Grenze).

- Die hinge-Funktion ist *konvex*, das heißt sie kann gut heuristisch optimiert werden; jedes lokale Minimum ist auch global.
- Wenn ich es richtig verstehen ist Hinge-loss alternative Art, SVM zu optimieren, die ähnlich, aber nicht zwangsläufig identische Ergebnisse garantiert.

38 Regression mit SVM

Man kann SVM auch zur Regression nutzen. Der Trick dabei ist: anstatt eine Straße *zwischen* Datenpunkten zu ziehen (mit möglichst breiten Rändern), versuchen wir eine Straße *durch* die Daten zu ziegen die

1. möglichst *dünn* ist, aber gleichzeitig
2. auf der möglichst *viele* Datenpunkte liegen.

Wir haben also das umgekehrte Problem: wir suchen eine Linie mit einem gewissen Rand; alles was innerhalb dieser Ränder liegt, verursacht keine Kosten, und nur außerhalb liegende Punkte verursachen Kosten, die wir minimieren.

Wir haben also wieder das **Grundprinzip von SVM**:

> Wir sind gegenüber vielen Änderungen im Datensatz invariant (im Gegensatz zur linearen Regression, die jeden Datenpunkt berücksichtigt).

Die SVR hat zwei wichtige Parameter:

1. ϵ; damit legen wir die breite der Straße fest;
2. C; damit legen wir fest, wie teuer es wird wenn ein Punkt außerhalb der Straße liegt.

Diese Regression basiert auf dem Prinzip der Stützvektoren: das sind in diesem Fall alle Punkt die außerhalb der Straße liegen. Allerdings werden hier ebenfalls – wie bei SVM-Klassifikation – die Parameter minimiert, so dass zusammen mit C soz. die Regularisierung eingebaut ist.

Formal sieht das wie folgt aus. Wir haben einen Datensatz

$$D \subseteq \mathbb{R}^n, \text{ wobei } D = \{(\vec{x}_i, y_i) : i \in I\}.$$

Das Modell Die Modellfunktion ist im einfachen Falle, wie in der linearen Regression, ein linearer Term:

$$f_{\vec{w},b}(\vec{x}) = \vec{w}^\top \vec{x} + b \tag{525}$$

Die Kosten Welche Kosten wollen wir optimieren? In der linearen Regression ist es

$$K(f(\vec{x}_i, y_i)) = (f(\vec{x}_i) - y_i)^2 \tag{526}$$

In SVR haben wir den Parameter ϵ; die Kosten auf einem Datenpunkt belaufen sich also auf

$$K(f(\vec{x}_i, y_i)) = \begin{cases} (f(\vec{x}) + \epsilon - y_i)^2, \text{ falls } f(\vec{x}) + \epsilon < y_i \\ (f(\vec{x}) - \epsilon - y_i)^2, \text{ falls } f(\vec{x}) - \epsilon > y_i \\ 0 \text{ andernfalls} \end{cases} \tag{527}$$

Also Kosten sind etwas kompliziert. Was wir mit einer SVR zu minimieren suchen ist also folgender Term:

$$\underset{\vec{w}}{argmin}\ \vec{w}^\top \vec{w} + C \cdot \sum_{i \in I} K(f_{\vec{w},b}(\vec{x}_i, y_i)) \tag{528}$$

Das bedeutet: die Kosten werden zugleich mit den Parametern an sich optimiert.

- Die Regressionsfunktion ist also deutlich komplexer als bei der linearen Regression
- Das betrifft einerseits bereits die Kostenfunktion, die invariant ist im Rahmen von $+/-\epsilon$
- Aber hier werden zusätzlich kleine Parameter $\vec{w}$ an sich bevorzugt!
- Nur b ist "frei", da es als Stützvektor keine Korrelation angibt.
- Der Parameter C bestimmt, wie stark die Abweichungskosten massgebend sind.

Das schöne an dieser Regression ist, dass wir die Regularisierung bereits eingebaut haben: wir bevorzugen flache Kurven grundsätzlich.

39 Wiederholung und Grundbegriffe

- Was ist Supervised vs. unsupervised learning (überwacht und unüberwacht) ? Alles was wir soweit besprochen haben ist überwachtes Lernen!
- Man spricht von zwei Arten von Fehlern eines ML-Modell, die Tendenz und die Varianz. Was ist das?
- Was ist Regressionsanalyse, ganz allgemein? Wie wird aus Regression Klassifikation? Und wann kann man keine Regression machen?
- Was ist ein lineares Modell?
- Was bedeutet lineare Separierbarkeit?
- Worin unterscheiden sich SVM und lineare/logistische Regression, sowohl in Klassifikation als auch in Regression?
- Was machen die Methoden .fit, .transform, .predict im Allgemeinen?
- Wir haben Kostenfunktionen besprochen. Wofür braucht man die? Welche zwei verschiedenen Zwecke haben wir besprochen?

What About Bias and Variance? Another approach to analysing learning methods (especially for predicting real-valued quantities) looks at the following two indicators of how well a method predicts some quantity:

Bias How much predictions depart from the truth on average.

Variance The average squared difference of predictions from their average.

The average squared error for the method can be decomposed as the sum of the squared bias and the variance. This leads to a strategy: choose a method that minimizes this sum, possibly trading off increased bias for reduced variance, or vice versa, by adjusting complexity, or introducing some form of regularization. There are two problems with this strategy: The bias and variance depend on the true situation, which is unknown. There is no reason to think that trying nevertheless to minimize squared bias plus variance produces a unique answer. Assessments of bias and variance play no role in the Bayesian approach.

40 Metriken und Kostenfunktionen

Metriken sind, im Gegensatz zu Kostenfunktionen, nicht für das Training, sondern zur Evaluation von Modellen gedacht. Wichtig: Accuracy, precision, recall, F-score.

Accuracy Akkuratheit liefert uns einfach (für Klassifikationsaufgaben). Also einfach folgende Formel:

$$(529)\quad acc(f, D) = \frac{\#\text{ richtige Klassifizierung von } f \text{ auf } D}{|D|}$$

Ganz interessant hierbei ist: wenn wir MSE applizieren auf eine 0,1 Klassifikation, dann bekommen wir genau dasselbe Ergebnis! Accuracy ist aber auf beliebig endlich vielen Klassen eindeutig definiert.

Frage: unter welchen Umständen ist accuracy wenig aussagekräftig?

Precision und Recall Nehmen wir an, wir haben eine Klassifikation nach $\{0, 1\}$. Wir definieren die Precision eines Klassifikators für Klasse 1 als

$$(530)\quad prec(f, D, 1) = \frac{\#\text{ richtige Klassifizierung von 1 in } D}{\#\text{ Klassifizierung von 1 in } D}$$

Precision sagt: wie genau ist unser Klassifikator für label 1, anders gesagt: wenn er 1 ausgibt, wie sicher können wir sein, dass es wirklich 1 ist? Nehmen wir an, wir haben sehr selten das label 1; unser Klassifikator wird immer 0 ausgeben. Er mag eine hohe accuracy haben, aber seine precision wird 0 sein!

Dual dazu wird der Recall definiert als:

$$(531)\quad rec(f, D, 1) = \frac{\#\text{ richtige Klassifizierung von 1 in } D}{\#\ 1 \text{ in } D}$$

Recall sagt: wie sicher ist es, dass unserer Klassifizierer label 1 erkennt, wenn es auftritt? Anders gesagt: wie sicher können wir sein, dass er ein label 1 Datenpunkt richtig einordnet?

Diese Unterscheidung entspricht grob der Unterscheidung von **Typ I** und **Typ II** Fehlern in der Statistik; wir unterscheiden *falsche Negative* von

falschen Positiven. Was uns wichtiger ist, hängt vom Kontext ab. Wichtig: beide Maße sind asymmetrisch, setzen also eine besondere Kategorie voraus, die explizit gemacht werden muss. Damit ist das Maß auch implizit binär – es gibt nur eine Kategorie, die zählt, und die anderen.

F-score Der F-score (auch F1-score) mittelt nun über die beiden:

$$F(f, D, 1) = 2 \cdot \frac{rec(f, D, 1) \cdot prec(f, D, 1)}{rec(f, D, 1) + prec(f, D, 1)} \tag{532}$$

Da Precision und Recall in $[0, 1]$ liegen, folgt dass der Zähler in $[0, 1]$ liegt; der Nenner in $[0, 2]$. Um diese inhärente Asymetrie auszugleichen wir der Term mit 2 multipliziert, also liegt der F-score ebenso in $[0, 1]$ (das kann man recht einfach zeigen).

Der F-score sorgt dafür, dass sowohl Precision als auch Recall berücksichtigt werden; wenn also ein Term 0 oder sehr niedrig ist, ist auch der F-score 0 bzw. sehr niedrig.

Confusion Matrix Die Konfusionsmatrix ist ein beliebtes Werkzeug bei der multi-kategorialen Klassifikation. Nehmen wir an, es gibt Klassen $L = \{l_1, ..., l_n\}$ (l wie label), und unser Klassifikator ist eine Funktion $C : \mathbb{R}^i \to L$. Weiterhin haben wir einen (Test)Datensatz $D \subseteq \mathbb{R}^i \times L$. Die Konfusionsmatrix $KM(C, D) \in \mathbb{R}^{n \times n}$ ($n = |L|$) ist defininert als eine $n \times n$-Matrix wobei

$$KM(C, D)_{m,o} = |\{\vec{x} : (\vec{x}, l_m) \in D, C(\vec{x} = l_o\}| \tag{533}$$

Also intuitiv: die Zeile sagt das korrekte Label (nach D), die Spalte sagt das vorhergesagte Label (nach C). Die Zahlen in der Diagonalen sind die korrekten Vorhersagen. NB: Die Konfusionsmatrix ist nicht invariant unter Transposition:

$$l_m \text{ wird oft klassifiziert als } l_o \not\equiv l_o \text{ wird klassifiziert als } l_m!$$

Die Konfusionsmatrix liefert oft wichtige Einsichten in die *Art von Fehlern*, die ein Klassifikator macht, was wiederum erlaubt den Klassifikator zu verbessern.

Pearson Korrelation Bislang haben wir nur kategoriale Metriken, also Metriken für Klassifikation. Die Pearson-Korrelation dagegen funktioniert für Regression, also für reellwertige Funktionen. Nimm an, man hat zwei Funktionen

$$f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$$

und einen Menge $M \subseteq \mathbb{R}^n$. Wir können, um das ganze etwas intuitiver zu machen, annehmen dass

- f_1 die "Datenfunktion" ist, also $D = \{(\vec{x}, f_1(\vec{x})) : \vec{x} \in M\}$
- f_2 die "gelernte" Funktion ist, also das Ergebnis der Regression.

Wichtig ist, dass wir die Funktionen f_1, f_2, restringiert auf M, als **Zufallsvariablen** betrachten können:

$$P(X_1 = y) := \begin{cases} 1/|\{\vec{x} \in M : f_1(\vec{x} = y\}| \text{ falls es so ein } \vec{x} \text{ gibt} \\ 0 \text{ andernfalls} \end{cases} \tag{534}$$

Wir definieren also die entsprechnden Zufallsvariablen X_1 und X_2. Diese haben natürlich einen Erwartungswert und eine Standardabweichung. Entsprechend können wir sie **standardisieren**:

$$\hat{X}_1(\vec{x}) = \frac{X_1(\vec{x}) - \mathcal{E}(X_1)}{\sigma(X_1)} \tag{535}$$

Wir errinern uns an die (Ko)varianz zweier Variablen:

$$\mathtt{V}(X) = \mathcal{E}((X - \mathcal{E}(X))^2) \tag{536}$$

$$Cov(X_1, X_2) = \mathcal{E}((X_1 - \mathcal{E}(X_1)) \cdot (X_2 - \mathcal{E}(X_2))) \tag{537}$$

Also anstatt die Abweichung vom Erwartungswert zu quadrieren werden die beiden Abweichungen multipliziert. Zuletzt wird der entsprechende Erwartungswert gebildet. Man beachte dass die Kovarianz durchaus negativ sein kein, im Gegensatz zur Varianz! Pearsons Korrelation ist die Kovarianz der standardisierten Variablen, also

$$Korr(X_1, X_2) = \mathcal{E}((\hat{X}_1 - \mathcal{E}(\hat{X}_1)) \cdot (\hat{X}_2 - \mathcal{E}(\hat{X}_2))) \tag{538}$$

Da – qua Konstruktion – $\mathcal{E}(\hat{X}_1) = \mathcal{E}(\hat{X}_2) = 0$, vereinfacht sich das nochmal zu

$$(539)\quad Korr(X_1, X_2) = \mathcal{E}(\hat{X}_1 \cdot \hat{X}_2) = \frac{Cov(X_1, X_2)}{\sigma(X_1)\sigma(X_2)}$$

(die letzte Gleichung folgt aus Satz für die lineare Transformationen von Kovarianzen; Verschiebungen, also Addition/Substraktion, ändern die Kovarianz nicht).

Wir haben die Korrelation von Merkmalen in Datensätzen bereits betrachten (correlation matrix). Hier sehen wir, wie man auch die Korrelation von Werten und Vorhersage (prediction array) als Metrik zur Evaluation nehmen kann.

41 k-means clustering

Zum Abschluss betrachten wir noch eine *clustering-Algorithmus*, was einen Fall von *unsupervised learning* darstellt (unüberwachtes Lernen). Das bedeutet soviel wie: wir haben keine annotierten Modelldaten, sondern nur die rohen Eingaben. Unsere Trainingsdaten unterscheiden sich grundsätzlich nicht von den Daten, die wir später klassifizieren möchten.

Ein clustering-Algorithmus versucht grundsätzlich, eine Menge von Vektoren in verschiedene Gruppen einzuteilen, die *cluster* genannt werden. Wir müssen also prüfen, ob es in unseren Daten ein Muster von Gruppierungen gibt. k-means clustering hat dabei einen Hyperparameter, nämlich die Anzahl der cluster, in die wir die Vektoren gruppieren möchten.

Konzeptuell sieht das wie folgt aus:

- Wir haben einen Datensatz $D \subseteq \mathbb{R}^n$.
- Wir suchen k-Punkte im Raum $\mathbb{R}^n$ (nennen wir C), so dass gilt:

für jeden Datenpunkt $\vec{x} \in D$, finden wir einen zugehörigen Punkt in C, der möglichst nahe liegt. Das bedeutet:

- wir möchten die Summe aller (euklidischen) Abstände von allen Punkten D zum jeweils nächsten Punkt in C minimieren (über $c \in C$).
- Damit können wir jedem Punkt in D einen Punkt in C zuordnen; das wäre dann das zugehörige Cluster.

Wir haben also ein einfaches Optimierungsproblem, das sich wie folgt darstellt: zunächst haben wir eine Zurdnung:

$$c(\vec{x}) = argmin_{\vec{c} \in C} \|\vec{x} - \vec{c}\|_2 \tag{540}$$

Damit lässt sich das Optimierungsproblem einfach beschreiben:

$$C^* = argmin_C \sum_{\vec{x} \in D} \|\vec{x} - c(\vec{x})\| \tag{541}$$

Dennoch ist das Problem etwas komplexer als es aussieht, denn wir müssen ja immer beide Gleichungen optimieren. Das Problem ist in der Tat NP-vollständig und wird normalerweise nur approximiert, allerdings mit Algorithmen, die garantiert konvergieren.

Das Verfahren ist folgendes:

1. Initialisiere die k Punkte $\vec{c}_1, ..., \vec{c}_k$ zufällig.

2. Jeder Datenpunkt wird einem Cluster zugeordnet nach 540; damit haben wir k Mengen: $C_1 = \{\vec{x} : c(\vec{x}) = \vec{c}_1\}$, ..., $C_k = \{\vec{x} : c(\vec{x}) = \vec{c}_k\}$

3. Wir aktualisieren die Punkte $\vec{c}_1$ zu $\vec{c}_1'$ nach folgendem Schema:

   (542) $$\vec{c}_1' = \frac{1}{|C_1|} \sum_{\vec{x} \in C_1} \vec{x}$$

   Das entspricht also der geometrischen Mitte ihres Clusters. Nun gehen wir zurück zu Schritt 2.

Dergestalt wandern also die Mittelpunkte der Cluster, und damit auch die Cluster selbst. Interessant an diesem Algorithmus ist, dass er mit mathematischer Sicherheit konvergiert (also ein lokales Minimum findet); aber nicht unbedingt das globale Optimum. Übrigens hat er eine starke Ähnlichkeit zu den berühmten EM-Algorithmen (expectation-maximization), die sehr wichtig im semi-überwachten machine learning sind.

Vorteile von k-means Der große Vorteil von k-means ist: er ist einfach! Insbesondere ist er eine **lineare Methode** (im weiteren Sinne: wenn wir zwei Punkte haben und den Raum trennen danach, welcher Punkt am nächsten ist, dann ergibt sich immer eine Hyperebene als Trennlinie. Das generalisiert sich auf beliebig viele Punke: $n+1$ Punkte bedeutet: $f(n)$ Hyperebenen (was ist f? gar nicht so einfach!). Man hat also jedenfalls ein klares Verständnis davon was passiert!

Nachteile von k-means k-means ist eine lineare Methode, wie gesagt. Wenn also keine lineare Separierbarkeit von clustern gegeben ist, schneidet sie schlecht ab.

Es gibt aber noch einen weiteren Nachteil: cluster werden allein aufgrund ihres Mittelpunktes erstellt. Das bedeutet z.B.: wenn wir zwei Haufen a,b haben, a mit *weiter Streuung*, b mit *enger Streuung*, werden evtl. Punkte von a dem Cluster von b zugeordnet werden, auch wenn sie weit abseits des Haufens b liegen, und ihre nächsten Nachbarn eindeutig zu a gehören. Für solche Eigenschaften ist k-means blind, es berücksichtigt nur Mittelpunkte, sonst nichts. Dafür bräuchte man komplexere Methoden, wie etwa Gaussian Mixture.

Hausaufgabe

Schreiben Sie einen Pseudo-Code-Algorithmus für k-means clustering. Es muss nicht kompilierbarer Code sein, aber sollte trotzdem alle Details der Berechnung berücksichtigen.

42 “Curse of Dimensionality”

Wenn wir Daten sammeln, haben wir oft das Problem hoher Dimensionalität. Das kann an mit zwei einfachen Beispielen belegen: Viele ML-Anwendungen haben ein Problem mit hohen Dimensionen. Hierbei gibt es drei große Probleme:

1. Viele Algorithmen laufen sehr langsam in hohen Dimensionen (entspricht vielen Parametern)
2. Daten in hohen Dimensionen sind grundsätzlich schwer zu visualisieren – viele grundlegende Muster erkennt man aber am besten in einer Visualisierung.
3. In hochdimensionalen Räumen sind Daten immer recht *dünn*; das wiederum bedeutet, dass Generalisierungen problematisch sind.

