Einführung in Machine Learning und Klassifikationsalgorithmen

Fabian Thorand

Internationale Sommerakademie der Studienstiftung in Rot an der Rot

20. August 2015

Was ist maschinelles Lernen?

- Ableitung von Wissen aus Beispielen/Erfahrung

Was ist maschinelles Lernen?

- Ableitung von Wissen aus Beispielen/Erfahrung
- dabei: Verallgemeinerung, nicht bloßes „Auswendiglernen“.

Arten des maschinellen Lernens

1. Überwachtes Lernen (Supervised Learning)

Arten des maschinellen Lernens

1. Überwachtes Lernen (Supervised Learning)
2. Unüberwachtes Lernen (Unsupervised Learning)

Arten des maschinellen Lernens

1. Überwachtes Lernen (Supervised Learning)
2. Unüberwachtes Lernen (Unsupervised Learning)
3. Bestärkendes Lernen (Reinforcement Learning)

Herausforderungen

Wahl des richtigen Modells (Bias-Varianz-Dilemma):

- Zu flexibel: Overfitting
- Zu inflexibel: Underfitting

Herausforderungen

Wahl des richtigen Modells (Bias-Varianz-Dilemma):

- Zu flexibel: Overfitting
- Zu inflexibel: Underfitting

Unterschiedliche Skalen der zu lernenden Features (Bsp. Schiff)

- Tiefgang ($\sim 10^1\,\mathrm{m}$) vs. Maschinenleistung ($\sim 10^6\,\mathrm{PS}$)
- Daten möglichst normieren

Herausforderungen

Wahl des richtigen Modells (Bias-Varianz-Dilemma):

- Zu flexibel: Overfitting
- Zu inflexibel: Underfitting

Unterschiedliche Skalen der zu lernenden Features (Bsp. Schiff)

- Tiefgang ($\sim 10^1\,\mathrm{m}$) vs. Maschinenleistung ($\sim 10^6\,\mathrm{PS}$)
- Daten möglichst normieren

Zahl der Dimensionen

- Wenige Stichproben in hochdimensionalem Raum schwer zu klassifizieren
- “curse of dimensionality”

Überwachtes Lernen

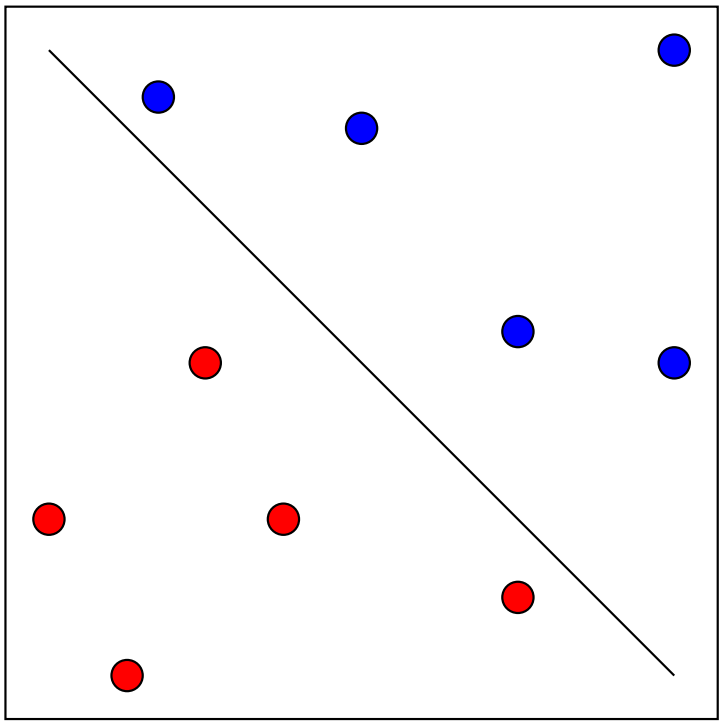
Anwendungsgebiete

- Anpassen von Modellen (Regression)
- Klassifikation

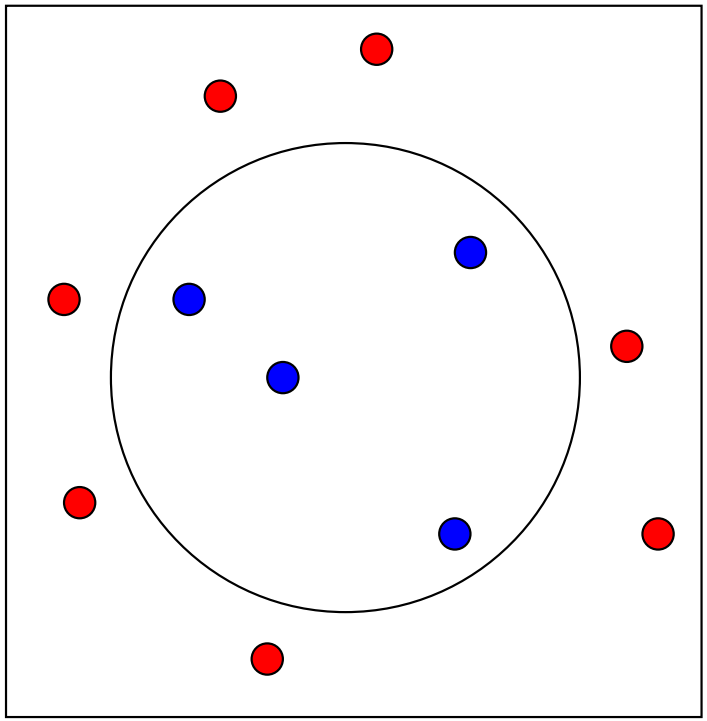
Trainingsmenge

Datenpunkte mit *Klassifikation/Zielwert*

Lineare Klassifikation



(a) linear separierbar



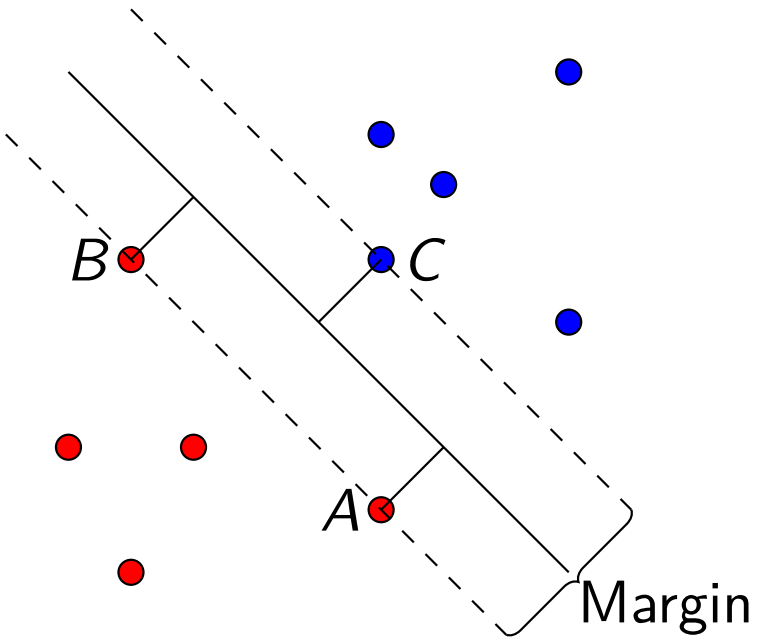
(b) nicht-linear separierbar

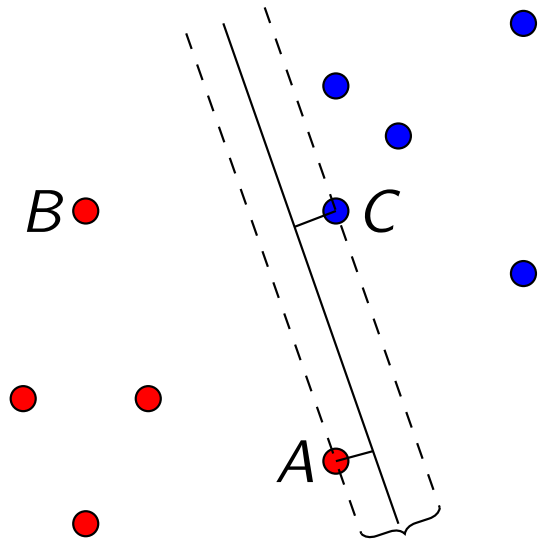
Rückblick: Perzeptron

Binäre Klassifikation parametrisiert über $w \in \mathbb{R}^n, b \in \mathbb{R}$

- Trennende Hyperebene: $f_{w,b}(x) = \langle w, x \rangle - b$
- Ausgabefunktion: $h_{w,b}(x) = sign(f_{w,b}(x))$
- Trainingsdaten $x^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{+1, -1\}$, $i \in \{1, \ldots, m\}$

Margin





Support Vector Machines (SVM)

Binäre Klassifikation parametrisiert über $w \in \mathbb{R}^n, b \in \mathbb{R}$

$$h_{w,b}(x) = sign(\langle w, x \rangle + b) \text{ mit } sign(z) = \begin{cases} +1 & \text{falls } z \geq 0 \\ -1 & \text{sonst} \end{cases}$$

Finde w und b, sodass der Margin maximiert wird.

Formalisierung des Margins

Definition (Funktionaler Margin)

Abstand in der Klassifizierungsfunktion

$$\begin{aligned} \hat{\gamma}^{(i)} &:= y^{(i)}(\langle w, x^{(i)} \rangle + b) \\ \hat{\gamma} &:= \min_{i=1,\ldots,m} \hat{\gamma}^{(i)} \end{aligned}$$

Skalierung von w und b führt zu verschiedenen funktionalen Margins für die gleichen Testdaten!

Formalisierung des Margins (fort.)

Definition (Geometrischer Margin)

Tatsächlicher (geometrischer) Abstand

$$\begin{aligned} \gamma^{(i)} &:= y^{(i)}\left(\left\langle \frac{w}{\|w\|}, x^{(i)} \right\rangle + \frac{b}{\|w\|}\right) &&= \frac{1}{\|w\|}\hat{\gamma}^{(i)} \\ \gamma &:= \min_{i=1,\ldots,m} \gamma^{(i)} &&= \frac{1}{\|w\|}\hat{\gamma} \end{aligned}$$

- Wir fordern $\hat{\gamma} = 1$
- $\gamma = \frac{1}{\|w\|}$ wird dann maximal, wenn $\|w\|$ minimal wird

Optimierungsproblem SVM

Zu lösendes Problem

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 \\ \text{u.d.N.} \quad & y^{(i)} \cdot (\langle w, x^{(i)} \rangle + b) \geq 1, i = 1, \ldots, m \end{aligned}$$

Duales Problem

Einige Umformungen und Ausnutzung der KKT-Bedingungen[1] führen zum äquivalenten dualen Problem

Duales Problem

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ \text{u.d.N.} \quad & \alpha_i \geq 0, i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned}$$

[1] Karush-Kuhn-Tucker-Bedingungen

Duales Problem (fort.)

Dabei gilt

- $\alpha_i > 0$ gdw. $x^{(i)}$ ein Stützvektor ist
- $w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$
- $b = -\frac{1}{2}(\max_{i:y^{(i)}=-1} \langle w, x^{(i)} \rangle + \min_{i:y^{(i)}=+1} \langle w, x^{(i)} \rangle)$

Duales Problem (fort.)

Dabei gilt

- $\alpha_i > 0$ gdw. $x^{(i)}$ ein Stützvektor ist
- $w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$
- $b = -\frac{1}{2}(\max_{i:y^{(i)}=-1} \langle w, x^{(i)} \rangle + \min_{i:y^{(i)}=+1} \langle w, x^{(i)} \rangle)$

Beobachtung

Die Trainingsvektoren $x^{(i)}$ werden nur innerhalb von Skalarprodukten verwendet.

Kernel-Trick

Idee

Nicht-linear separierbare Daten sind in einem höher-dimensionalen Raum/mittels anderer Features möglicherweise doch linear separierbar.
(Bei gegebener Transformation $\phi : \mathbb{R}^n \to \mathbb{R}^d$)

Problem

$\phi(x)$ kann teuer zu berechnen sein (bei sehr vielen Dimensionen)

Kernel-Trick – Umsetzung

Erinnerung

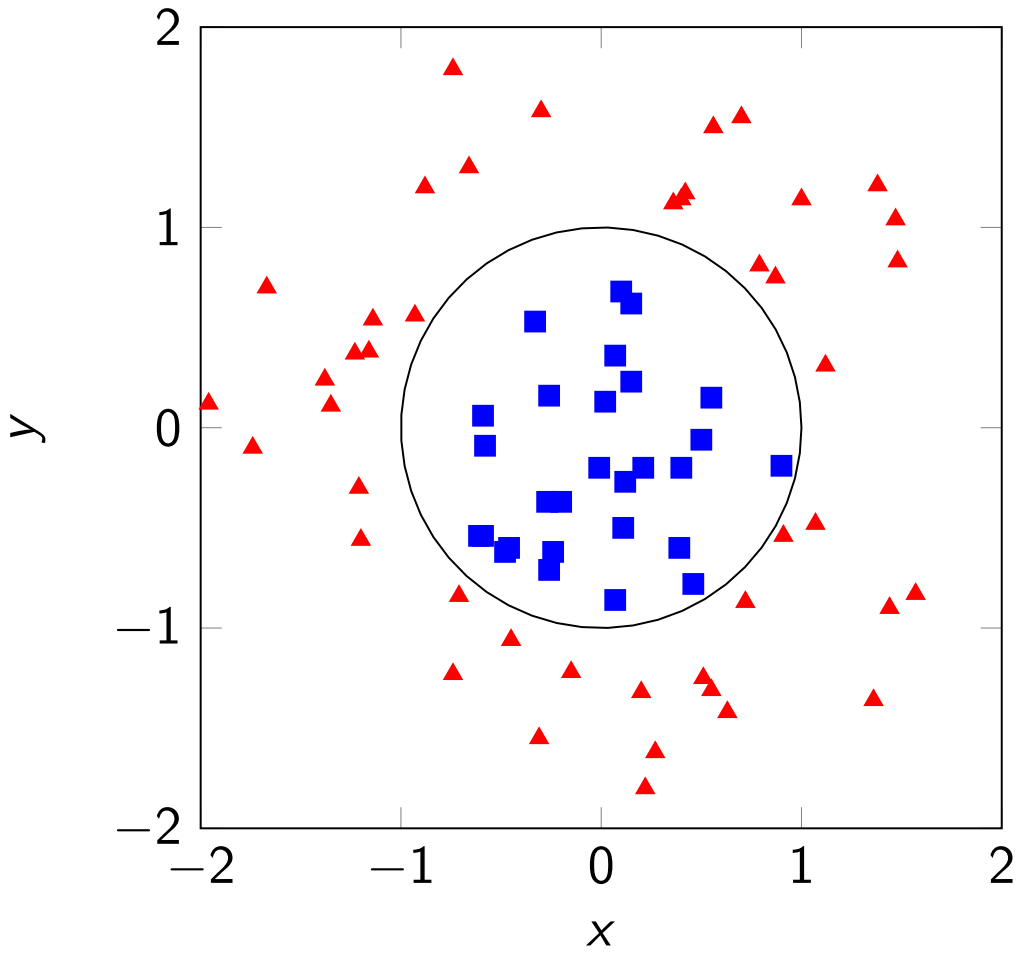
Datenvektoren werden nur in Skalarprodukten verwendet

Ersetze $\langle \phi(x), \phi(y) \rangle$ durch Kernelfunktion

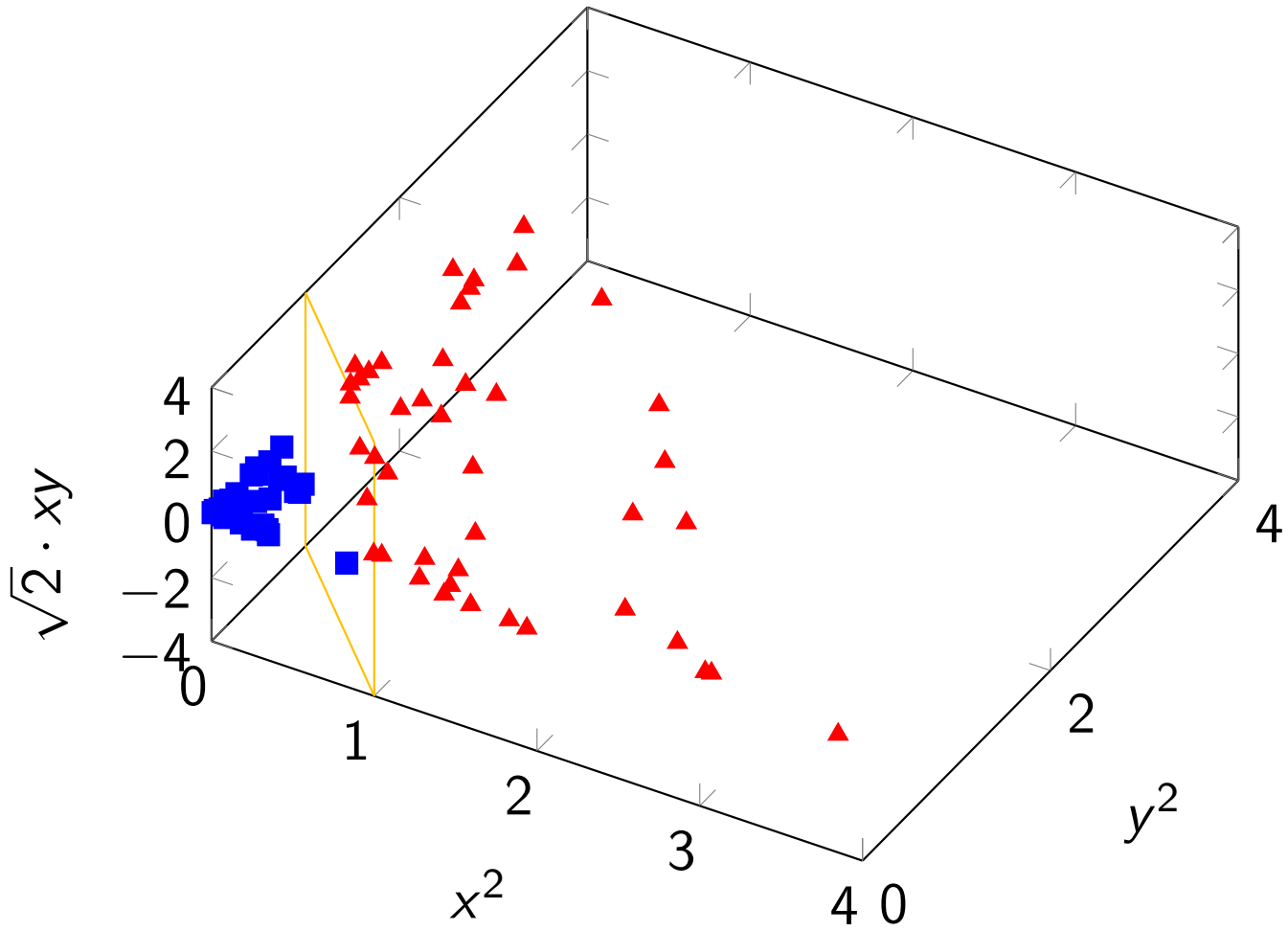
$$\begin{aligned} K : \mathbb{R}^n \times \mathbb{R}^n &\to \mathbb{R} \\ (x, y) &\mapsto \langle \phi(x), \phi(y) \rangle \end{aligned}$$

K ist in vielen Fällen deutlich einfacher zu berechnen

Kernel-Trick – Beispiel I



Kernel-Trick – Beispiel II



Kernel-Trick – Beispiel III

$$\phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} \cdot x_1 x_2 \end{pmatrix}$$

$$K\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}\right) = \left\langle \phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right), \phi\left(\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}\right) \right\rangle = \left\langle \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right\rangle^2$$

Nicht exakt linear separierbare Daten

- Kernel-Trick funktioniert nicht immer
  Bsp.: stark verrauschte Daten
- Einführung von Schlupf-Variablen

Zu lösendes Problem

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{u.d.N.} \quad & y^{(i)} \cdot \left(\langle w, x^{(i)}\rangle + b\right) \geq 1 - \xi_i, i = 1, \ldots, m \\ & \xi_i \geq 0, i = 1, \ldots, m \end{aligned}$$

Ungenauigkeiten werden zugelassen, aber mit Faktor C bestraft.

Nicht exakt linear separierbare Daten (fort.)

Duales Problem

$$\begin{aligned} \max_\alpha \quad & W(\alpha) := \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ \text{u.d.N.} \quad & 0 \leq \alpha_i \leq C, i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned}$$

SVM – Algorithmus

SMO (sequential minimal optimization) [3]

Wiederhole bis zur Konvergenz:

1. Wähle Koordinatenpaar α_i, α_j (welches größten Fortschritt ermöglicht)
2. Optimiere $W(\alpha)$ bzgl. α_i und α_j, fixiere die restlichen Koordinaten

Naive Bayes

- Feature-Vektor: $\mathbf{x} = (x_1, \ldots, x_n)^T$
- Kategorien: $C_1, \ldots, C_K$
- Ziel: Bestimmung von $p(C_k|\mathbf{x})$ für $k = 1, \ldots, K$

Naive Bayes

- Feature-Vektor: $\mathbf{x} = (x_1, \ldots, x_n)^T$
- Kategorien: $C_1, \ldots, C_K$
- Ziel: Bestimmung von $p(C_k|\mathbf{x})$ für $k = 1, \ldots, K$

Klassifizierung dann über

$$h(\mathbf{x}) = \arg \max_{k=1,\ldots,K} p(C_k|\mathbf{x})$$

Naive Bayes

- Feature-Vektor: $\mathbf{x} = (x_1, \ldots, x_n)^T$
- Kategorien: $C_1, \ldots, C_K$
- Ziel: Bestimmung von $p(C_k|\mathbf{x})$ für $k = 1, \ldots, K$

Klassifizierung dann über

$$h(\mathbf{x}) = \arg \max_{k=1,\ldots,K} p(C_k|\mathbf{x})$$

$p(C_k|\mathbf{x})$ ist oft zu groß, um es explizit zu modellieren.

Naive Bayes (fort.)

- Annahme: $x_1, \ldots, x_n$ sind voneinander unabhängig.
- Anwendung von Bayes' Theorem

$$\begin{aligned} p(C_k|\mathbf{x}) &= \frac{p(C_k)p(\mathbf{x}|C_k)}{p(\mathbf{x})} \\ &= \frac{p(C_k)\prod_{i=1}^{n} p(x_i|C_k)}{p(\mathbf{x})} \end{aligned}$$

- $p(C_k)$, $p(x_i|C_k)$ und $p(\mathbf{x})$ können aus Trainingsdaten geschätzt werden

Beispiel: Spam-Filter

- Zwei Klassen: $C_1 = \text{Spam}, C_2 = \text{Kein Spam}$
- Wörterbuch mit z.B. 5000 Wörtern
- Text repräsentiert durch $\mathbf{x} = (x_1, \ldots, x_{5000}) \in \{0, 1\}^{5000}$
- x_i gibt an, ob Wort i im Text vorkommt

Beispiel: Spam-Filter (fort.)

Trainingsbeispiele:

- $C_1 : \mathbf{t}^{(1,1)}, \ldots, \mathbf{t}^{(1,n_1)} \in \{0,1\}^{5000}$
- $C_2 : \mathbf{t}^{(1,1)}, \ldots, \mathbf{t}^{(2,n_2)} \in \{0,1\}^{5000}$

Dann:

- $p(C_k) = \frac{n_k}{n_1+n_2}, k \in \{1,2\}$
- $p(x_j = 1|C_1) = \frac{\sum_{i=1}^{n_1} 1\{t_j^{(1,i)}=1\}}{n_1}$
- $p(x_j = 1|C_2) = \frac{\sum_{i=1}^{n_2} 1\{t_j^{(2,i)}=1\}}{n_2}$

Weitere Verfahren

- Entscheidungsbäume
- Boosting (Meta-Verfahren)

Unüberwachtes Lernen

Anwendungsgebiete

Automatisches Erkennen von Zusammenhängen in Daten

- Clustering
- Principal Component Analysis
- SOM (Self-organizing maps)

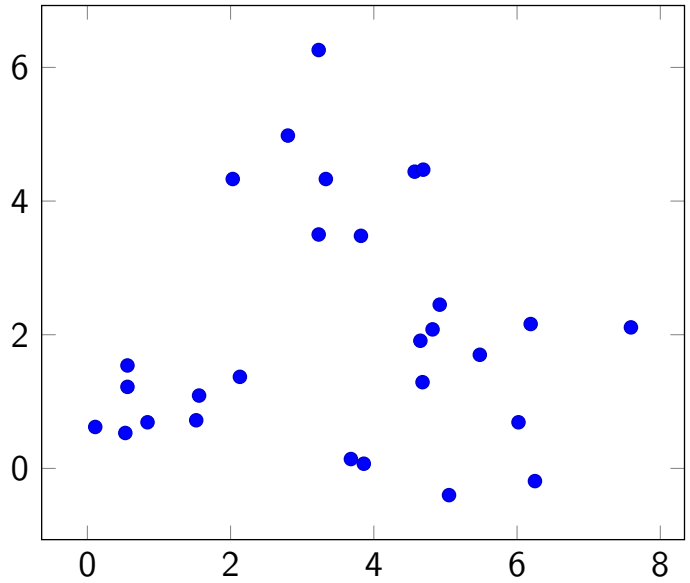
Trainingsmenge

nur *Datenpunkte*

Clustering

Ziel

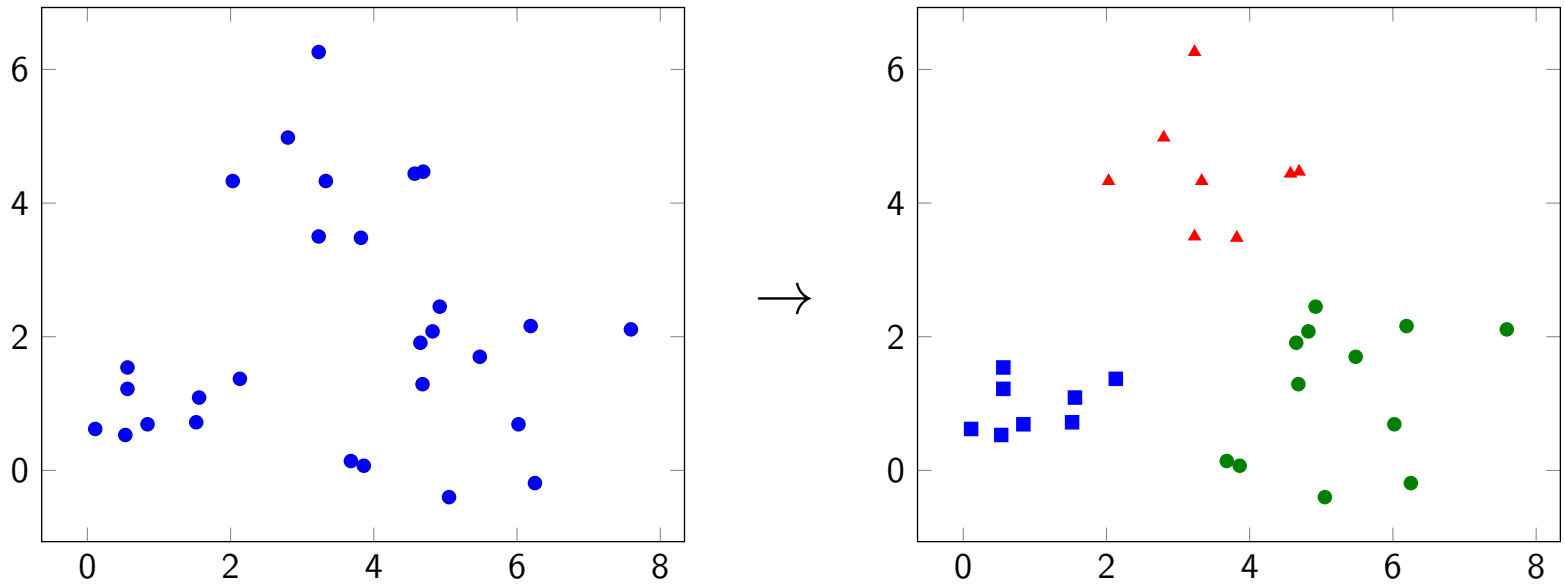
Einteilung der Daten in ähnliche Gruppen



Clustering

Ziel

Einteilung der Daten in ähnliche Gruppen



k-Means Algorithmus

Beschreibung

1. Wähle k zufällige Datenpunkte als Clusterzentren.
2. