41 "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

$$\int_0^1 \int_0^1 |x - y| dx \; dy = 2 \int_0^1 \int_0^1 x - y \; dx \; dy = \frac{1}{3} \tag{516}$$

Im Einheitsquadrat ist das ganze schon etwas komplizierter.

Die (einfache) Distanz zweier Punkte ist definiert durch:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \tag{517}$$

Die durchschnittliche Distanz wäre:

$$\int_0^1 \int_0^1 \int_0^1 \int_0^1 d(p, q) = \int_0^1 \int_0^1 \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} dp_1 \; dp_2 \; dq_1 \; dq_2 \approx 0.52 \tag{518}$$

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.

42 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.