Punkt 1 und 2 sollten relativ klar sein. Zu Punkt drei einige Beobachtungen. Nimm zwei Punkte im Intervall $[0, 1]$. Was ist die durchschnittliche (euklidische) Distanz zweier solcher Punkte? Das berechnet sich durch

$$\int_0^1 \int_0^1 |x - y| dx \; dy = 2 \int_0^1 \int_0^1 x - y \; dx \; dy = \frac{1}{3} \tag{543}$$

Im Einheitsquadrat ist das ganze schon etwas komplizierter.

Die (einfache) Distanz zweier Punkte ist definiert durch:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \tag{544}$$

Die durchschnittliche Distanz wäre:

$$\int_0^1 \int_0^1 \int_0^1 \int_0^1 d(p, q) = \int_0^1 \int_0^1 \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} dp_1 \; dp_2 \; dq_1 \; dq_2 \approx 0.52 \tag{545}$$

Mit steigender Anzahl von Dimensionen wächst die Größe des Integrals. Nun die Frage: wie sieht die durchschnittliche Distanz im hochdimensionalen Raum, sagen wir mit 1,000,000 Dimensionen? Die Antwort lautet: zwei

zufällig ausgewählte Punkte haben im Durchschnitt eine euklidische Distanz von 408.25 – im Einheitshyperquadrat, also Seitenlänge 1!

Wir haben also im Normalfall relativ große Entfernungen von Punkten, obwohl die Entfernungen in jeder Dimension recht klein ist – einfach weil es viele Dimensionen gibt. Nun ist natürlich klar dass dünn gelagerte Daten (große Abstände zwischen Datenpunkten) immer problematisch sind, v.a. für komplexe Lernalgorithmen (wie z.B. in neuronalen Netzen), da eine starke Gefahr von Overfitting besteht. Also:

- Wir möchten grundsätzlich nur ungern mit Daten hoher Dimensionalität arbeiten.
- Das Ziel in allen Anwendungen mit hochdimensionalen Daten muss also sein, deren Dimensionalität zu reduzieren.

43 Projektionen

Die einfachste Art und Weise, Dimensionalität zu reduzieren ist mittels **Projektion**. Eine Projektion ist eine einfache Abbildung

$$\pi_i : \mathbb{R}^n \to \mathbb{R}^{n-1},$$

wobei $i \leq n$. Es wird einfach eine Dimension wegprojiziert, nämlich die i-te.

Das ist die einfachste Art die Dimensionalität zu reduzieren, aber leider eine Art, bei der viel Information verloren gehen kann. Betrachten wir erstmal, unter welchen Bedingungen Projektion gut funktioniert. Nehmen wir an, alle Datenpunkte liegen in einer (Hyper-)ebene, *und* die Ebene liegt parallel zu einer Achse, dann können wir die zugehörige Dimension wegprojizieren, ohne etwas zu verlieren.

Nehmen wir dagegen an, die Daten liegen in einer $n-1$-dimensionalen Hyperebene. Wenn wir nun eine Dimension wegprojizieren, dann werden die Daten *verzerrt*. Man kann sich jetzt überlegen, welche Dimension die geringste Verzerrung verursacht; außerdem kann man die Verzerrung mit einer einfachen linearen Abbildung neutralisieren; das gehört aber bereits zu PCA.

44 Mannigfaltigkeiten

Man kann oftmals beobachten, das Punkte in einer "Ebene" liegen, aber die Ebene ist nicht eben, sondern z.B. gerundet. Wenn wir an unsere Erde denken, haben wir (vor GPS) Längen und Breitengrade für jeden Punkt der Erdoberfläche, also nur 2 Koordinaten, obwohl wir uns ja gerade *nicht* in einer Ebene befinden, sondern auf einer (durchaus nicht wirklich runden) Kugel im 3D-Raum.

Ein Gebilde, in dem man im n-dimensionalen Raum jeden Punkt mit $n-1$ Koordinaten bestimmen kann, nennt man eine **Mannigfaltigkeit**. Typische Mannigfaltigkeiten sind:

- Kugeln
- Zylinder
- Der "Rettungsring"
- Ein "gewellter Schal"
- ...

Eine Mannigfaltigkeit ist **differenzierbar**, wenn es an jedem Punkt höchstens eine eindeutige Tangente gibt, die die Mannigfaltigkeit an diesem Punkt berührt, aber sie nicht schneidet (aber womöglich an einem entfernteren Punkt).

Je nachdem wie komplex eine Mannigfaltigkeit ist, lässt sich die Dimension einfach oder schwierig reduzieren.

- Wenn die Daten auf einer Mannigfaltigkeit liegen, sollte das aber *im Prinzip* weitgehend verlustlos möglich sein!
- Man nehme das Beispiel des Globus: wir können die Winkel oder Flächengröße präservieren, aber nicht Winkel und Flächengrößen (üblicherweise macht man einen Kompromiss, aber Regionen nah an den Polen werden dabei "aufgebläht")
- Das Problem ist, die entsprechende Abbildung zu finden, wenn wir die Mannigfaltigkeit nicht kennen!

Im Normalfall werden wir eine solche Mannigfaltigkeit nicht haben, aber eine Visualisierung zeigt normalerweise ganz gut, ob die Daten annähernd in einer solchen Mannigfaltigkeit liegen. Das Problem ist: nur niedrigdimensionale Daten lassen sich visualisieren!

45 Locally Linear Embeddings

Locally linear embeddings (LLE) gehören zum Bereich des Manifold learnings, also Reduktion auf Basis von (nicht-linearen) Mannigfaltigkeiten. Falls die Daten auf einer solchen Mannigfaltigkeit liegen, funktioniert die Reduktion mit LLE sehr gut, anderfalls eher weniger.

Das Vorgehen ist folgendes: wir haben gegeben

- einen Datensatz $D \in \mathbb{R}^m$, sowie
- einen Hyperparameter k.

Als ersten Schritt suchen wir nun die k nächsten Nachbarn für alle $\vec{x} \in D$; zugleich versuchen wir, $\vec{x}$ darzustellen als eine lineare Funktion dieser Nachbarn: seien

$$\vec{n}_1^x, ..., \vec{n}_k^x$$

die k nächsten Nachbarn von $\vec{x}$. Wir möchten jeden Punkt $\vec{x}$ mit seinen Nachbarn rekonstruieren, also suchen wir **quadratische** (!) Matrizen $W_1, ..., W_k$ so dass

$$rec(\vec{x}) = \sum_{i=1}^{k} W_i \vec{n}_i^x \tag{546}$$

Hier gibt es eine weitere Beschränkung:

- jede Zeile von W_i soll sich auf 1 summieren (also nicht die Norm, sondern wirklich die Summe).
- Also ist jede Rekonstruktion eine gewichtete Summe der Nachbarn

die Matrizen W_i findet man, indem man ein klassisches Optimierungsproblem löst:

$$Fehler(\vec{x}) = \sum_{\vec{x} \in D} \|\vec{x} - rec(\vec{x})\|_2 \tag{547}$$

Man beachte dass die Matrizen W_i invariant sind gegenüber Rotationen und anderen linearen Transformationen!

Nachdem wir die **Nachbarschaftsrekonstruktion** haben, können wir anfangen, die Dimension zu reduzieren. Wir suchen diejenigen

$$\vec{y} \in \mathbb{R}^{m'}, \text{ wobei } m' < m,$$

die den Rekonstruktionsfehler ebenfalls minimieren:

$$(548)\ \ Fehler(\vec{x}) = \sum_{\vec{y} \in D'} \|\vec{y} - rec(\vec{y})\|_2$$

wobei die Rekonstruktion wie oben definiert ist, und die Matrizen W die oben konstruierten sind:

$$(549)\ \ red(\vec{x}) = \sum_{i=1}^{k} W_i' \vec{n}_k^x$$

hier gilt natürlich:

$$W_i' \in \mathbb{R}^{m' \times m},\ m' < m$$

Wir müssen noch die Rekonstruktion definieren: wir haben haben ja $D \subseteq \mathbb{R}^{m'}$, $W \in \mathbb{R}^{m' \times m}$. Daraus folgt:

$$(550)\ \ W^\top W \in \mathbb{R}^{m \times m}$$

Und wir können also $\vec{x}$ rekonstruieren als

$$(551)\ \ rec(\vec{x}) = \sum_{\vec{y} \in neigh(\vec{x})} W^\top W \vec{y}$$

Und wir versuchen den Fehler zu minimieren:

$$(552)\ \ argmin_{W_i} \sum_{\vec{x}_i \in D} \|\vec{x} - W_i^\top W_i \vec{x}_i\|_2$$

Das liefert uns dann diejenigen W, die das ganze global minimieren. Die Optimierung läuft – glaube ich – iterativ und beinhaltet einige Schritte die mir nicht ganz klar sind.

So zumindest findet man das in der Literatur. Es bleiben mir aber Dinge unklar. Ich denke:

- Die niedrigere Dimension $W_i \in \mathbb{R}^{m' \times m}$, $m' < m$ kann man sich vorstellen als eine Art PCA;
- nur dass sie eben lokal durchgeführt wird, nicht auf dem ganzen Datensatz!

So liefert uns jedes

$$(553)\ \vec{x}' = \sum_{\vec{y} \in neigh(\vec{x})} W\vec{y}$$

eine Reduktion! So ist die Rekonstruktion aus der Reduktion bereits eindeutig definiert.

Das verstehe ich und es ergibt einen Sinn – aber es ist nicht das was ich lese.

46 Principal component analysis

46.1 Kovarianz und Varianz

Wir haben Datenpunkte $D \subseteq \mathbb{R}^n$, dementsprechend Variablen/Merkmalen

$$X_1, ..., X_n$$

Bei PCA geht es grob gesagt darum, herauszufinden, welche dieser Variablen für die meiste Varianz zuständig ist; genauer gesagt:

> Welche lineare Kombination von Variablen hat die maximale Varianz?

Bildlich gesprochen: Es geht darum, den Datensatz so zu rotieren, dass möglichst viel Varianz entlang der Achsen des Raumes läuft.

Ist das erledigt können wir als nächsten Schritt dann mittels Projektionen den n-dimensionalen Raum einbetten in einen m-dimensionalen Raum ($m < n$), und dabei möglichst wenig Information zu verlieren.

Das Verfahren basiert auf Matrizen, da D dargestellt werden kann als eine Matrix

$$M \in \mathbb{R}^{|D| \times n}$$

– also jede Zeile ist ein Datenpunkt, jede Spalte die Werte einer Variable. Wir brauchen also erstmal einige Begriffe zu Matrizen: die Varianz und die Kovarianz.

Ein **Zufallsvektor** ist die Generalisierung einer Zufallsvariable, hat die Form $(X_1, ..., X_n)$, mit

- den Erwartungswerten $\mathcal{E}(X_i) = \mu_i$,
- den Varianzen $\mathtt{V}(X_i) = \mathcal{E}((X_i - \mathcal{E}(X_i))^2) = \sigma_i^2$.

Der Erwartungswert des Vektors ist also wiederum ein Vektor

$$\mu = (\mu_1, ..., \mu_n) \tag{554}$$

Man definiert die Kovarianz zweier Variablen als

$$Cov(X, Y) = \mathcal{E}[(X - \mathcal{E}(X)) \cdot (Y - \mathcal{E}(Y))] \tag{555}$$

Also anstatt die Abweichung vom Erwartungswert zu quadrieren werden die beiden Abweichungen multipliziert. Zuletzt wird der entsprechende Erwartungswert gebildet. Man beachte dass die Kovarianz durchaus negativ sein kein, im Gegensatz zur Varianz!

Interpretation der Kovarianz:

- Die Kovarianz zweier Variablen X, Y ist positiv, wenn die beiden eine positive Korrelation haben: je gröss er X, desto grösser Y und umgekehrt. Dabei gilt: je mehr, desto stärker.
- Die Kovarianz zweier Variablen X, Y ist negativ, wenn die beiden eine negative Korrelation haben: je grösser X, desto kleiner Y und umgekehrt. Dabei gilt: je mehr, desto stärker.
- Ist die Kovarianz sehr nahe bei 0, dann gibt es keine lineare Korrelation.

Verschiebungssatz für Kovarianz:

(556) $Cov(X,Y) = \mathcal{E}(XY) - \mathcal{E}(X)\mathcal{E}(Y)$

Das bedeutet: die Kovarianz ist der Erwartungswert der verbundverteilung minus das Produkt der Erwartungswerte der Einzelverteilungen. Ebenso gilt natürlich folgendes:

(557) $Cov(X,X) = \mathtt{V}(X)$

Das heisst: die Kovarianz einer Variable mit sich selbst ist nichts anderes als die Varianz. Das sollte reichen um die Kovarianzmatrix zu definieren. Nehmen wir eine Matrix

$$\begin{pmatrix} a_{11} & a_{12} & .... & a_{1n} \\ & ... & & \\ a_{k1} & ... & & a_{kn} \end{pmatrix}$$

Die repräsentiert k Datenpunkte und n Zufallsvariablen $X_1, ..., X_n$. Die Kovarianzmatrix ist definiert als

$$\begin{pmatrix} Cov(X_1,X_1) & Cov(X_1,X_2) & ... & Cov(X_1,X_n) \\ & ... & & \\ Cov(X_n,X_1) & Cov(X_1,X_2) & ... & Cov(X_n,X_n) \end{pmatrix}$$

Das bedeutet: wir haben eine $n \times n$ Matrix die symmetrisch ist; die Diagonale liefert uns die jeweiligen Varianzen. Intuitiv will man diejenigen Variablen haben, die für die meiste Varianz verantwortlich sind, also unereinander möglichst unabhängig sind.

Definition 23 *Eine Matrix M ist **orthogonal**, falls*

1. *M quadratisch ist und*
2. *$M^\top M = 1_M$, also die Einheitsmatrix der entsprechnenden Dimensionen ist.*

Orthogonale Matrizen haben folgende wichtige Eigenschaften:

- Dadurch, dass sie quadratisch sind, können wir sie auffassen als Abbildungen $\mathbb{R}^n \to \mathbb{R}^n$; Vektoren behalten also ihre Dimension.
- Sei $\| - \|$ eine Norm, die die Länge des Vektors angibt. Dann gilt (für orthogonales M): $\|M\vec{x}\| = \|\vec{x}\|$. Sprich: die Länge eines Vektors wird nicht verändert!
- Wir haben $\vec{x}^\top \vec{y} = (M\vec{x})^\top (M\vec{y})$. Das bedeutet: auch das Produkt bleibt unverändert. Daraus folgt:
- $cos(\vec{x}, \vec{y}) = \frac{\vec{x}^\top \vec{y}}{\|\vec{x}\| \cdot \|\vec{y}\|} = cos(M\vec{x}, M\vec{y})$
- Das bedeutet: Multiplikation mit M präserviert auch die Winkel zwischen Vektoren.

Also, alles Zusammengefasst: gegeben $D \subseteq \mathbb{R}^n$, ist

$$\{M\vec{x} : \vec{x} \in D\} \text{ eine } \textbf{Rotation} \text{ von } D\text{:}$$

Alle Abstände aller Punkte bleiben gleich, ebenso die Abstände vom Ursprung.

Definition 24 *Die **Hauptachsentransformation** von D ist gegeben durch die orthogonale Matrix, die aus den Eigenvektoren der Kovarianzmatrix von D gebildet wird.*

Wir müssen zunächst den Begriff des **Eigenvektors** klären: Ein Eigenvektor einer Matrix ist in der linearen Algebra ein vom Nullvektor verschiedener Vektor, dessen Richtung durch die (der Matrix entsprechenden) Abbildung nicht verändert wird. Ein Eigenvektor wird also nur skaliert und man bezeichnet den Skalierungsfaktor als Eigenwert der Abbildung.

Definition 25 *Eigenvektoren einer Matrix, d.h. linearen Abbildung sind Vektoren, die durch diese Abbildung nicht ihre Richtung ändern, sondern nur Skaliert werden; gegeben M, suche das $\vec{v}$ so dass es ein λ gibt mit*

$$M\vec{v} = \lambda\vec{v}$$

$\vec{v}$ ist der Eigenvektor, λ nennt man den Eigenwert von M. Nachdem jede Matrix eine lineare Abbildung darstellt, hat jede Matrix einen Eigenvektor.

Geometrisch kann man das sich so vorstellen: eine lineare Abbildung ist eine verzerrte Verschiebung; aber für jede einfache Verzerrung gibt es (mindestens) eine Linie, die sich nur in der Länge ändert. Das finden von Eigenvektoren ist schwierig und funktioniert im allgemeinen Fall nur mittels Näherung.

46.2 PCA mittels Eigenzerlegung, SWZ

Eigenzerlegung der Kovarianzmatrix Die Kovarianzmatrix einer Datenmatrix $\mathbf{X}$ mit $n \times p$ Dimensionen (k die Anzahl der Daten, n Anzahl der Merkmale) lässt sich sehr einfach erstellen:

(558) $cov(\mathbf{X}) = \mathbf{X}^\top \mathbf{X} \ / \ k - 1$

Diese Matrix $cov(\mathbf{X})$ ist quadratisch, hat also eine Eigenzerlegung. Eine Eigenzerlegung von $cov(\mathbf{X})$ besteht aus drei Matrizen $\mathbf{V}, D(\lambda), \mathbf{V}^\top$, wobei

1. $cov(\mathbf{X}) = \mathbf{V} \cdot D(\lambda) \cdot \mathbf{V}^\top$
2. In $\mathbf{V}$ ist jede Spalte ein Eigenvektor von $cov(\mathbf{X})$
3. $D(\lambda)$ ist eine Diagonalmatrix, d.h. alle Einträge außerhalb der Diagonale sind 0, und die Diagonaleinträge sind die Eigenwerte von $cov(\mathbf{X})$

Die Spalten von $\mathbf{V}$ (in ihrer Reihenfolge) sind dann genau die Hauptkomponenten von $\mathbf{X}$. Erstaunlich!

Das Problem ist, die Eigenzerlegung einer Matrix sehr aufwändig zu berechnen ist, und oft nur näherungsweise berechnet werden kann. Oftmals nutzt man aus praktischen Erwägungen die Singulärwertzerlegung anstelle der Eigenzerlegung.

Eigenzerlegungen gibt es nur für quadratische Matrizen; Singulärwertzerlegungen gibt es für jede Matrix. Daher ist das Verfahren wichtiger, da es allgemein angewendet werden kann.

Singulärwertzerlegung Wir nehmen hier an, dass der Mittelwert von $\mathbf{X}$ $\mathbf{0}_V$ ist (das kann man leicht sicherstellen indem man den Mittelwert subtrahiert). Man kann die Hauptkomponente berechnen mit der **Singulärwertzerlegung**. Die ist wie folgt definiert:

Definition 26 *$\boldsymbol{UDV}^\top$ ist eine SWZ von $\boldsymbol{X}$, falls gilt:*

1. $\boldsymbol{X} = \boldsymbol{UDV}^\top$
2. $\boldsymbol{U}$, $\boldsymbol{V}$ *sind orthogonale Matrizen*
3. *D ist eine Diagonalmatrix (nur die Diagonaleinträge sind ungleich 0), nicht notwendig quadratisch*

Die Einträge von $\boldsymbol{D}$ (entlang der Diagonale) heißen die Singulärwerte von $\boldsymbol{A}$; die Spalten von $\boldsymbol{U}$ bzw. $\boldsymbol{V}$ heißen die linken bzw. rechten Singulärvektoren.

Diese Zerlegung kann direkt auf $\mathbf{X}$ (anstelle von $cov(\mathbf{X})$) appliziert werden.

- Man bekommt die Hauptkomponenten von $\mathbf{X}$ als die Zeilen von $V^\top$ (also die Spalten von V)
- Die Rotation der Daten bekommt man also mittels $\mathbf{X} \cdot V$
- Die Reduktion der Dimension auf i bekommt man durch $\mathbf{X} \cdot V[:, 0:i]$
- Die Rekonstruktion der Daten bekommt man durch $\mathbf{X} \cdot V[:, 0:i] \cdot V[:, 0:i]^\top$

Man beachte: die Rekonstruktion ist genau dann fehlerfrei/exakt, wenn die Daten $\mathbf{X}$ in einer Hyperebene liegen.

46.3 Geometrische Darstellung

Nehmen wir an, wir haben zwei Variablen (also eine Ebene). Eine PCA liefert uns nun zwei orthogonale Geraden, also praktisch ein neues Koordinatensystem. Dieses neue Koordinatensystem hat zwei Eigenschaften: die erste Achse ist so gezogen, dass entlang ihr die **maximale Varianz** liegt. Die Varianz eines Datensatzes haben wir bereits besprochen. Wir definieren zunächst

$$(559)\quad mittel(D) = \frac{1}{|D|}\sum_{\vec{x}\in D}\vec{x}$$

Hiermit lässt sich nun die Varianz des Datensatzes definieren:

$$(560)\quad var(D) = \frac{1}{|D|}\sum_{\vec{x}\in D}\|\vec{x} - mittel(D)\|_2^2 \in \mathbb{R}$$

also im Falle $D \subseteq \mathbb{R}$ haben wir die Abweichung zum Mittelwert im Quadrat. Das gute ist dass diese Methode sich auf beliebig dimensionale Daten verallgemeinert. Wir können also von jedem Datensatz seine Varianz bestimmen.

Das gilt natürlich auch von jeder Projektion des Datensatzes: sei π_i eine Projektion, dann verlieren wir in $\pi_i(D)$ natürlich etwas Varianz, denn in einer Dimension werden alle Entfernungen annulliert. Wir haben also

$$(561)\quad var(\pi_i(D)) \leq var(D)$$

Wieviel Varianz genau wir verlieren ist ein gutes und geläufiges Maß dafür, ob eine Projektion gut oder schlecht ist. Hierfür nehmen wir eine etwas andere Perspektive ein; anstatt Dimensionen *weg*zuprojizieren nutzen wir inverse Projektion, die alles bis auf eine Dimension wegprojizieren. Wir schreiben also, für $D \in \mathbb{R}^n$,

$$(562)\quad \overline{\pi}_i(D) = \{x_i : (x_1, ..., x_i, ..., x_n) \in D\}$$

Wir können nun sagen: die i-te Dimension **erklärt** x Prozent der Varianz, falls

$$(563)\quad \frac{var(\overline{\pi}_i(D))}{var(D)} \cdot 100 = x$$

Die erste Achse wäre also in unserem 2D Fall die Gerade durch die Daten mit der meisten Varianz; die zweite Achse die (eindeutige) Orthogonale.

Als nächstes nehmen wir den 3-Dimensionalen Fall, also $D \subseteq \mathbb{R}^3$. Wir suchen nun widerum zunächst eine Gerade, die durch die Daten geht, nämlich diejenige, entlang derer wir die meiste Varianz finden. In diesem Fall ist aber die zweite Hauptkomponente nicht allein durch Orthogonalität definiert; es ist die eine Gerade, die

1. senkrecht auf der anderen Gerade steht, und
2. von allen diesen Geraden die Varianz maximiert

Un so geht es weiter. Man beachte dabei folgendes: bei n Dimensionen macht es daher durchaus Sinn, die n Hauptkomponenten zu finden – in diesem Fall haben wir aber keine Kompression, sondern einfach eine **Rotation** des Raumes, also der verschiedenen Dimensionen. Wir rotieren also die Dimensionsachsen so, dass alle Datenpunkt möglichst nahe daran liegen.

Das finden der Hauptkomponenten ist **monoton**, d.h. wenn wir $i + 1$ Hauptkomponente suchen, dann suchen wir zunächst die i Hauptkomponenten, und dann auf Basis davon die $i + 1$te Hauptkomponente.

46.4 Algebraische Darstellung und Kodierung

Kodierung und lineare Rekonstruktion Nehmen wir einmal an, wir möchten unsere

$$\text{Daten } D \subseteq \mathbb{R}^n$$

kodieren, und zwar in einem Vektorraum mit *weniger Dimensionen*. Das würde die Darstellung natürlich kompakter machen, denn insbesondere in der distributionellen Semantik/word embeddings gibt es sehr hochdimensionale Vektorräume, was oft ein Problem darstellt.

Andererseits ist klar, dass falls $i < n$, im Normalfall jede Abbildung

$$f : D \to \mathbb{R}^i$$

in einem gewissen Maß Information *verliert*. Man kann dieses Konzept wie folgt definieren. Nehmen wir an, wir haben eine solche Abbildung f. Gibt es dann eine Abbildung g so dass

$$g(f(d)) = d \tag{564}$$

für alle $d \in D$? Im allgemeinen Fall nicht. Was wir aber machen können ist folgendes: wir suchen eine Abbildung f für die es ein g gibt so dass

$$g(f(d)) \approx d \tag{565}$$

Wir suchen also dasjenige f, welches das am besten leistet. Wichtig ist hier aber eine Einschränkung: g soll eine lineare Abbildung sein, also

$$g(\vec{d}) = M\vec{d} \tag{566}$$

wobei

1. $M \subseteq \mathbb{R}^{n \times i}$ eine Matrix ist, und
2. zuätzlich die Spalten on M, als Vektoren, orthogonal zueinander stehen.

Zur Erklärung: eine Matrix hat orthogonale Spalten, wenn für beliebige Spaltenvektoren $\vec{v}_i, \vec{v}_j$ gilt:

$$\vec{v}_i^\top \vec{v}_j \neq 0 \text{ gdw. } i = j$$

Das bedeutet soviel wie: alle Vektoren stehen orthogonal zueinander. Man sagt eine **Matrix ist orthogonal**, falls alle Spalten orthogonal sind und M quadratisch ist. M oben wird aber nicht quadratisch sein, denn M soll ja eine Abbildung g modellieren,

$$g : \mathbb{R}^i \to \mathbb{R}^n,\ n > i$$

das ist ja der Punkt der ganzen Sache (wenn wir alle Hauptkomponenten suchen ohne Dimensionsreduktion, dann haben wir eine orthogonale Matrix).

Skalierung und Normalisierung Weiterhin möchten wir, wie bereits bei SVM, das Problem der Skalierung umgehen um eine eindeutige Abbildung zu bekommen. Wir verlangen daher dass für alle Spaltenvektoren

$$\vec{v}_i = M_{:,i}$$

gilt dass

(567) $\|\vec{v}_i\| = 1$

Also die Norm soll die Einheit sein (das lässt sich leicht machen: dividiere jeden Vektor punktweise mit seiner Norm). Damit können wir sicher sein, dass unsere Matrix M, gegeben die Kodierung f, eindeutig bestimmt ist. Eine solche Matrix hat die Eigenschaft dass gilt:

(568) $M^T M = 1_M$

(das ist sogar äquivalent zu Orthorgonal& Norm 1) Also die Matrix, mit sich selbst multipliziert, ergibt die Einheitsmatrix. Was wir nun suchen ist der Code von x, von jedem Datenpunkt x. Der ist wie folgt spezifiert:

(569) $code(\vec{x}) = \underset{\vec{c}}{argmin} \|\vec{x} - g(\vec{c})\|_2$

Da die euklidische Distanz positiv ist und die quadrat-Funktion streng monoton, können wir ebenso die quadratische euklidische Distanz nehmen:

(570) $code(\vec{x}) = argmin_{\vec{c}} \|\vec{x} - g(\vec{c})\|_2^2$

wobei

(571) $\|\vec{x} - g(\vec{c})\|_2^2 = (\vec{x} - g(\vec{c}))^\top (\vec{x} - g(\vec{c}))$

Jetzt kann man folgende Transformationen vornehmen (binomische Formel!)

$$(\vec{x} - g(\vec{c}))^\top(\vec{x} - g(\vec{c})) = \vec{x}^\top\vec{x} - \vec{x}^\top g(\vec{c}) - g(\vec{c})^\top\vec{x} + g(\vec{c})^\top g(\vec{c}) \tag{572}$$

$$= \vec{x}^\top\vec{x} - 2\vec{x}^\top g(\vec{c}) + g(\vec{c})^\top g(\vec{c}) \tag{573}$$

(binomische Formel, da $\vec{x}^\top g(\vec{c}) = g(\vec{c})^\top\vec{x}$). Diese Funktion kann man wieder vereinfachen, denn $\vec{x}^\top\vec{x}$ ist unabhängig von $\vec{c}$. Wir suchen also

$$\underset{\vec{c}}{argmin}(-2\vec{x}^\top g(\vec{c}) + g(\vec{c})^\top g(\vec{c})) \tag{574}$$

Jetzt setzen wir die Definition von g ein:

$$code(\vec{x}) = \underset{\vec{c}}{argmin} - 2\vec{x}^\top M\vec{c} + (M\vec{c})^\top M\vec{c} \tag{575}$$

$$= \underset{\vec{c}}{argmin} - 2\vec{x}^\top M\vec{c} + \vec{c}^\top M^\top M\vec{c} \tag{576}$$

$$= \underset{\vec{c}}{argmin} - 2\vec{x}^\top M\vec{c} + \vec{c}^\top\vec{c} \tag{577}$$

Dieses Problem lässt sich mittels Gradienten lösen: wir brauchen

$$\nabla_{\vec{c}} - 2\vec{x}^\top M\vec{c} + \vec{c}^\top\vec{c} = 0 \tag{578}$$

was wiederum bedeutet:

$$-2M^\top\vec{x} + 2\vec{c} = 0 \tag{579}$$

$$\vec{c} = M^\top\vec{x} \tag{580}$$

Das lässt sich wiederum effizient lösen. Aber dies gilt nur für ein festgelegtes M, was wir ja nicht haben! Zunächst schauen wir uns an, wie wir, gegeben eine Dekodierungsfunktion M, den Vektor $\vec{x}$ kodieren. Wir definieren

$$f(\vec{x}) = M^\top\vec{x} \tag{581}$$

Weiterhin haben wir dann die Rekonstruktion r, definiert wie folgt:

(582) $r(\vec{x}) = g(f(\vec{x})) = MM^\top \vec{x}$

Wir suchen nun die **optimale Matrix** M^*. Das ist nun folgendes Optimierungsproblem:

(583) $M^* = argmin_M \sqrt{\sum_{i,j} (x_{ij} - r(\vec{x}_i)_j)^2}$

unter Bedingung das

(584) $M^\top M = 1_M$

(diese Seitenbedingung ist äquivalent zu den beiden Bedingung von orthogonalen Spalten und Norm von 1 in allen Spalten) Das ist also wieder ein Optimierungsproblem mit Seitenbedingung, das sich einigermaßen gut lösen lässt (aber nicht mit elementaren Mitteln)

46.5 Dimensionen reduzieren mit PCA – Varianz

Wenn wir PCA in einem n-dimensionalen Datensatz D ausführen, dann können wir das nutzen, um die Dimensionalität beliebig zu reduzieren: wir nehmen die Matrix $M^* \in \mathbb{R}^{l \times n}$, wobei $l < n$, und bilden unseren Datensatz

$$D' = \{M^* \vec{x} : \vec{x} \in D\} \tag{585}$$

Das ist geradeaus und kein Problem mehr. Die neue Frage ist:

Welche Anzahl von Dimensionen ist die richtige? Diese Frage lässt sich wie folgt beantworten: wir können für jede Reduktion prüfen, für wieviel Varianz der Daten sie aufkommt. Die Varianz eines Datensatzes haben wir bereits besprochen. Wir definieren zunächst

$$mittel(D) = \sum_{\vec{x} \in D} \frac{\vec{x}}{|D|} \tag{586}$$

Hiermit lässt sich nun die Varianz des Datensatzes definieren:

$$var(D) = \sum_{\vec{x} \in D} \frac{\|mittel(D) - \vec{x}\|_2}{|D|} \tag{587}$$

also im Falle $D \subseteq \mathbb{R}$ haben wir die Abweichung zum Mittelwert im Quadrat. Das gute ist dass diese Methode sich auf beliebig dimensionale Daten verallgemeinert. Wir können also von jedem Datensatz seine Varianz bestimmen.

Das gilt natürlich auch von jeder Projektion des Datensatzes: sei π_i eine Projektion, dann verlieren wir in $\pi(D)$ natürlich etwas Varianz, denn in einer Dimension werden alle Entfernungen annulliert. Wir haben also

$$var(\pi_i(D)) \leq var(D) \tag{588}$$

Wieviel Varianz genau wir verlieren ist ein gutes und geläufiges Maß dafür, ob eine Projektion gut oder schlecht ist. Hierfür nehmen wir eine etwas andere Perspektive ein; anstatt Dimensionen *weg*zuprojizieren nutzen

wir inverse Projektion, die alles bis auf eine Dimension wegprojizieren. Wir schreiben also, für $D \in \mathbb{R}^n$,

$$\overline{\pi}_i(D) = \{x_i : (x_1, ..., x_i, ..., x_n) \in D\} \tag{589}$$

Wir können nun sagen: die i-te Dimension **erklärt** x Prozent der Varianz, falls

$$\frac{var(\overline{\pi}_i(D))}{var(D)} \cdot 100 = x \tag{590}$$

Dadurch dass die verschiedenen Dimensionen orthogonal zueinander liegen ist klar, dass jede Dimension für einen anderen Anteil der Varianz aufkommt, also:

$$\sum_{i=1}^{n} \frac{var(\overline{\pi}_i(D))}{var(D)} = 1 \tag{591}$$

Was hat das jetzt mit PCA zu tun? Die Antwort hierauf lautet:

- PCA liefert uns diejenigen Dimensionen mit der meisten Varianz;
- PCA liefert uns eine natürlich Ordnung der Dimensionen, so dass wir einfach nacheinander projizieren können.
- Die Hauptkomponenten sind orthogonal zueinander; d.h. der Anteil der Varianzen summiert sich auf 1!
- Da alle Hauptkomponenten die Norm 1 haben, wird die originale Varianz der Daten beibehalten, d.h. weder verringert noch vergrößert.

Das interessante ist hierbei, dass die Dimensionen der PCA keineswegs mit den ursprünglichen Dimensionen übereinstimmen – wir haben ja eine Rotation der Daten. Intuitiv bedeutet dass: PCA berücksichtigt bereits interne (lineare) Korrelationen zwischen einzelnen Dimensionen. PCA liefert uns hier also die jeweils optimalen Ergebnisse.

Nun also zurück zu unserer Frage: auf wieviele Dimensionen können wir reduzieren? Die Antwort lautet hier wieder: legen wir zunächst eine Art

"Vertrauensgrenze" fest, z.B. 95%. Was wir als nächstes machen: wir nennen die j Hauptkomponenten (wobei $j > n$), so dass

$$(592)\quad \sum_{i=1}^{j} \frac{var(\overline{\pi}_i(D))}{var(D)} \geq 0.95$$

In diesem Sinne können wir PCA nutzen, bis wir 95% der Varianz erklärt haben; wieviele Dimensionen wir dafür brauchen hängt natürlich von den Daten ab!

46.6 PCA und Klassifikation/Regression

Nehmen wir an, wir haben einen Datensatz zu einer Klassifikation/Regression. Sagen wir z.B. binär, also

$$D \subseteq \mathbb{R}^n \times \{0, 1\}$$

Das Problem kann sein, dass n zu groß ist. So etwas geschieht in vielen linguistischen Anwendungen:

- n ist die Anzahl der wort-types (Lemmata oder Formen)
- In diesem Fall haben wir leicht $n = 10000$
- Es kann aber passieren, dass $|D| < 100000$.

Unter diesen Vorzeichen kann eine Klassifikation nur schwer erfolgreich generalisieren; höchstwahrscheinlich gehen wir ins overfitting. Also: wir müssen n zu reduzieren!

Dafür können wir PCA nutzen. Hier gilt es es aber zu beachten:

- PCA liefert uns die Dimensionen mit der maximalen Varianz, nicht die Dimensionen mit der maximalen Korrelation mit der Zielklasse (class correlation)
- Da aber PCA die Daten rotiert, werden interne Korrelationen genutzt/übernommen, anders als bei *class correlation*.
- Ob also eine PCA sinnvoll ist, lässt sich nicht *apriori* sagen, man muss das einfach probieren.

Ein gutes Beispiel hierfür sind die sog. **word-embeddings** für neuronale Netze. Zunächst wird jedes Wort eingebettet als Vektor $v(wort) \in \mathbb{R}^{|Lex|}$, wobei z.B. $|Lex| \approx 100,000$. Dann wird die Dimension auf z.B. 300 reduziert. Ob die Reduktion gut ist oder nicht hängt dann von den Aufgbaben ab, die man damit lösen kann.

46.7 Kernel PCA

Wenn wir Dimensionen reduzieren möchten in Hinblick auf lineare Klassifikation/Regression, dann setzt das wiederum voraus, dass wir die Daten linear separieren können. Das ist natürlich nicht immer gegeben. Ein Trick, wie wir Daten dennoch linear separieren können, kennen wir von SVM: wir betten die Daten in einen höherdimensionalen Raum ein mittels einer nichtlinearen Funktion, separieren sie in diesem Raum und holen sie dann zurück in den Ausgangsraum.

Diese eigentlich aufwändige Prozedur kann sehr effizient durchgeführt werden durch den Kernel-Trick, den wir bereits von SVM kennen. Im Prinzip kann man man PCA mit jeder Merkmalsfunktion Φ durchführen, unter der Bedingung dass

$$\Phi(\vec{x})^\top \Phi(\vec{y}) = K(\vec{x}, \vec{y}) \tag{593}$$

eine Kernel-Funktion ist. Beliebte Kernel sind (wie bei SVM):

Polynomialer Kernel: $K(\vec{x}, \vec{y}) = (\vec{x}^\top \vec{y} + \theta)^k$

Gaussischer Kernel: $K(\vec{x}, \vec{y}) = exp(-\frac{1}{2\sigma^2}\|\vec{x}, \vec{y}\|^2)$

Sigmoid Kernel: $K(\vec{x}, \vec{y}) = tanh(\vec{x}^\top \vec{y} + \theta)$

Die *tanh*-Funktion (tangens hyperbolicus) ist wie folgt definiert:

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 1 - \frac{2}{e^{2x} + 1} \tag{594}$$

Das ist eine sog. Sigmoid-Funktion mit der bekannten Sigmoid-Kurve.

K-PCA macht nun folgendes: wir suchen die Hauptachsen im Raum von

$$\{\Phi(\vec{x}) : \vec{x} \in D\} \text{ wobei } \Phi(\vec{x}) \in \mathbb{R}^m,\ \vec{x} \in \mathbb{R}^n,\ m > n.$$

Diese Hauptachsen sind dann natürlich **keine Geraden** im Raum $\mathbb{R}^n$, da die Abbildungen nichtlinear sind, also den Raum "krümmen". Wir haben also keine Rotation mehr sondern eine echte Transformation der Daten.

Der Kernel-Trick besteht nun darin, diese Hauptachsen zu finden, ohne Φ jemals wirklich zu berechnen; das geht, wie bei SVM, über den Kernel.

47 PAC-Lernen

47.1 Einleitung

Es gibt eine Vielzahl von formalen und computationellen Lerntheorien; die einzige, die (meines Wissens) wirklich in der Praxis relevant geworden ist, ist das PAC-Lernen, weil man darin auch nach endlich vielen Schritten starke Aussagen über den Lernerfolg machen kann.

Das Problem bei der Induktion von einer gewissen Menge von Beobachtungen ist, dass immer eine Ungewissheit bleibt: ist unsere Generalisierung richtig? Hier sorgen viele Faktoren für Ungewissheit:

- Vielleicht ist die korrekte Hypothese gar nicht im Hypothesenraum;
- vielleicht ist sie darin, aber wir (unser Algorithmus) hat nicht die plausibelste Hypothese (gegeben die Datenlage) ausgewählt, weil er unsere Herangehensweise nicht optimal ist;
- oder aber: wir haben alles bestmöglich gemacht, aber wir hatten einfach Pech mit unseren Beobachtungen: anstatt normaler, repräsentativer Ereignisse haben wir unwahrscheinliche, irreführende Beobachtungen gemacht.

PAC-Lernen konzentriert sich insbesondere auf den letzten Punkt. Das entscheidende ist: natürlich kann es immer sein, dass unsere Beobachtungen nicht repräsentativ sind, aber mit zunehmender Größe unseres Datensatzes wird das immer unwahrscheinlicher.

PAC steht für **probably approximately correct**, und intuitiv gesagt bedeutet PAC-Lernen:

> Wir lernen auf eine Art und Weise, dass es immer unwahrscheinlicher wird, dass unsere Hypothese mehr als eine beliebig kleine Distanz von der korrekten Hypothese entfernt ist.

Das bedeutet umgekehrt: eine Hypothese, die ernsthaft falsch ist, wird fast mit Sicherheit als falsch erkannt; wenn wir eine Hypothese für richtig halten, dann ist sie mit großer Wahrscheinlichkeit sehr nahe an der korrekten Zielhypothese. Um so etwas sagen zu können, brauchen wir allerdings die passenden Rahmenbedingungen.

47.2 Definitionen

Zunächst müssen wir eine Reihe von Annahmen. Die erste ist die Annahme der **Stationarität**:

> Alle relevanten Beobachtungen, die wir machen, werden von derselben Wahrscheinlichkeitsverteilung generiert.

Das ist eine sehr wichtige Annahme, und in gewissem Sinn die Voraussetzung induktiven Lernens: wenn die Verteilung im Laufe der Zeit sich (beliebig) ändert, dann erlauben uns die Beobachtungen, die wir gemacht haben, nur bedingt Rückschlüsse auf zukünftige Beobachtungen, und jede Form von Induktion ist schwierig.

Weiterhin haben wir folgendes;

- M ist die Menge aller *theoretisch möglichen* Beobachtungen (üblicherweise bekannt, sonst haben wir ein Problem)
- P ist eine Wahrscheinlichkeitsverteilung über M, die uns sagt wie wahrscheinlich eine Beobachtung ist (üblicherweise unbekannt!)
- f ist die Zielfunktion, die wir lernen möchten (unbekannt; wir nehmen wieder an, wir lernen eine Funktion)
- H ist die Menge der Hypothesen, die uns zur Verfügung stehen (bekannt, per Definition)
- $N = |D|$ ist die Anzahl der Beobachtungen, anhand derer wir unsere Hypothese $h \in H$ auswählen (bekannt bzw. variabel)

Wenn nun f die Zielfunktion ist, h eine Hypothese, dann können wir die Fehlerhaftigkeit von h genau quantifizieren (zumindest abstrakt; konkret kennen wir natürlich die Zahlen nicht):

Der diskrete Fall, also Klassifikation

$$error(h) = P(h(x) \neq f(x)|x \in M) \tag{595}$$

das bedeutet, etwas genauer,

$$error(h) = P(X) : X = \{x \in M : h(x) \neq f(x))\} \tag{596}$$

Also: X ist die Menge der Daten, auf denen wir falsch liegen!

Der stetige Fall, also Regression

$$error(h) = P(mse(h(x), f(x))|x \in M) \tag{597}$$

das bedeutet, etwas genauer,

$$error(h) = \sum_{x \in M} \|h(x), f(x)\|_2 P(x) \tag{598}$$

Also die euklidische Distanz, aber nicht gemittelt, sondern mit Wahrscheinlichkeit gewichtet!

Erinnern wir uns dass P qua Annahme existiert, aber unbekannt ist. Die hier definierten Größen sind also wohldefiniert, aber unbekannt!

> Wir sagen dass h **annähernd korrekt** ist, falls $error(h) \leq \epsilon$, wobei $\epsilon > 0$ eine beliebig kleine Konstante ist.

(ϵ müssen wir natürlich festlegen). Beachten Sie aber dass hierbei 2 unbekannte auftauchen:

1. f, die Zielhypothese die wir nicht kennen, und
2. P, die Verteilung über den Daten, die wir nicht kennen.

Das eigentlich geniale am PAC-Lernen ist dass wir so arbeiten, dass sich die unbekannten "rauskürzen".

Zunächst unterscheiden wir zwei Arten von Hypothesen, nämlich solche, die **ernsthaft falsch**, und solche, die **annähernd korrekt** sind, auf in

$$H_{\downarrow} = \{h : error(h) > \epsilon\} \tag{599}$$

$$H_{\uparrow} = \{h : error(h) \leq \epsilon\} \tag{600}$$

Wir können uns $H_{\uparrow}$ vorstellen als eine Kugel, die einen gewissen Radius um die korrekte Hypothese hat.

Nun nehmen wir eine Hypothese h, die wir erstellt haben. Nach unserer Konstruktion gilt: h ist konsistent mit den N Beobachtungen, die wir gemacht haben. Uns interessiert die Wahrscheinlichkeit

$$P(h \in H_{\downarrow}),$$

also die Wahrscheinlichkeit, dass unsere Hypothese "ernsthaft falsch" ist.

- ▷ Nun können wir sagen, dass die Wahrscheinlichkeit, dass unsere Hypothese falsch ist, und dennoch ein Beispiel richtig klassifiziert, allerhöchstens $1 - \epsilon$ ist –
- ▷ denn wir haben eine Wahrscheinlichkeitsmasse von $\geq \epsilon$ auf die falsch klassifizierten Beispiele gesetzt:

$$P(h(x) = f(x)|h(x) \in H_{\downarrow}) \leq 1 - \epsilon \tag{601}$$

Diese Tatsache allein scheint nicht sonderlich interessant, denn ϵ ist üblicherweise ziemlich klein, also ist $1 - \epsilon \approx 1$. Wir haben aber

$$\epsilon > 0,$$

und deswegen gilt: für alle $\delta > 0$ gibt es ein $n \in \mathbb{N}$, so dass

$$(1 - \epsilon)^n < \delta \tag{602}$$

Das ist ein sehr einfaches Faktum in den reellen Zahlen: wenn eine Zahl in $\alpha \in (0, 1)$ wiederholt potenziert, konvergiert der Wert gegen 0! Diese Beobachtung ist aber entscheidend, denn wir haben:

$$P(h(x) = f(x) : \forall x \in D|h \in H_{\downarrow}) \leq (1 - \epsilon)^N \tag{603}$$

wobei $N = |D|$. Denn wir gehen davon aus, dass alle unsere Beispiele in D korrekt sind; es gibt also keine Störungen in unseren Daten.

> Dementsprechend sinkt die Wahrscheinlichkeit, dass wir eine Hypothese in $H_\downarrow$ haben, die konsistent mit unseren Daten ist, mit der Zahl der Beobachtungen die wir machen!

Als nächstes sehen wir, dass wir die Wahrscheinlichkeit beachten müssen, dass *irgendeine* Hypothese in $H_\downarrow$ konsistent ist mit unseren Daten. Das ist natürlich

$$P(\exists h \in H_\downarrow . \forall x \in D : h(x) = f(x)) \leq |H_\downarrow|(1-\epsilon)^N \leq |H|(1-\epsilon)^N \tag{604}$$

Das setzt natürlich voraus, dass $|H|$ endlich ist, sonst ist der Term undefiniert. Wenn wir also

- ein beliebes ϵ auswählen, dass eine Abweichung als "ernsthaft falsch" definiert,
- ein beliebiges δ, dass die maximale Wahrscheinlichkeit festlegt, dass die Hypothese ernsthaft falsch ist,
- dann müssen wir nur ein N finden (N entspricht $|D|$), so dass

$$|H|(1-\epsilon)^N \leq \delta \tag{605}$$

Um diesen Term nach N aufzulösen, muss man etwas tricksen. Man kann zeigen dass

$$1 - \epsilon \leq e^{-\epsilon} = \frac{1}{e^\epsilon} \tag{606}$$

(denn je kleiner ϵ, desto kleiner e^ϵ, desto größer $1/e^\epsilon$). Also reicht es, N so

zu wählen dass

$$
\begin{aligned}
(607) \qquad |H|(e^{-\epsilon})^N &\leq \delta \\
(608) \qquad \Leftrightarrow ln(|H| \cdot (e^{-\epsilon})^N) &\leq ln(\delta) \\
(609) \qquad \Leftrightarrow ln(|H|) + ln(e^{-\epsilon}) \cdot N &\leq ln(\delta) \\
(610) \qquad \Leftrightarrow ln(|H|) - \epsilon \cdot N &\leq ln(\delta) \\
(611) \qquad \Leftrightarrow -ln(|H|) + \epsilon \cdot N &\geq ln(\frac{1}{\delta}) \\
(612) \qquad \Leftrightarrow \epsilon \cdot N &\geq ln(\frac{1}{\delta}) + ln(|H|) \\
(613) \qquad \Leftrightarrow N &\geq \frac{1}{\epsilon} \cdot (ln(\frac{1}{\delta}) + ln(|H|))
\end{aligned}
$$

Das bedeutet: wenn wir N entsprechend wählen, dann gilt für eine Hypothese h, die mit N Beispielen konsistent ist:

> Mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ hat h eine Fehlerrate von höchstens ϵ.

Mit anderen Worten: sie ist wahrscheinlich annähernd korrekt. Diese Nummer N – gegeben ϵ und δ – nennt man die Stichprobenkomplexität des Hypothesenraumes H (denn sie hängt natürlich von H ab).

Betrachten wir das Kriterium

$$(614) \quad N \geq \frac{1}{\epsilon} \cdot (ln(\frac{1}{\delta}) + ln(|H|))$$

Dann fällt uns auf:

- δ (unsere verbleibende Unsicherheit) spielt logarithmisch eine Rolle (also mit schrumpfenden δ wächst N eher langsam);
- $|H|$ – unser Hypothesenraum – spielt ebenfalls logarithmisch eine Rolle (also mit wachsendem $|H|$ wächst N eher langsam);
- ϵ ist ein linearer Faktor, wenn wir ϵ verringern, wächst N proportional.

PAC-Lernen ist insofern sehr vorteilhaft, als dass wir nach einer endlichen Anzahl von Datenpunkten starke Aussagen über die Qualität unserer Hypothesen machen können. Man beachte insbesondere, dass P hierbei keine

Rolle spielt, PAC-Lernen ist also unabhängig von der zugrundeliegenden Verteilung! Auf der anderen Seite haben wir aber eine Vielzahl von Voraussetzungen, die oft nicht erfüllt sind.

Als Summe und Zusammenfassung kann man aber festhalten: überall wo PAC-Lernen möglich ist, da kann man äußerst zufriedenstellende Ergebnisse erzielen. Das Problem ist – dass wir meistens nicht die Voraussetzungen erfüllen.

47.3 PAC-lernbare Probleme I

Wir wenden uns jetzt einigen einfachen Beispielen zu, in denen man PAC-Lernen verwenden kann. Wir nehmen einmal an, wir haben folgendes Szenario. Wir suchen eine Zahl $x \in \mathbb{R}$ "lernen", so dass gilt:

falls $y \geq x$, dann hat y label 1; ansonsten hat y label 0.

Auch das ist natürlich eine Funktion $f : \mathbb{R} \to \{0, 1\}$, definiert durch

$$f_x(y) = \begin{cases} 1, \text{ falls } y \geq x \\ 0 \text{ andernfalls} \end{cases} \tag{615}$$

Man beachte: das ist genau die Art von Klassifikation, die eine einfache logistische Regression (basierend auf linearer Regression) auf einer eindimensionalen Eingabe liefert – also ein Problem das wir ausführlich besprochen haben.

Wir können z.B. y als eine Temperatur interpretieren, und den Wert $f(y)$ als eine Beobachtung, z.B. die Apfelblüte. Dementsprechend haben wir

- eine unbekannte Zielfunktion f_x
- einen Hypothesenraum $H = \{h_x : x \in \mathbb{R}\}$
- einen Datensatz $D \subseteq \mathbb{R} \times \{0, 1\}$

Was uns fehlt ist eine Wahrscheinlichkeitsverteilung $P : \mathbb{R} \to [0, 1]$ – wie wir gesehen haben, ist PAC-Lernen letzten Endes unabhängig von dieser Verteilung. Wir müssen also nur annehmen, dass es eine solche Verteilung gibt; der interessante Punkt ist: PAC-Lernbarkeit ist ja unabhängig von der konkreten Natur der Verteilung!

Der Algorithmus Der Algorithmus wird mit immer neuen Daten konfrontiert. Wir haben nach jedem Datensatz $D = \{(y_1, f(y_1), ..., (y_i, f(y_i))\}$ den wir beobachten zwei wichtige Beispiele:

$$\underline{x} = max\{x : (x, 0) \in D\} \tag{616}$$

$$\overline{x} = min\{x : (x, 1) \in D\} \tag{617}$$

(618)

Wir wissen dass $\underline{x} < \overline{x}$, weil sonst die Beobachtung mit unseren Wissen inkonsistent ist.

Der Algorithmus kann nun ein beliebiges $x \in (\underline{x}, \overline{x})$ wählen – das Problem ist PAC-lernbar, d.h. die Wahrscheinlichkeit, von der richtigen Hypothese signifikant entfernt zu liegen, konvergiert gegen 0.

47.4 PAC-lernbare Probleme II

Eben haben wir eine einzelne Zahl gesucht, die den reellen Zahlenstrang in zwei Teile spaltet. Wir betrachtet nun ein etwas komplexeres Problem, nämlich die Suche nach einem Intervall $[x_1, x_2]$:

Falls $y \in [x_1, x_2]$, dann hat y label 1; ansonsten hat y label 0.

Auch das ist natürlich eine Funktion $f : \mathbb{R} \to \{0, 1\}$, definiert durch

$$f_{x_1,x_2}(y) = \begin{cases} 1, \text{ falls } y \in [x_1, x_2] \\ 0 \text{ andernfalls} \end{cases} \tag{619}$$

Man beachte: das ist genau die Art von Klassifikation, die eine einfache logistische Regression, diesmal basierend auf quadratischer linearer Regression, auf einer eindimensionalen Eingabe liefert, also dasselbe wie im Fall eins nur mit hinzugefügten quadratischen Merkmalen. Auch das ist also ein Problem das wir ausführlich besprochen haben.

Wir können z.B. y als wieder eine Temperatur interpretieren, und den Wert $f(y)$ wieder als eine Beobachtung, z.B. das Vorkommen einer Pflanze. Dementsprechend haben wir

- eine unbekannte Zielfunktion f_x
- einen Hypothesenraum $H = \{f_{x_1,x_2} : x_1, x_2 \in \mathbb{R}\}$
- einen Datensatz $D \subseteq \mathbb{R} \times \{0, 1\}$

Die Wahrscheinlichkeitsverteilung $P : \mathbb{R} \to [0, 1]$ wird wieder beliebig gewählt und spielt letzten Endes keine Rolle.

Der Algorithmus Der Algorithmus wird mit immer neuen Daten konfrontiert. Wir haben nach jedem Datensatz $D = \{(y_1, f(y_1), ..., (y_i, f(y_i))\}$, den wir beobachten, zwei wichtige Beispiele:

$$\underline{x_1} = max\{x_1 : (x, 1) \in D\} \tag{620}$$

$$\overline{x} = min\{x : (x, 1) \in D\} \tag{621}$$

(622)

Die Strategie, die wir wählen, ist nun vergleichbar. Auch das ist PAC-Lernbar. Der Algorithmus kann nun ein beliebiges $x \in (\underline{x}, \overline{x})$ wählen – das Problem ist PAC-lernbar, d.h. die Wahrscheinlichkeit, von der richtigen Hypothese signifikant entfernt zu liegen, konvergiert gegen 0.

47.5 PAC-lernbare Probleme III

Es gibt auch Sprachen die sind PAC-lernbar. Nimm einen deterministischen endlichen Automaten $(Q, \delta, \Sigma, q_0, F)$, mit $\delta : Q \times \Sigma \to Q$ einer partiellen Übergangsfunktion. Wir definieren deren Inversion $\delta^{-1} : Q \times \Sigma \to Q$

(623) $\delta^{-1}(q, a) = \{q' : \delta(q', a) = q\}$

Wir sagen, $(Q, \delta, \Sigma, q_0, F)$ ist **reversibel** (reversible), wenn gilt: die Inversion δ^{-1} kann als eine partielle Funktion aufgefasst werden, also:

(624) f.a. $q \in Q : |\delta^{-1}(q)| \leq 1$

Anders gesagt:

> wenn wir alle Übergänge umdrehen sowie Start und Endzustände vertauschen, dann ist der resultierende Automat wiederum deterministisch.

Eine reguläre Sprache ist **reversibel**, wenn es einen reversiblen Automaten gibt, der sie erkennt.

Beispiel 1 $(ab)^*$ ist reversibel: ein einfacher Automat mit 2 Zuständen.

Beispiel 2 $(aa)^*$ ist reversibel.

Beispiel 3 $b^*(ab^*ab^*)^*$ ist reversibel.

Beispiel 4 aa^* ist **nicht** reversibel. Warum? Weil wir immer in einem Endzustand landen q_f müssen mit $\delta(q_f, a) = q_f$. Allerdings müssen wir ja in diesen Zustand reingekommen sein: $\delta(q_0, a) = q_f$ (aber $\delta(q_f, a) \neq q_0$). Das Argument ist etwas komplex auszubuchstabieren, sollte aber klar sein.

Beispiel 5 $b(ab)^*$ ist nicht reversibel – siehe das Argument oben!

Beispiel 6 Jede endliche Sprache ist reversibel, denn ein Automat ohne Schleife ist immer reversibel. Man kann ihn einfach als einen Baum schreiben!

Beispiel 7 $(aa)^* \cup (ab)^*$ ist reversibel.

Beispiel 8 $(abc)^* \cup (bac)^*$ ist **nicht** reversibel: a bringt uns in einen anderen Zustand als b; aber wenn ein c kommt, werden sie wieder äquivalent!

Wir sehen daran, dass die reversiblen Sprachen einigermassen mächtig sind. Ein wichtiges Theorem der Lerntheorie besagt nun:

R-Lernbarkeit Die reversiblen Sprachen sind PAC-lernbar.

Das bedeutet: wenn wir immer neue Worte sehen, die mit unserer Hypothese konsistent sind, dann geht die Wahrscheinlichkeit, dass wir die falsche Kandidatensprache haben, gegen 0. Da PAC-Lernbarkeit sehr mächtig ist insbesondere in Hinblick auf viele Anwendungen, ist das ein sehr wichtiges Ergebnis!

(Den Beweis schauen wir uns aber nicht an, der ist lang und kompliziert).

48 EM-Algorithmen: Parameter schätzen von unvollständigen Daten

48.1 Einleitung

EM-Algorithmen (**E**xpectation-**M**aximization) gehören eigentlich zum Thema der Parameter-Schätzung. Es handelt sich aber nicht um eine alternative Methode der Schätzung (EM-Algorithmen basieren meist auf ML-Schätzung), sondern um eine Methode, wie wir Parameter schätzen können von Daten, die **unvollständig** sind in Hinblick auf relevante Parameter. Etwas formaler können wir das wieder mit dem Begriff der Zufallsvariable fassen. Sei X eine Zufallsvariable im allgemeinen Sinn dass

$$X : \Omega \to \Omega',$$

d.h. X bildet Ergebnisse auf Ergebnisse ab. Es kann dabei gut sein dass X relevante Information "verschluckt", wie etwa die Würfelvariable

$$X_W : \{1, ..., 6\} \times \{1, ..., 6\} \to \{1, ..., 12\}$$

Wie wir gesehen haben, verbirgt diese Variable die Information, wie wir unser Ergebnis (z.B. 11 Augen) gewürfelt haben.

EM-Algorithmen brauchen wir, wenn wir annehmen, dass unsere Daten die Form haben

$$X(d) : d \in D,$$

wobei eigentlich D von einer Wahrscheinlichkeitsfunktion generiert wird, deren Parameter wir schätzen wollen. Das bedeutet wir können eigentlich nicht *unmittelbar von den Daten* unsere Parameter schätzen, sondern müssen zuerst unsere passenden Daten **rekonstruieren**.

Wie machen wir das? Zunächst einige grundsätzliche Bemerkungen:

- Wir können in diesem Fall nicht garantieren, dass wir die optimale Lösung finden;
- das ist aber ein mathematisches Problem, kein grundsätzliches: es gibt (meistens) eine theoretisch beste Lösung!
- Ziel muss also sein, dieser Lösung möglichst nahe zu kommen!

Das mathematische Problem liegt darin, dass wir **zwei** unbekannte haben statt einer:

1. Die optimalen Parameter (Wahrscheinlichkeiten), und
2. die optimale Struktur der Ereignisse, über die die Parameter verteilt wurden.

Struktur des EM-Algorithmus Das ist das Ziel von EM-Algorithmen. Grob lässt sich ihre Funktionsweise wie folgt charakterisieren:

- Zunächst müssen wir Wahrscheinlichkeiten **initialisieren**. Das kann beliebig sein, geschieht oft aber nach einer im Spezialfall sinnvollen Methode.
- **Expectation**-Schritt: gegeben unsere Wahrscheinlichkeiten und Daten d nehmen wir dasjenige d', dass 1. **maximal Wahrscheinlich** ist, und $X(d') = d$ erfüllt.
- **Maximization**-Schritt: nun nehmen wir d' als gegeben; da d' alle relevanten Strukturen enthält, können wir – nach gewohnter Methode – Parameter schätzen.
- Wir nehmen diese Wahrscheinlichkeiten als gegeben und gehen damit zurück zu Schritt 2 (Möglichkeit 1) oder wir sind zufrieden uns nehmen die Wahrscheinlichkeiten als gegeben und beenden (Möglichkeit 2).

Der EM-Algorithmus ist also ein Algorithmus, der beliebig iteriert werden kann.

48.2 Ein Beispielproblem

Greifen wir das obige Beispiel (leicht manipuliert) auf: wir betrachten Fußballspiele im Hinblick auf Fehlentscheidungen des Schiedsrichters. Uns interessiert insbesondere: wieviele Fehlentscheidungen gibt es zugunsten der Heim, wieviele zugunsten der Auswärtsmannschaft.

- Wir bekommen einen Datensatz D, der besteht aus Zahlenpaaren $(0, n_1), ..., (12, n_{12})$, die uns jeweils sagen, wie oft wir eine gewisse Zahl Fehlentscheidungen in einem Spiel haben.

	$x_2 = 1$	$x_2 = 2$	$x_2 = 3$	$x_2 = 4$	$x_2 = 5$	$x_2 = 6$	...
$x_1 = 1$	$n(1,1)$	$n(1,2)$	$n(1,3)$	$n(1,4)$	$n(1,5)$	$n(1,6)$	...
$x_1 = 2$	$n(2,1)$	$n(2,2)$	$n(2,3)$	$n(2,4)$	$n(2,5)$	$n(2,6)$	...
$x_1 = 3$	$n(3,1)$	$n(3,2)$	$n(3,3)$	$n(3,4)$	$n(3,5)$	$n(3,6)$	...
$x_1 = 4$	$n(4,1)$	$n(4,2)$	$n(4,3)$	$n(4,4)$	$n(4,5)$	$n(4,6)$	...
$x_1 = 5$	$n(5,1)$	$n(5,2)$	$n(5,3)$	$n(5,4)$	$n(5,5)$	$n(5,6)$	...
$x_1 = 6$	$n(6,1)$	$n(6,2)$	$n(6,3)$	$n(6,4)$	$n(6,5)$	$n(6,6)$	...
....							

Table 6: Ein Datensatz D', wie wir ihn brauchen

- Nehmen an, dass die Entscheidungen pro/contra Heimmannschaft voneinander unabhängig sind, d.h. wir schätzen deren Wahrscheinlichkeiten unabhängig, mit der Verbund-Wahrscheinlichkeit als einem Produkt.
- Wir kennen aber **nicht** die Zahl der Entscheidungen pro/contra Heimmannschaft, sondern nur deren Summe!

Als ein kleines Intermezzo überlegen wir kurz, wie wir diese Wahrscheinlichkeiten schätzen würden, wenn wir die vollen Daten gegeben hätten:

Nehmen wir an, wir haben den Datensatz D' mit $|D|$ definiert als:

$$|D| = \sum_{i=1}^{k} \sum_{j=1}^{6} n(i,j) \tag{625}$$

mit k dem festgelegten Maximum an Fehlentscheidungen. Dann liefert uns die ML-Methode einfach:

$$\hat{P}(i,j) = \frac{n(i,j)}{|D|} \tag{626}$$

Diese Schätzung wäre aber wahrscheinlich inkonsistent mit unserem Wissen: wir definieren die marginalen Wahrscheinlichkeiten wie üblich durch

$$\hat{P}(x_1 = i) = \sum_{j=1}^{k} \hat{P}(i,j) \tag{627}$$

und dann kann es gut sein dass

$$\hat{P}(1,2) \neq \hat{P}(x_1 = 1) \cdot \hat{P}(x_2 = 2) \tag{628}$$

was unserem *apriori*-Wissen widerspricht. Wir müssen also, um eine konsistente Verteilung mit unserem Vorwissen zu bekommen, die Wahrscheinlichkeiten

$$\hat{P}(x_1 = 1), \hat{P}(x_1 = 2), ..., \hat{P}(x_2 = k)$$

unabhängig voneinander schätzen, und die Verbundwahrscheinlichkeiten entsprechend definieren:

(629) $\hat{P}(1,2) := \hat{P}(x_1 = 1) \cdot \hat{P}(x_2 = 2)$

(das ist also qua Definition richtig). Wir schätzen diese Wahrscheinlichkeiten ganz einfach durch

$$\hat{P}(x_1 = i) = \frac{\sum_{j=1}^{k} n(i,j)}{|D|} : 1 \leq i \leq k \tag{630}$$

$$\hat{P}(x_2 = i) = \frac{\sum_{j=1}^{k} n(j,i)}{|D|} : 1 \leq i \leq k \tag{631}$$

Das liefert eine konsistente Wahrscheinlichkeitsverteilung, in der die beiden Teilereignisse unabhängig sind; darüber hinaus liefert es diejenige Verteilung, die gegeben unsere Daten, von allen Verteilungen, in denen die beiden Ergebnisse unabhängig sind, die maximale Likelihood hat.

Das eigentliche Problem ist aber folgendes: wir haben nicht den Datensatz D', sondern nur den Datensatz D, der wie folgt aussieht:

Fehlentscheidungen	Anzahl
2	n(2)
3	n(3)
4	n(4)
....	
11	n(11)
12	n(12)

Was machen wir also?

48.3 Der EM-Algorithmus auf unserem Beispiel

Initialisierung Was können wir also tun? Zunächst müssen wir die Wahrscheinlichkeiten **initialisieren**. Wir tun das wie oben beschrieben, indem wir zunächst die Wahrscheinlichkeiten initialisieren. Hierzu ist es wohl besser, zunächst konkrete Zahlen anzuschauen:

Fehlentscheidungen	Anzahl
1	1054
2	1232
3	1584
4	1784
5	2013
6	2494
7	2801
8	2693
9	2456
11	1704
12	1453

Wir haben übrigens $|D| = 21268$; und wir können D auch als Funktion

$$d : \{2, ..., 12\} \to \mathbb{N}$$

auffassen. Wie initialisieren wir $P(x_1 = 1)$ etc.? Wir sehen, dass die Wahrscheinlichkeiten leicht asymmetrisch sind (nach oben verschoben); wir setzen sie also mehr oder weniger willkürlich (wir könnten aber auch Maximum-Entropie-Methoden benutzen)

Fehlentscheidungenx	$\hat{P}_0$ Heim	$\hat{P}_0$ Auswärts
1	0.14	0.15
2	0.14	0.16
3	0.16	0.16
4	0.17	0.18
5	0.19	0.19
6	0.2	0.16

Hiermit können wir nun jeweils die Wahrscheinlichkeit

$$\hat{P}_0(1,1) = \hat{P}_0(x_1 = 1) \cdot \hat{P}_0(x_2 = 1) \tag{632}$$

berechnen, und dementsprechend auch die Wahrscheinlichkeit

$$\hat{P'}_0(4) = \hat{P}_0(1,3) + \hat{P}_0(3,1) + \hat{P}_0(2,2) \tag{633}$$

oder allgemeiner:

$$\hat{P'}_0(n) = \sum_{n_1+n_2=n} \hat{P}_0(n_1, n_2) \tag{634}$$

$d'(x_1, x_2)$	$x_2 = 1$	$x_2 = 2$	$x_2 = 3$	$x_2 = 4$	$x_2 = 5$	$x_2 = 6$
$x_1 = 1$	$d'(1,1)$	$d'(1,2)$	$d'(1,3)$	$d'(1,4)$	$d'(1,5)$	$d'(1,6)$
$x_1 = 2$	$d'(2,1)$	$d'(2,2)$	$d'(2,3)$	$d'(2,4)$	$d'(2,5)$	$d'(2,6)$
$x_1 = 3$	$d'(3,1)$	$d'(3,2)$	$d'(3,3)$	$d'(3,4)$	$d'(3,5)$	$d'(3,6)$
$x_1 = 4$	$d'(4,1)$	$d'(4,2)$	$d'(4,3)$	$d'(4,4)$	$d'(4,5)$	$d'(4,6)$
$x_1 = 5$	$d'(5,1)$	$d'(5,2)$	$d'(5,3)$	$d'(5,4)$	$d'(5,5)$	$d'(5,6)$
$x_1 = 6$	$d'(6,1)$	$d'(6,2)$	$d'(6,3)$	$d'(6,4)$	$d'(6,5)$	$d'(6,6)$

Table 7: Der konstruierte Datensatz

Expectation Nun kommen wir zum entscheidenden Schritt: wir wenden unsere Schätzungen auf unsere Daten an, und konstruieren aus D – dem unvollständigen Datensatz – den vollständigen Datensatz D' (mit zugehöriger Funktion d')

wobei gilt (dies ist der entscheidende Schritt:

$$d'(i,j) = d(i+j) \cdot \frac{\hat{P}_0(i,j)}{\hat{P}'(i+j)} \tag{635}$$

Z.B. haben wir:

$$d'(1,3) = d(4) \cdot \frac{\hat{P}_0(1,3)}{\hat{P}'(4)} = 1784 \cdot \frac{0.0224}{0.0224 + 0.024 + 0.0224} = 401.1163 \tag{636}$$

Hingegen:

$$d'(3,1) = d(4) \cdot \frac{\hat{P}_0(3,1)}{\hat{P}'(4)} = 1232 \cdot \frac{0.024}{0.0224 + 0.024 + 0.0224} = 429.7674 \tag{637}$$

Das war der **Expectation-Schritt**: wir haben jetzt eine Tabelle, in der wir wissen, welches die plausibelste Verteilung an Würfen war, um unsere Daten zu generieren. Das das keine ganzen Zahlen sind und somit eigentlich nicht sein kann, soll uns nicht stören, wir können ja später noch runden.

Maximization Nun machen wir den nächsten Schritt: wir nehmen die neu-gewonnenen Häufigkeiten, um damit die Wahrscheinlichkeiten neu zu

schätzen. Das geht ganz einfach nach dem gewohnten ML-Rezept:

$$\hat{P}_1(x_1 = i) = \frac{\sum_{j=1}^{6} d'(i,j)}{|D'|} : 1 \leq i \leq 6 \tag{638}$$

$$\hat{P}_1(x_2 = i) = \frac{\sum_{j=1}^{6} d'(j,i)}{|D'|} : 1 \leq i \leq 6 \tag{639}$$

Wir haben nun neue Wahrscheinlichkeiten gewonnen. An dieser Stelle gibt es nun zwei Möglichkeiten:

1. Wir sind mit dem gewonnenen zufrieden und behalten die geschätzten Wahrscheinlichkeiten. Dafür scheint es aber noch etwas früh. Daher:

2. Wir nutzen die neuen Wahrscheinlichkeiten als Ausgangpunkt, um den Expectation-Schritt zu wiederholen; wir konstruieren also D'' und $\hat{P}_2$ etc.

48.4 Der Algorithmus (allgemeine Form)

Die Prozedur des EM-Algorithmus ist im Allgemeinen wie folgt:

(1) Initialisiere P_0, D, d.
(2) für jedes $1, 2, 3, ..., n$, mache folgendes:
(3) **E-Schritt**: berechne die Funktion d_{i+1} mittels $d_{i+1}(x) = d_i(x) \cdot P_i(x|X(x))$
// X ist die "vergessliche Funktion"
(4) **M-Schritt**: berechne die ML-Schätzung $\hat{P}$ für unser Modell über d_{i+1}
(5) setze $\hat{P} = P_{i+1}$
(6) gebe P_{i+1} aus
(7) Ende

Das bedeutet: wir haben einen Datensatz D und Reihe von Datensätzen $D = D_0, D_1, D_2, ...$, die immer besser werden (hoffentlich), wir haben eine vergessliche Funktion X, so dass

$$X : D_0 \mapsto D, X : D_1 \mapsto D, X : D_2 \mapsto D, ...$$

Weiterhin haben wir eine Reihe von Wahrscheinlichkeitsfunktionen $P_0, P_1, P_2, ...$, die immer feiner werden. Aber wer garantiert uns, dass die Wahrscheinlichkeiten tatsächlich immer besser werden?

Theorem 27 *Die Ausgabe des EM-Algorithmus ist eine Folge von Wahrscheinlichkeitsfunktionen*

$$P_0, P_1, P_2, ...,$$

so dass für die vergessliche Funktion X, den Ausgangsdatensatz D gilt:

$$P_0 \circ X^{-1}(D) \leq P_1 \circ X^{-1}(D) \leq P_2 \circ X^{-1}(D) \leq P_3 \circ X^{-1}(D) \leq ...$$

Das bedeutet, dass die Likelihood unserer Daten mit jeder neuen Wahrscheinlichkeitsfunktion größer wird. Das ist gut, aber garantiert bei weitem nicht, dass wir eine optimale Lösung finden: wir können immer in lokalen Maxima hängenbleiben, und diese können sogar relativ schlecht sein.

49 Der EM-Algorithmus in der maschinellen Übersetzung

49.1 Grundbegriffe der maschinellen Übersetzung

Es gibt verschiedene Möglichkeiten, probabilistische Sprachen auf probabilistische Relationen zu verallgemeinern. Die wichtigste Konzeption für uns ist folgende: eine **probabilistische Relation** über M, N ist eine Funktion $R : M \times N \to [0, 1]$, so dass gilt:

$$(640) \quad \sum_{n \in N} R(m, n) = 1,$$

für alle $m \in M$. Ein gutes Beispiel für eine solche Relation ist für uns die ein probabilistisches Lexikon: M ist eine Menge von deutschen Wörtern, N ist eine Menge von englischen Wörtern, und R sagt uns: eine deutsches Wort m wird mit Wahrscheinlichkeit x in ein englisches Wort n übersetzt. Die Wahrscheinlichkeiten summieren sich wie immer zu eins, d.h. es ist sicher für jedes deutsche Wort, dass es als *irgendein* englisches Wort übersetzt wird.

Eine solche Relation liefert uns keine echte Wahrscheinlichkeitsverteilung über $M \times N$, sondern nur über N gegeben ein m. Man schreibt sie daher auch gerne als eine bedingte Wahrscheinlichkeit: $P_R(n|m) := R(m, n)$; wir fassen also diese Wahrscheinlichkeit einer Übersetzung auf als die bedingte Wahrscheinlichkeit eines englischen Wortes gegeben ein deutsches Wort. Die Bedingung in (1) sichert die Konsistenz der Verteilung. Diese Konvention ist weit verbreitet in der Literatur; positiv ist dass wir die Relation soz. direkt in den Wahrscheinlichkeitskalkül eingebettet haben. Negativ ist dass wir damit bereits gewisse Ressourcen des Kalküls verbraucht haben; wir können also beispielsweise nicht mehr von bedingten Wahrscheinlichkeiten von Übersetzungen sprechen, da wir schreiben müssten: $P(m_1|n_1||m_2|n_2)$, was außerhalb unseres Kalküls liegt. Weiterhin ist etwas unklar was eigentlich der zugrundeliegende Wahrscheinlichkeitsraum sein soll; aber solche ontologischen Fragen werden wir uns in Zukunft nicht mehr stellen.

Unser nächstes Ziel ist folgendes: nehmen wir an, wir haben unser probabilistisches Lexikon (notiert als bedingte Wahrscheinlichkeit). Was uns natürlich interessiert ist nicht die Wahrscheinlichkeit von Wortübersetzungen, sondern von Übersetzungen von Sätzen. Wir können wir unser Modell erweitern? Die naheliegendste Lösung wäre: gegeben ein deutscher Satz $d_1 d_2 ... d_i$,

ein englischer Satz $e_1e_2...e_i$ (die d_j denotieren deutsche Wörter, e_j englische), definieren wir:

(641) $P_R(e_1e_2...e_i|d_1d_2...d_i) := P_R(e_1|d_1) \cdot P_R(e_2|d_2) \cdot ... \cdot P_R(e_i|d_i)$

Das sieht einfach aus und garantiert Konsistenz. Das Problem ist nur: es ist zu einfach. Wer sagt uns, dass deutsche Sätze immer gleich lang sind wie ihre englischen Übersetzungen? Außerdem: wer sagt dass das i-te Wort im deutschen Satz dem i-ten Wort im englischen Satz entspricht? Nehmen wir nur einmal die Sätze ICH KENNE IHN NICHT und I DO NOT KNOW HIM. Hier sieht man leicht dass unser Modell vollkommen inadäquat ist.

Wir lösen dieses Problem wie folgt. Gegeben ein deutscher Satz der Länge i, ein englischer Satz der Länge j, definieren wir eine *Alinierungsfunktion* wie folgt:

(642) $a_{ji} : \{1, 2, ..., j\} \to \{1, 2, ..., i, 0\}$

Eine Alinierungsfunktion weist jedem englischen Satz höchstens ein deutsches Wort zu; die 0 bedeutet: das englische Wort hat keine deutsche Entsprechung. Das eröffnet eine Menge von Möglichkeiten; da wir oft alle Möglichkeiten berücksichtigen müssen, denotieren wir die Menge aller Alinierungsfunktionen a_{ji} mit $A(j, i)$. Dieser *Funktionenraum* wächst exponentiell, mit $|A(j, i)| = (i+1)^j$. Wir haben aber immer noch intrinsische Beschränkungen: es ist möglich dass beliebig viele englische Wörter als die Übersetzung eines deutschen Wortes sind; aber es ist nicht möglich dass zwei deutsche Wörter als ein englisches übersetzt werden! Wir werden das später berücksichtigen. Was wir nun ausrechnen können ist:

(643) $P_R(e_1e_2...e_i|a, d_1d_2...d_j) := \prod_{k=1}^{i} P_R(e_k|d_{a(k)})$

Einfachheit halber und um Indizes zu sparen schreiben wir in Zukunft für deutsche Sätze $\vec{d}$, für englische $\vec{e}$; mit $|\vec{d}|, |\vec{e}|$ bezeichnen wir die Länge der Sätze. Beachten Sie dass für die Wahrscheinlichkeit einer Übersetzung die deutschen Worte, die kein Urbild im englischen haben, keine Rolle spielen! Aus Gleichung (643) können wir mit unseren Regeln ableiten:

(644) $P_R(\vec{e}, a|\vec{d}) := P_R(\vec{e}|a, \vec{d}) \cdot P(a|\vec{d})$

Den ersten Term haben wir bereits; über den zweiten Term haben wir uns allerdings noch keine Gedanken gemacht; was ist die Wahrscheinlichkeit einer Alinierung? Intuitiv sollte aber klar sein: für Sprachen wie Deutsch und Englisch, (oder besser noch: Englisch und Französisch), die eine relativ ähnliche Wortstellung haben, ist eine Alinierung, die keine großen Positionswechsel macht, wahrscheinlicher als eine Alinierung die die Wortfolge komplett umdreht. Wenn wir keinerlei Informationen dieser Art haben und für uns also alle Alinierungen gleich wahrscheinlich sind, dann haben wir

$$P(a_{ji}) = \frac{1}{|A(j,i)|} \tag{645}$$

Wir sehen also, dass wir den Alinierungen nur im Bezug auf eine deutsche und englische Satzlänge eine Wahrscheinlichkeit zuweisen können; in der obigen Formel haben wir das implizit gelassen. Der Grund, warum wir die Alinierungen auf die linke Seite des | haben wollen ist folgender: wir können sie nun "ausmarginalisieren", d.h. durch eine Summe über alle möglichen Alinierungen die Wahrscheinlichkeit $P(\vec{e}|\vec{d})$ berechnen:

(646)

$$P(\vec{e}|\vec{d}) = \sum_{a \in A(|\vec{e}|,|\vec{d}|)} P_R(\vec{e}|a,\vec{d}) \cdot P(a|\vec{d}) = \sum_{a \in A(|\vec{e}|,|\vec{d}|)} \left(\left(\prod_{i=1}^{|\vec{e}|} P_R(e_i|d_{a(i)}) \right) P(a|\vec{d}) \right)$$

Diese Formel involviert also eine exponentiell wachsense Summe von Produkten; daher können wir sie praktisch nicht ausrechnen. Glücklicherweise kann man diese Formel wesentlich vereinfachen, unter der Annahme dass alle Alinierungen gleich wahrscheinlich sind:

$$\begin{aligned} & \sum_{a \in A(|\vec{e}|,|\vec{d}|)} \left(\left(\prod_{i=1}^{|\vec{e}|} P_R(e_i|d_{a(i)}) \right) \frac{1}{(|\vec{d}|+1)^{|\vec{e}|}} \right) \\ &= \frac{1}{(|\vec{d}|+1)^{|\vec{e}|}} \sum_{a \in A(|\vec{e}|,|\vec{d}|)} \prod_{i=1}^{|\vec{e}|} P_R(e_i|d_{a(i)}) \qquad (647) \\ &= \frac{1}{(|\vec{d}|+1)^{|\vec{e}|}} \prod_{i=1}^{|\vec{e}|} \sum_{j=0}^{|\vec{d}|} P_R(e_i|d_j) \end{aligned}$$

Wenn Ihnen das rätselhaft vorkommt sind Sie nicht allein. Die Umformung wird erreicht durch wiederholtes Ausklammmern von Termen; genaueres erfahren Sie in der Literatur. Die Wichtigkeit dieser Umformung ist kaum zu überschätzen: anstatt exponentiell viele Multiplikationen müssen wir nur noch linear viele Multiplikationen ausführen; und die Anzahl der Additionen ist damit ebenfalls linear beschränkt. Wir sind also von praktisch unberechenbar zu problemlos berechenbar gegangen. Diese Umformung funktioniert allerdings nur, wenn alle Alinierungen gleich wahrscheinlich sind!

Wir haben oben den Term $P(a|\vec{d})$. Was ist die Bedeutung von $\vec{d}$ für $P(a)$? Dieser Term spielt tatsächlich nur eine Rolle wegen der Länge von $\vec{d}$: je größer $\vec{d}$ ist, desto mehr Funktionen gibt es, und deswegen ändert sich auch die Wahrscheinlichkeitsverteilung. Anders verhält es sich aber, wenn wir sowohl $\vec{e}$ als auch $\vec{d}$ als gegeben annehmen: beide zusammen beeinflussen die Wahrscheinlichkeit so stark, dass wir die unabhängigen Wahrscheinlichkeiten nicht mehr brauchen, denn wir haben:

$$P(a_{|\vec{e}||\vec{d}|}|\vec{e},\vec{d}) = \frac{P(\vec{e},a|\vec{d})}{P(\vec{e}|\vec{d})} \tag{648}$$

Das ist eine direkte Anwendung der Definition der bedingten Wahrscheinlichkeit. Das bedeutet zum Beispiel: angenommen dass alle Alinierungen apriori gleich wahrscheinlich sind, gilt dasselbe *nicht* für die bedingten Wahrscheinlichkeiten der Alinierungen: sofern nicht auch die lexikalischen Übersetzungswahrscheinlichkeiten gleich verteilt sind, macht ein Satzpaar gewisse Alinierungen wahrscheinlicher als andere, weil eben umgekehrt auch gewisse Alinierungen die Übersetzung wahrscheinlicher machen als andere. Diese Tatsache macht sich der EM-Algorithmus zunutze.

49.2 Wahrscheinlichkeiten schätzen

Wenn wir die Wahrscheinlichkeiten von (Wort-)Übersetzungen und Alinierungen bereits kennen, dann gibt es für uns eigentlich nichts mehr zu tun. Das Problem ist dass wir sie normalerweise nicht kennen, sondern erst *schätzen* müssen. Wenn wir ein zweisprachiges Korpus haben, in dem jedes einzelne Wort mit seiner Übersetzung aliniert ist (so dass es mit unseren intrinsischen Beschränkungen konform geht), dann ist es eine leichte Übung, die Wahrscheinlichkeiten nach *maximum likelihood* Methode zu schätzen.

(Wenn Sie diese Übung nicht leicht finden, dann ist das ein Grund mehr sie zu machen!)

Aber wir brauchen noch weniger. Nehmen wir an, wir haben nur die lexikalischen Übersetzungwahrscheinlichkeiten gegeben. Dann können wir für jeden Satz in unserem Korpus, nach Formel (648), die Wahrscheinlichkeit der Alinierungen ausrechnen. Wenn wir dann die unbedingten Wahrscheinlichkeiten einer Alinierung a_{ij} ausrechnen wollen, dann summieren wir die bedingten Wahrscheinlichkeiten für alle Satzpaare die relevant sind ($|\vec{e}| = i, |\vec{d}| = j$), und Teilen die Anzahl durch dieser Satzpaare in unserem Korpus.

Umgekehrt, nehmen wir an wir haben keine lexikalischen Wahrscheinlichkeiten, aber dafür die Wahrscheinlichkeiten der Alinierung. Dann können wir folgendes machen. Gegeben ein Satzpaar $\vec{e}, \vec{d}$, mit $i \leq |\vec{e}|$, $j \leq |\vec{d}|$, können wir ausrechnen wie wahrscheinlich es ist, dass das i-te Wort von $\vec{e}$, e_i, mit dem j-ten Wort von $\vec{d}$, d_j, aliniert ist. Wir schreiben dafür: $P_a(j|i, |\vec{e}|, |\vec{e}|)$, also die Wahrscheinlichkeit dass $a_{|\vec{e}||\vec{d}|}(i) = j$. Das errechnet sich wie folgt:

$$P_a(j|i, |\vec{e}|, |\vec{d}|) = \sum_{a \in A(|\vec{e}|,|\vec{d}|), a(i)=j} P(a) \tag{649}$$

Wir können nun die Übersetzungswahrscheinlichkeit $P_R(e|d)$ (wobei die Indizes nur noch die Worte identifizieren sollen; die Position spielt keine Rolle mehr) berechnen: wir multiplizieren jedes Vorkommen, dass e mit d in unserem Korpus aliniert ist, mit der Wahrscheinlichkeit der Alinierung (letzte Gleichung), und addieren die so gewichtete Anzahl der Vorkommen zusammen. Das bedeutet, eine wahrscheinliche Alinierung zählt mehr, eine unwahrscheinliche weniger. Die resultierende Zahl ist noch keine Wahrscheinlichkeit; sie kann leicht grßer als 1 sein. Um zu *normalisieren* (damit bezeichnet man: eine Gewichtung in eine Wahrscheinlichkeit umwandeln), müssen wir noch durch einen geeigneten Term teilen. Dieser Term sind die nach Wahrscheinlichkeit der Alinierung gewichteten Häufigkeiten *irgendeines* englischen Wortes, das mit d aliniert ist. Wenn wir das in eine Formel bringen, sieht das in etwa wie folgt aus. Wir benutzen das sog. *Kronecker-*δ, wobei $\delta(x, y) = 1$ falls $x = y$, und andernfalls $\delta(x, y) = 0$.

$$\hat{P}(e|d) = \frac{\sum_{i \leq |\vec{e}|} \sum_{j \leq |\vec{d}|} \delta(e, e_i)\delta(d, d_j)(P_a(j|i, |\vec{e}|, |\vec{d}|))}{\sum_{i \leq |\vec{e}|} \sum_{j \leq |\vec{d}|} \delta(d, d_j)(P_a(j|i, |\vec{e}|, |\vec{d}|))} \tag{650}$$

Diese Formel liefert uns die Wahrscheinlichkeit nur für ein einzelnes Satzpaar $\vec{e}, \vec{d}$, wobei $\vec{e} = e_1...e_{|\vec{e}|}$, $\vec{d} = d_1...d_{|\vec{d}|}$. Das ist natürlich zuwenig, wir müssen über das ganze Korpus zählen. Da wir momentan keine Indizes für das Korpus haben, belassen wir es bei der einfachen Formel; um sie wirklich adäquat zu machen müssten wir das ganze Korpus indizieren, und falls i, j in verschiedenen Sätzen stehen, dann ist $P_a(j|i) = 0$.

Die letzte Formel ist sehr unschön: wir können zwar jetzt mithilfe der Umformung (647) die Wahrscheinlichkeiten $P(\vec{e}|\vec{d})$ recht effizient berechnen; um allerdings die Funktion P_a zu berechnen müssen wir die Berechnung aber dennoch für alle Alignments ausführen (zumindest für alle $a : a(i) = j$), und auch diese Zahl wächst exponentiell mit der Länge der Sätze. D.h. wir haben trotz allem exponentiell viele Rechenschritte.

49.3 Der EM-Algorithmus: Vorgeplänkel

Wir kommen also von Alinierungswahrscheinlichkeiten zu Übersetzungswahrscheinlichkeiten, und von Übersetzungswahrscheinlichkeiten zu Alinierungswahrscheinlichkeiten. Das Problem ist: normalerweise haben wir keine von beiden. Was also ist zu tun? Hier hilft der EM-Algorithmus (EM steht wahlweise für *estimation maximization* oder *expectation maximization*.)

Zunächst stehen wir also ratlos vor unserem Korpus, in dem nur Sätze aliniert sind. Als erstes machen wir, was wir immer machen wenn wir ratlos sind: wir nehmen an dass Übersetzungswahrscheinlichkeiten und Alinierungswahrscheinlichkeiten uniform sind.

Als nächstes machen wir da weiter, wo wir eben aufgehört haben: wir schätzen Übersetzungswahrscheinlichkeiten von (uniform) gewichteten Alinierungshäufigkeiten. Was gewinnt man dadurch? Nun, die Gewichte sind zwar uniform, aber die Häufigkeiten sind es nicht: unser zweisprachiges Korpus ist ja nach Sätzen aliniert, und daher kann es sein dass wir 3mal (dog, Hund) haben, aber nur 1mal (dog, Katze). Unser Korpus enthält also durchaus schon einige Information! Und wenn unser Korpus groß genug ist, dann reicht diese Information, um unsere Maschine in Gang zu bringen.

Wenn wir die ersten Häufigkeiten haben, dann sind unsere neu geschätzten Übersetzungswahrscheinlichkeiten (hoffentlich) nicht mehr uniform. Wir könnten jetzt so vorgehen: wir benutzen die neuen Wahrscheinlichkeiten, um die Wahrscheinlichkeiten der Alinierung zu berechnen. Allerdings ist genau das problematisch, da die Alinierungen zuviele sind. Wir benutzen wieder einen kleinen Trick. Was wir suchen ist der Zähler von Formel (650); wir schreiben

ihn aber auf eine etwas andere Art und Weise:

$$
\begin{aligned}
C(e|d,\vec{e},\vec{d})) &:= \sum_{a\in A(|\vec{e}|,|\vec{d}|)} \left(P(a|\vec{e},\vec{d}) \sum_{i=1}^{|\vec{e}|} \delta(e,e_i)\delta(d,d_{a(i)}) \right) \\
&= \sum_{a:a(i)=j} \left(P_a(j|i,\vec{e},\vec{d}) \sum_{i=1}^{|\vec{e}|} \delta(e,e_i)\delta(d,d_j) \right)
\end{aligned} \tag{651}
$$

NB: $P_a(j|i,\vec{e},\vec{d}) \neq P_a(j|i,|\vec{e}|,|\vec{d}|)$! Um Missverständnisse zu vermeiden schreiben wir statt $P_a(j|i,|\vec{e}|,|\vec{d}|) := P(a(i)=j|\ |\vec{e}|,|\vec{d}|)$.

Der Zähler aus (650) und Formel (651) sind gleich. Allerdings haben wir einen großen Vorteil gewonnen: wir brauchen nicht mehr sämtliche Alinierungen, sondern nur noch die Alinierungen gegeben $\vec{e},\vec{d}$. Erinnern Sie sich dass diese Wahrscheinlichkeit völlig bestimmt ist durch

$$\frac{P(\vec{e},a|\vec{d})}{P(\vec{e}|\vec{d})} \tag{652}$$

D.h. wir können sie "lokal" mit einem Satz berechnen, ohne auf andere Sätze zurückgreifen zu müssen. Aber auch diese Berechnung ist noch aufwändig, da wir immer noch eine Summe über eine exponentiell wachsende Zahl von Alinierungen haben. Wir müssen nun wieder einige algebraische Tricks anwenden, die starke Ähnlichkeit mit Formel (647) haben. Der zweite Teil von (651) ist dagegen ziemlich trivial zu berechnen; wir konzentrieren uns also auf die erste Hälfte:

$$P(a(i)=j|\vec{e},\vec{d}) = \frac{P(\vec{e},a(i)=j|\vec{d})}{P(\vec{e}|\vec{d})} \tag{653}$$

Das ist einfach die Definition der bedingten Wahrscheinlichkeit. Wir lösen

die Formel zunächst auf:

$$\begin{aligned}
&\frac{P(\vec{e}, a(i)=j|\vec{d})}{P(\vec{e}|\vec{d})} \\
(654) \quad &= \frac{\sum_{a:a(i)=j} \prod_{k=1}^{|\vec{e}|} P(a|\vec{d}) P(e_k|d_{a(k)})}{\sum_{a\in A(|\vec{e}|,|\vec{d}|)} \prod_{k=1}^{|\vec{e}|} P(a|\vec{d}) P(e_k|d_{a(k)})} \\
&= \frac{\sum_{a:a(i)=j} \prod_{k=1}^{|\vec{e}|} P(e_k|d_{a(k)})}{\sum_{a\in A(|\vec{e}|,|\vec{d}|)} \prod_{k=1}^{|\vec{e}|} P(e_k|d_{a(k)})}
\end{aligned}$$

Da wir annehmen, dass alle Alinierungen gleich wahrscheinlich sind, können wir den Term $P(a|\vec{d})$ ausklammern (Distributivgesetz) und rauskürzen. Wir schreiben nun die Summe, die über alle Alinierungen läuft, etwas expliziter auf:

$$\begin{aligned}
&= \frac{\sum_{a:a(i)=j} \prod_{k=1}^{|\vec{e}|} P(e_k|d_{a(k)})}{\sum_{a\in A(|\vec{e}|,|\vec{d}|)} \prod_{k=1}^{|\vec{e}|} P(e_k|d_{a(k)})} \\
(655) \quad &= \frac{P(e_i|d_j) \sum_{a(1)=0}^{|\vec{d}|+1} \cdots \sum_{a(i-1)=0}^{|\vec{d}|+1} \sum_{a(i+1)=0}^{|\vec{d}|+1} \cdots \sum_{a(|\vec{e}|)=0}^{|\vec{d}|+1} \prod_{k=1}^{|\vec{e}|} P(e_k|d_{a(k)})}{\sum_{a(1)=0}^{|\vec{d}|+1} \cdots \sum_{a(|\vec{e}|)=0}^{|\vec{d}|+1} \prod_{k=1}^{|\vec{e}|} P(e_k|d_{a(k)})}
\end{aligned}$$

Wir haben hier nur die Alinierungen explizit ausgeschrieben. Als nächstes benutzen wir den Trick, den wir schon in Formel (647) benutzt haben; durch iteriertes anwenden des Distributivgesetzes können wir das Produkt über die Summen heben:

$$\begin{aligned}
&= \frac{P(e_i|d_j) \sum_{a(1)=0}^{|\vec{d}|+1} \cdots \sum_{a(i-1)=1}^{|\vec{d}|+1} \sum_{a(i+1)=0}^{|\vec{d}|+1} \cdots \sum_{a(|\vec{e}|)=0}^{|\vec{d}|+1} \prod_{k=1}^{|\vec{e}|} P(e_k|d_{a(k)})}{\sum_{a(1)=0}^{|\vec{d}|+1} \cdots \sum_{a(|\vec{e}|)=0}^{|\vec{d}|+1} \prod_{k=1}^{|\vec{e}|} P(e_k|d_{a(k)})} \\
(656) \quad &= \frac{P(e_i|d_j) \prod_{k=1}^{|\vec{e}|} \sum_{a(1)=0}^{|\vec{d}|+1} \cdots \sum_{a(i-1)=1}^{|\vec{d}|+1} \sum_{a(i+1)=0}^{|\vec{d}|+1} \cdots \sum_{a(|\vec{e}|)=0}^{|\vec{d}|+1} P(e_k|d_{a(k)})}{\prod_{k=1}^{|\vec{e}|} \sum_{a(1)=0}^{|\vec{d}|+1} \cdots \sum_{a(|\vec{e}|)=0}^{|\vec{d}|+1} P(e_k|d_{a(k)})} \\
&= \frac{P(e_i|d_j)}{\sum_{a(i)=0}^{|\vec{d}|} P(e_i|d_{a(i)})}
\end{aligned}$$

D.h. am Ende haben wir eine sehr einfache Formel dastehen; wir können die Wahrscheinlichkeit $P(a(i) = j|\vec{e}, \vec{d})$ in linear vielen Schritten über die Länge von $\vec{e}, \vec{d}$ berechnen.

49.4 Der eigentliche Algorithmus

Folgende Konventionen: mit $\mathfrak{K}$ bezeichnen wir unser Korpus; da die Reihenfolge der Satzpaare keine Rolle spielt, nehmen wir an dass $\mathfrak{K} := \{(\vec{e}_i, \vec{d}_i) : i \in I\}$ eine Menge von $|I|$ Satzpaaren ist. Mit lex(E) bzw. lex(D) bezeichnen wir das englische bzw. deutsche Lexikon.

Wir nehmen jetzt den Term und definieren ihn in seiner vereinfachten Form. Da wir die Wahrscheinlichkeiten nur noch lokal berechnen, fallen dabei einige Indizes weg; insbesondere brauchen die Wortpaare, deren Übersetzungswahrscheinlichkeit wir schätzen möchten, keinen Index mehr. Beachten Sie aber dass es sich bei der Umformung um die (verkürzte) Umformung in (656) handelt.

$$\begin{aligned} C(e|d, \vec{e}, \vec{d})) &:= \sum_{a:a(i)=j} [P_a(j|i, \vec{e}, \vec{d}) \sum_{i=1}^{|\vec{e}|} \delta(e, e_i)\delta(d, d_j)] \\ &= \frac{P(e|d)}{\sum_{j=0}^{|\vec{d}|} P(e|d_j)} \sum_{i=1}^{|\vec{e}|} \delta(e, e_i)\delta(d, d_{a(i)}) \end{aligned} \tag{657}$$

Wir ändern nun diese Formel, um sie induktiv anwenden zu können: nehmen Sie an, wir haben eine Sequenz von Wahrscheinlichkeitsfunktionen $P_n : n \in \mathbb{N}_0$. Dann definieren wir

$$C_n(e|d, \vec{e}, \vec{d})) := \frac{P_n(e|d)}{\sum_{j=0}^{|\vec{d}|} P_n(e|d_j)} \sum_{i=1}^{|\vec{e}|} \delta(e, e_i)\delta(d, d_{a(i)}) \tag{658}$$

Wir nehmen also in C_n Referenz auf die Wahrscheinlichkeitsfunktion die wir benutzen. Wir kommen nun zum eigentlichen Algorithmus. Wir setzen für alle $e \in lex(E), d \in lex(D)$,

$$P_0(e|d) := \frac{1}{|lex(E)|} \tag{659}$$

Wir setzen also die Übersetzungswahrscheinlichkeiten uniform. Die (unbedingten) Alinierungswahrscheinlichkeiten setzen wir ebenfalls uniform; daran wird sich auch im Laufe des Algorithmus nichts ändern.

Als nächstes definieren wir:

$$(660)\quad P_{n+1}(e|d) := \frac{\sum_{(\vec{e},\vec{d})\in\mathfrak{K}} C_n(e|d,\vec{e},\vec{d})}{\sum_{e'\in lex(E)}\sum_{(\vec{e},\vec{d})\in\mathfrak{K}} C_n(e'|d,\vec{e},\vec{d})}$$

Und damit sind wir auch schon fertig: wir haben P_0, und gegeben irgendein P_n können wir auch P_{n+1} ausrechnen mithilfe der Funktion C_n. Beachten sie dass Formel (660) genau dasselbe macht wie (650), nur dass

1. wir keine unbedingten Alinierungswahrscheinlichkeiten brauchen,
2. sauber über das Korpus quantifiziert haben, und
3. das Ganze wesentlich einfacher berechnen können!

Was wir also möchten ist $P_n(e|d)$ für ein ausreichend großes $n \in \mathbb{N}$. Zwei Dinge sind entscheidend:

1. Die Folge von Verteilung $P_n(e|d) : n \in \mathbb{N}$ konvergiert gegen eine Verteilung $P_\infty(e|d) : n \in \mathbb{N}$, und
2. $P_\infty(e|d) : n \in \mathbb{N}$ ist ein *lokales Maximum* der Likelihood Funktion von $P(e|d)$ gegeben $\mathfrak{K}$.

49.5 EM für IBM-Modell 1: Ein Beispiel

Nehmen wir an, wir haben die folgenden Daten (ein satzaliniertes, zweisprachiges Korpus):

Korpus:

1. (Hund bellte, dog barked)
2. (Hund, dog)

1. Initialisiere uniform:

$P_0(e\|d)$	dog	barked
Hund	$\frac{1}{2}$	$\frac{1}{2}$
bellte	$\frac{1}{2}$	$\frac{1}{2}$
NULL	$\frac{1}{2}$	$\frac{1}{2}$

2a. Anteilige Häufigkeiten auf Satzebene In Bezug auf Folie 8: Die schwarzen Brüche sind $P_0(e|d)$. Die Summe $\sum_{j=0}^{|\vec{d}|} P(e|d_j)$ wird in der letzten Zeile gebildet. In rot sehen Sie dann die eigentlichen anteiligen Häufigkeiten bzw. wie sie berechnet wurden.

$C_1(e\|d, \vec{e}_1, \vec{d}_1)$	dog	barked
Hund	$\frac{1}{2}\cdot\frac{2}{3} = \frac{1}{3}$	$\frac{1}{2}\cdot\frac{2}{3} = \frac{1}{3}$
bellte	$\frac{1}{2}\cdot\frac{2}{3} = \frac{1}{3}$	$\frac{1}{2}\cdot\frac{2}{3} = \frac{1}{3}$
NULL	$\frac{1}{2}\cdot\frac{2}{3} = \frac{1}{3}$	$\frac{1}{2}\cdot\frac{2}{3} = \frac{1}{3}$
$\sum \quad (\rightarrow Z)$	$\frac{3}{2}$	$\frac{3}{2}$

$C_1(e\|d, \vec{e}_2, \vec{d}_2)$	dog
Hund	$\frac{1}{2}\cdot 1 = \frac{1}{2}$
NULL	$\frac{1}{2}\cdot 1 = \frac{1}{2}$
$\sum \quad (\rightarrow Z)$	1

Die Zahlen $\frac{2}{3}$ ist der Kehrwert des Nenners in 660 (daher Multiplikation statt Division), der sich wiederum durch (658) und (657) berechnen läßt (so kommen wir auf $\frac{3}{2}$).

2b. Anteilige Häufigkeiten auf Korpusebene

$C(e\|d)$	dog	barked	$\sum \quad (\rightarrow C(d))$
Hund	$\frac{1}{3} + \frac{1}{2} = \frac{5}{6}$	$\frac{1}{3}$	$\frac{7}{6}$
bellte	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$
NULL	$\frac{1}{3} + \frac{1}{2} = \frac{5}{6}$	$\frac{1}{3}$	$\frac{7}{6}$

Wir sehen: hier zeigen unsere Daten einen deutlichen Effekt!

3. Neue Parameter

$P_1(e\|d)$	dog	barked
Hund	$\frac{5}{6}\cdot\frac{6}{7} = \frac{5}{7}$	$\frac{1}{3}\cdot\frac{6}{7} = \frac{2}{7}$
bellte	$\frac{1}{2}$	$\frac{1}{2}$
NULL	$\frac{5}{7}$	$\frac{2}{7}$

Iteriere 2a + 2b + 3 Bemerkung: Sowohl die Häufigkeiten als auch die Parameter $P_k(e|d)$ stimmen mit denen des allgemeinen Algorithmus (Folie 6+7) überein!

Aufgabe 12

Abgabe bis zum 4.7. vor dem Seminar.

Führen Sie das Beispiel fort, indem Sie zwei weitere Iterationen des EM-Algorithmus machen, und liefern Sie die resultierenden Übersetzungswahrscheinlichkeiten $P_3(\text{dog}|\text{Hund})$ etc.

50 Naive Bayes Klassifikatoren (aka *idiot Bayes)*

Die Unabhängigkeit der Merkmale, Maximierung von Likelihood x Apriori.

Bei Naive Bayes Klassifikation werden eine Menge Merkmale als Vorhersage genommen, unter der Annahme, dass diese Merkmale untereinander unabhängig sind, und alle gleich wichtig. Darin besteht soz. die Naivität. Man sagt also:

$P(w_1...w_i|x) = P(w_1|x)...P(w_i|x)$

Also:

$P(x|w_1...w_i) = P(w_1...w_i|x)\frac{P(x)}{P(w_1...w_i)}$

Durch unsere Annahmen bekommen wir also:

$P(x|w_1...w_i) = P(w_1|x)...P(w_i|x)\frac{P(x)}{P(w_1)...P(w_i)}$

Das sind normalerweise alles Wahrscheinlichkeiten, die wir sehr gut und einfach abschätzen können (einfache Wahrscheinlichkeiten haben repräsentative Häufigkeiten).

Der Vorteil ist also dass das Modell sehr einfach gelernt werden kann und robust ist. Der Nachteil ist, theoretisch, dass es auf Annahmen basiert, die im Normalfall falsch sind. Bei einfachen Anwendungen ist das praktisch kein Problem; bei komplexeren Anwendungen, oder wenn man passable Ergebnisse verbessern will, dann kann das ein Problem sein.

50.1 Der naive Bayes Spamfilter

Worte sind unabhängig! Wahrscheinlichkeiten sind Häufigkeiten.

50.2 BN

Bayesianische Netze – sternförmig. Die Generalisierung. Hier geht es soz. darum, die optimale Belegung eines BN zu finden. Das macht nicht soviel Sinn – besser ein CRF

51 Bayesian Learning

Wir sagen, es gibt (diskrete) Funktion, und einen uniform normalverteilten Fehlen (E=0,V unbekannt). Wir suchen diejenige Hypothese, die am Wahrscheinlichsten ist gegeben die Daten.

Insbesondere folgende Vorteile: – Die Wahrscheinlichkeiten können ständig geupdatet werden – Das Modell liefert uns zugleich eine Quantifizierung der Unsicherheit – Alles ist probabilistisch abgedeckt – kein Overfitting, nicht zuwenig Daten (das Modell selbst gibt seine Qualität an)

Vorsicht: bei gleicher Hypothesenwahrscheinlichkeit wird das eben ML!

Ein Modell, dazu Parameter. Die knnen kontinuierlich oder diskret sein. Wir suchen die besten Parameter!

Alternativ: Modelle vergleichen. Das Integral über diese Funktion!

52 Monte Carlo Methoden

Monte Carlo estimates

52.1 Gibbs-Sampling

?

52.2 MCMC

?

53 Hidden Markov Modelle

54 Zur Methodik des maschinellen Lernens

54.1 Abriß der Methode

Im Rahmen des maschinellen Lernens und der Wahrscheinlichkeitstheorie wird natürlich oft die Frage gestellt: ist es so dass ein Lern-Algorithmus bessere Ergebnisse erzielt als ein anderer? Tatsächlich wird das meistens empirisch überprüft, nach folgender Methode:

Wir haben einen Datensatz D gegeben.

Wir partitionieren den Datensatz D in zwei disjunkte Teilsätze $D_{training}, D_{test}$.

Wir nutzen nun $D_{training}$, um unseren Klassifikator zu **trainieren**, also z.B. um unseren Entscheidungsbaum festzulegen, unser BN zu induzieren etc.

Zuletzt nutzen wir D_{test}, um unseren Klassifikator zu **evaluieren**, d.h. wir prüfen, wie gut er auf diesen Daten abschneidet. Diese Evaluation wird normalerweise als Kennzeichen aufgefasst für die Qualität unseres Klassifikators.

Wir sehen also, dass diese Methodik in gewissem Sinne eine Alternative zur Methodik der klassischen/bayesianischen Statistik liefert: anstatt den gesamten Datensatz auf Regelmäßigkeiten zu prüfen, nutzen wir einen Teil, ziehen Generalisierungen, und prüfen dann, ob die Generalisierungen richtig sind. Das soll uns vor dem Problem des *Overfitting* schützen: jeder noch so große Datensatz erlaubt "falsche" Generalisierungen, die auf Artefakten beruhen. Ein einfaches Beispiel hierfür: jedes Korpus K hat einen längsten Satz S mit k Worten. Dementsprechend könnten wir immer den Schluss ziehen:

$$\text{jeder Satz hat Länge} \leq k.$$

Die Tatsache, dass wir unsere Generalisierungen prüfen auf einem Datensatz, den wir vorher nicht gesehen haben, soll das verhindern bzw. einschränken. Insbesondere glaubt man:

> Wenn eine Generalisierung über $D_{training}$ auch für D_{test} gilt, dann hat sie gute Chancen, allgemein korrekt zu sein.

Dafür gibt es natürlich keine Garantie, aber zumindest zeigt dass, das die Generalisierungen keine Artefakte der Daten sind.

54.2 Zwei Probleme

1. Es gibt bei dieser Methode allerdings 2 Dinge zu beachten. Das erste ist folgendes Problem: das Aufsplitten der Daten ergibt nur dann einen Sinn und einen echten Vorteil, wenn der Datensatz für den Lernalgorithmus **tatsächlich unsichtbar** ist. Das ist aber leichter gesagt als getan: man nehme z.B. folgenden Fall:

> Wir trainieren den Klassifikator C auf dem Datensatz $D_{training}$; dann testen wir ihn auf D_{test}. Dabei bemerken wir: C macht auf D_{test} einen sehr charakteristischen Fehler relativ häufig. Also ändern wir C dergestalt zu C', dass wir wissen, dass es diesen Fehler nicht mehr macht. Die Ergebnisse sind dementsprechend auch besser.

Ist nun aber C' besser als C? Das lässt sich schwer sagen, und zwar aus folgendem Grund: zwar wurde C' **optimiert**, aber: es wurde optimiert im Hinblick auf D_{test} – von daher ist es nicht verwunderlich, dass C' besser darauf abschneidet als C! Man sagt auch: es sind Informationen von D_{test} nach C' eingeflossen. Der gesamte Vorteil der obigen Methode bestand aber darin, dass das nicht geschah, damit wir eben prüfen konnten dass Generalisierungen nicht nur im Hinblick auf die Daten geschehen! Um also den Vorteil wirklich zu verifizieren, bräuchten wir einen neuen Testsatz D_{test2} – der selten zur Verfügung steht.

2. Das zweite Problem ist grundlegender: wir wissen nicht, ob unser Klassifikator C *nur zufällig* eine gute Performanz auf D_{test} hat. Denn in vielen Fällen ist die Menge der klassifizierten Objekte unendlich, und D ist nur ein kleiner Ausschnitt daraus (eine sog. Stichprobe). Und wir wissen nie, ob wir nicht in dieser Stichprobe eine systematische Tendenz haben (durch die Art, wie wir sie genommen haben), die auf der Population insgesamt nicht erfüllt ist!

> In unserem Beispiel der Entscheidungsbäume: es kann gut sein, dass in unseren Daten gewisse seltene Konstellationen niemals vorkommen. Dementsprechend unterschätzen wir gewisse Faktoren, weil sie einfach in D (also sowohl $D_{training}$ als auch D_{test}) nie eine Rolle spielen.

Hiergegen kann man natürlich nichts machen. Insbesondere führt uns diese Überlegung auf die NFL-Theoreme.

55 *No free lunch* – Gibt nix umsonst

55.1 Einleitung

Jeder Kurs über das maschinelle Lernen sollte die berühmten *no-free-lunch* (NFL) Theoreme behandeln, da sonst die Gefahr besteht, dass man grundsätzliche Dinge missversteht. NFL heißt soviel wie "gibt nix umsonst"; warum sie so heißen sollte bald klar werden. Diese Theorem sind von David Wolpert, dessen Papiere aber etwas sperrig sind.

Wir haben bis hierher viel über verschiedene Algorithmen besprochen, haben sehr elaborierte Methoden gesehen die in der Praxis sehr gut funktionieren. Das könnte uns zu dem Glauben verleiten, dass manche Algorithmen allgemein besser sind als andere. Die NFL-Theoreme besagen nun, dass genau das nicht stimmt. Auf einer allgemeinen Klasse von Lernproblemen lernen alle – wirklich alle, auch die sinnlosesten – Lernalgorithmen gleich gut. Und wenn ein Algorithmus auf einer Teilklasse von dieser Klasse besser funktioniert – z.B. der Teilklasse die wir beobachten – dann ist das nur deswegen, weil er auf einer anderen Teilklasse schlechter funktioniert.

Im Umkehrschluss bedeutet dass: wir können empirisch nie etwas über einen Algorithmus erfahren, sondern nur über diejenige Teilklasse unseres Problems, mit der wir getestet haben. Angewandtes maschinelles Lernen sagt uns also nur etwas über Phänomene, nicht über Algorithmen. Aber was genau heißen diese Begriffe, die wir eben benutzt haben? Das machen wir jetzt präzise.

55.2 Die NFL Theoreme und was sie bedeuten

Das Ziel (die objektive Funktion) Wir suchen eine Funktion

$$f : X \to Y$$

Das ist die objektive Funktion. Wolperts Ansatz ist allerdings etwas allgemeiner: er betrachtet **probabilistische Funktionen**, d.h. Funktionen, die für eine Eingabe $x \in X$ nicht ein $y \in Y$ ausgeben, sondern eine Wahrscheinlichkeitsverteilung

$$f(x) : Y \to [0, 1]$$

Natürlich ist leicht zu sehen, dass das nur eine Verallgemeinerung darstellt. Uns interessieren also Funktionen

$$f : (X \times Y) \to [0, 1]$$

so dass

$$\sum_{y \in Y} f(x, y) = 1, \text{ f.a. } x \in X$$

Wir bezeichnen die Menge dieser Verteilungen mit

$$prob(Y^X)$$

In unseren Daten finden wir allerdings keine Verteilungen, sondern tatsächlich Datenpunkte (x, y); also ist

$$D \subseteq X \times Y$$

Wir haben weiterhin die Annahme, dass alle Funktion $f \in prob(Y^X)$ apriori gleich wahrscheinlich sind – wir lernen also ohne Vorwissen. Algorithmen (bei uns) sind Funktionen, die Versuchen die richtige Funktion zu approximieren.

Kostenfunktion Wenn wir trainieren und bekommen ein falsches Ergebnis (unsere Vorhersage unterscheidet sich von der Beobachtung), dann kostet das, und dafür brauchen wir eine Kostenfunktion

$$k : Y \times Y \to \mathbb{R}.$$

Welche das ist, spielt keine Rolle, sie muss nur eine Bedingung erfüllen, nämlich sie muss **homogen** sein, das bedeutet folgendes. Sei δ die Kronecker-delta Funktion definiert durch

$$\delta(x, y) = \begin{cases} 1, \text{ falls } x = y \\ 0 \text{ andernfalls} \end{cases} \tag{661}$$

Weiterhin bezeichnen wir

- mit c eine beliebige Konstante;
- mit y_f das "objektiv richtige" y so dass (x, y_f) in unseren Daten D;
- mit y_h die Vorhersage unseres Algorithmus.

Nimm nun die Funktion

$$\delta(c, k(y_h, y_f)) : Y^3 \to \{0, 1\} \tag{662}$$

in drei Argumenten, die uns 0 oder 1 liefert, je nachdem ob $k(x, y)$ den Wert c liefert. Wir können aus dieser Funktion nun das dritte Argument "herausmarginalisieren":

$$\Lambda(c, y_h) = \sum_{y_f \in Y} \delta(c, k(y_h, y_f)) : Y^2 \to \{0, 1\} \tag{663}$$

Diese Funktion zählt, wie oft, für gegebenes c und y_h, die Distanz von beliebigen y_f und y_h den Wert c beträgt. Wir sagen k ist **homogen**, falls gilt: für beliebige y, y', c gilt

$$\Lambda(c, y) = \Lambda(c, y') \tag{664}$$

Das bedeutet: mit unserer Hypothese können wir keinen Einfluss darauf nehmen (unabhängig von der wahren, objektiven Funktion f), wie oft wir einen gewissen Kostenwert bekommen. Anders gesagt:

- In völliger Unkenntnis der objektiven Funktion wird unsere Kostenfunktion keine Hypothese bevorzugen.

Das ist eine sehr allgemeine Bedingung die in der Praxis so gut wie immer erfüllt wird, da sie einfach besagt: es gibt nach Kostenfunktion keine bessere oder schlechtere Hypothesen, wenn wir noch keine Daten gesehen haben. Da die Funktion Λ, falls sie auf einer homogenen Kostenfunktion basiert, dem zweiten Argument keine Rolle zumisst, können wir Λ als auch eine Funktion mit einem Argument auffassen, und schreiben in Zukunft

(665) $\Lambda(c)$

Eine einfache Funktion, die das nicht erfüllt, wäre z.B. eine Funktion

(666) $k(y_h, y_f) = max(\epsilon, y_h) \cdot (y_h - y_f)^2, \quad \epsilon > 0$

(das ϵ ist da um einen trivial beste Hypothese $y_h = 0$ zu verhindern). Diese Funktion bevorzugt kleine y_h unabhängig von y_f, ist also voreingenommen.

Stichproben sammeln Wir nehmen an, es gibt eine Wahrscheinlichkeitsverteilung

$$\theta : X \to [0, 1],$$

mittels derer wir Stichproben x aus X entnehmen, um dann schauen welchen Wert f dieser Stichprobe zuweist (wir beobachten also einen Punkt $(x, f(y))$). Das liegt unserem Training *und* unseren Tests zugrunde. Was wichtig ist, ist die Stichprobenwahrscheinlichkeit (likelihood) eines Datensatzes D, gegeben durch

$$P(D|f)$$

Diese Wahrscheinlichkeit hängt natürlich von θ ab. Wir sagen dass diese Verteilung (likelihood) **vertikal** ist, falls sie unabhängig ist von $f(x, y_f)$, falls x nicht in D vorkommt. Z.B. eine typische likelihood wäre gegeben durch

$$P(D|f) = \prod_{(x,y)\in D} \theta(x) f(x, y) \tag{667}$$

(das wäre IID sampling). Wir brauchen Stichproben sowohl zum Testen als auch zum Training, und wir nehmen an das beide Male dieselbe Verteilung und Methode zugrunde liegt (das ist im Normalfall gewährleistet).

Der Algorithmus und sein Training Was wir weiterhing brauchen ist der Algorithmus a. Der Algorithmus ändert in jedem Schritt seine vorhersagen: initialisiert ist er eine Funktion

$$a_0 \in Y^X.$$

Nun ist es wichtig, dass der Algorithmus seine Daten in einer bestimmte Reihenfolge bekommt. Wenn er Kosten k_1 für x_1 und seine Vorhersage $a_0(x_1)$ bekommt:

(668) $k_1 := k(y_1, a_0(x_1))$

dann ändert er sich zu $a_1 \in X^Y$ usw. Der Algorithmus ist erstmal eine Folge von Funktionen

$$\langle a_0 : X \to Y, a_1 : X \to Y, a_2 : X \to Y, ... \rangle$$

Wir geben dem Algorithmus das Argument x_1, berechnen Kosten k_1, liefern diesen Wert als Eingabe (backpropagation), und dann bekommen wir a_1. Als nächstes bekommen wir

(669) $k_2 = k(a_1(x_2), y_2)$

Das ist **Training** unseres Algorithmus auf unseren Daten D.

Wahrscheinlichkeit von Kosten im Test Wir können nun definieren, was die Wahrscheinlichkeit ist, gewisse Kosten zu haben, gegeben einen Datensatz D. Wir fassen dafür k als eine Zufallsvariable auf:

$$P(k = x|f, a, D)$$

bezeichnet die Wahrscheinlichkeit, im nächsten gesammelten Beispiel von der zugrundeliegenden Verteilung f, mit Algorithmus a trainiert auf Datensatz D, die Kosten x zu haben. Z.B.

(670) $P(k = 0|f, a, D)$

Das ist die Wahrscheinlichkeit, das nächste Beispiel richtig zu raten. Wie ist diese Funktion genau definiert? Das hängt von unserem sampling ab, im IID-Fall wäre das wie folgt:

(671) $$P(k = z|f, a, D) = \sum_{k(y,a_D(x))=z} \theta(x) f(x, y)$$

Also die Wahrscheinlichkeit, das wir ein x ziehen, mal die Wahrscheinlichkeit dass $f(x) = y$, für alle (x, y) so dass $k(a_D(x), y) = z$, und das aufsummiert. (Die Bedingung: f, D muss konsistent sein?)

55.3 Einige NFL Theoreme

Wichtig ist: alle Theoreme beziehen sich auf *off training set error* (OTS), also der Fehler wird nur auf Datenpunkten gemessen, die nicht im unserem Trainingsset vorkamen. Andernfalls würde das Ergebnis nicht gelten, den auf diesen Punkten können wir sehr leicht bessere/schlechtere Algorithmen ausmachen.

Das erste Wolpertsche NFL Theorem besagt (grob):

Theorem 28 (NFL 1) *Sei Λ wie oben für eine homogene Kostenfunktion definiert. Dann gilt: der normalisierte Durchschnitt über alle f im Kandidatenraum von $P(k = x|f, a, D)$ ist gleich $\Lambda(x)/r$ für ein Konstante r, unabhängig von a. Also insbesondere: für beliebige Algorithmen $a1, a2$ gilt:*

$$\sum_{f \in prob(Y^X)} P(k = z|f, D, a1) = \sum_{f \in prob(Y^X)} P(k = z|f, D, a2)$$

Das bedeutet: die Wahrscheinlichkeit, dass wir mit einem Algorithmus, gegebenen Trainingsset in Unkenntnis der objektiven Funktion als nächstes einen Fehler von k haben (z.B. $k = 0$, also richtig zu liegen), ist für **jeden Algorithus gleich**, und eine einfache Funktion über unsere Kostenfunktion; der Algorithmus ist irrelevant.

Der nächste Schritt besteht darin, das ganze unabhängig zu machen von D – der Datensatz soll also nicht gegeben sein; stattdessen nur seine Größe. Das Problem dabei ist, dass verschiedene fs unterschiedlich beitragen, da sie den Daten unterschiedliche Wahrscheinlichkeiten zuweisen. Wir haben:

$$(672)\quad P(k = z|a, f, |D| = m) = \sum_{|D|=m} P(k = z|f, a, D)P(D|f)$$

Das folgt aus den normalen Regeln zur Marginalisierung, und $P(D|f)$ ist *nicht* uniform über f (die likelihood der Daten ändert sich mit der objektiven Funktion). Dennoch gibt es auch hierfür ein NFL Theorem:

Theorem 29 (NFL 2) *Sei Λ wie oben für eine homogene Kostenfunktion definiert, und die Wahrscheinlichkeit $P(D|f)$ eine vertikale Verteilung für alle D, f. Dann gilt: der normalisierte Durchschnitt über alle f im Kandidatenraum von $P(k = x|\ f, a, |D| = m)$ ist gleich $\Lambda(x)/r$ für ein Konstante r, unabhängig von a. Also insbesondere: für beliebige Algorithmen $a1, a2$ gilt:*

$$\sum_{f \in prob(Y^X)} P(k = z|\ f, |D| = m, a1) = \sum_{f \in prob(Y^X)} P(k = z|\ f, |D| = m, a2)$$

Das bedeutet: auch wenn wir nur die Größe der Trainingsdaten als gegeben betrachten, sind alle Algorithmen gleich gut. Um das Ergebnis zu verstehen, muss man sehen dass f, gegeben im Term, ja die eigentliche Unbekannte ist. Wenn nun ein Algorithmus $a1$ auf einer Teilmenge (der Größe m) von $f \in Y^X$ konsisten nur geringere Kosten erzeugt als $a2$ (also konsistent besser ist), dann bedeutet dass das wir – vereinfacht gesprochen – eine entsprechende Funktion $f' \in prob(Y^X)$ haben, auf der $a1$ *konsistent schlechter* abschneidet.

Permutierbare Funktionen – ein notwendiges& hinreichendes Kriterium Man kann dieses Theorem noch etwas stärker formulieren: die Bedingung $f \in Y^X$ (also der gesamte Funktionenraum ist involviert) ist etwas zu schwach. Eine **Permutation** ist eine Funktion

$$\pi : M \to M, \text{ wobei } \pi^{-1} \circ \pi(x) = x$$

Also eine Bijektion auf einer Menge. Sei $F \subseteq Y^X$ eine Menge von Funktionen. Wir sagen diese Menge ist *geschlossen unter Permutation*, wenn folgende Bedingung erfüllt wird:

$$f \in F \Longrightarrow f \circ \pi \in F$$

Also wir können Eingaben beliebig permutieren, wir bleiben in der Klasse. Das erlaubt es uns z.B. den Ausgaberaum beliebig einzuschränken. Als Beispiel nimm folgende Klasse: sei $X = \mathbb{R} = Y$. Dann ist

- $F_{mon} \subseteq Y^X$, die klasse der monotonen Funktionen, ist nicht geschlossen unter Permutation (setze $\pi(n) = n + 1$ falls n gerade), $\pi(n) = n - 1$ falls n ungerade, $\pi(x) = x$ falls $n \notin \mathbb{N}$.
- Die Klasse der linearen Funktionen (oder Polynome) ist nicht geschlossen unter Permutation (nimm obiges Beispiel, das macht die Funktion unstetig)
- Die Klasse der Wahrscheinlichkeitsfunktion $f : X \to [0, 1]$ ist geschlossen unter Permutation.

Der formale Abschluss unter Permutation hat eine intuitive Bedeutung: ein Funktionenraum ist geschlossen unter Permutation, wenn wir in gewissem Sinne keine *apriori*-Aussage über Eingaben – Ausgabe Relationen machen können. Nun lässt sich das Theorem stärker formulieren:

Theorem 30 (NFL3) *Für zwei beliebige Algorithmen a_1, a_2 gilt:*

$$\sum_{f \in F} P(k = z|\ f, |D| = m, a_1) = \sum_{f \in F} P(k = z|\ f, |D| = m, a_2)$$

genau dann wenn F unter Permutation abgeschlossen ist.

Der Abschluss unter Permutation ist also hinreichend und notwendig, damit die NFL Theoreme gelten. Ähnlich lassen sich andere Theoreme stärker formulieren, wir lassen das aber. Diese Theoreme sind sehr robust, und gelten noch für eine ganze Reihe weiterer Fälle. Das sog. zweite NFL Theorem ist aber z.B. wesentlich komplexer und nimmt an, dass sich die objektive Funktion über die Zeit ändert. Das macht natürlich einen Unterschied, da die Tatsache, dass wir eine gewissen Teil als Daten zuerst sehen, natürlich einen Unterschied, da unsere Funktion später sich anders verhalten kann. Auch diese Erweiterung ändert also nichts an den Ergebnissen.

55.4 Was bedeutet das also...

... wenn eine Klasse von Lernalgorithmen (z.B. neuronale Netze) auf einer gewissen Klasse von Problemen gut funktioniert?

- Man darf die NFL Theoreme nicht dahingehend missverstehen, dass das reiner Zufall wäre, das wäre praktisch ausgeschlossen!

Stattdessen bedeutet das soviel wie: wir haben eine Teilklasse von Funktionen, die nicht unter Permutation geschlossen sind, anders gesagt,

- wir machen gewisse *apriori* Annahmen über unseren möglichen Funktionenraum!

Zu sagen, dass ein Ansatz gut funktioniert, bedeutet: zu sagen, dass unser Funktionenraum auf eine gewisse Weise von vornherein eingeschränkt ist. Wir machen also Annahmen über unsere möglichen objektiven Funktionen. Das bedeutet, anders gesagt: effektives Lernen ist nur möglich, wenn wir starke Annahmen machen über was wir lernen wollen. Es gibt aber umgekehrt keine Möglichkeit, objektiv zu prüfen ob diese Annahmen richtig sind.

Das eine Klasse von Algorithmen sehr erfolgreich ist (SVM, Deep Learning) bedeutet: der Funktionenraum ist so, dass er gut zu diesen Algorithmen passt.

55.5 NFL und Probabilistische Optimierung

Es gibt einen interessanten Zusammenhang zwischen NFL und probabilistischer Optimierung. Unter probabilistischer Optimierung verstehen wir hier folgendes: wir nehmen an, wir haben ein diskretes Klassifikationsproblem, sagen wir Einfachheit halber

$$f : M \to \{0, 1\}$$

Dieses Problem hat eine gewisse Zeitkomplexität. Bsp. dafür sind:

- Gegeben eine CFG G und ein Wort w, ist $w \in L(G)$?
- Ist eine gewisse Formel α der Aussagen/Prädikatenlogik eine Tautologie? (NP vollständig)

Nun gibt es verschiedene Methoden, diese Probleme anzugehen; eine Methode wäre, Wahrscheinlichkeiten zu nutzen (z.B. nutzen wir eine Strategie wie *random walk*). Das kann mitunter sehr effektiv sein. Die NFL Theoreme besagen nun: wann immer eine gewisse probabilistische Strategie auf einer Teilklasse von M besser arbeitet, wird sie auf einer anderen Teilklasse schlechter abschneiden (vorausgesetzt M ist abgeschlossen unter Permutation im obigen Sinn). Das bedeutet: alle probabilistischen Methoden sind im Mittel gleich gut!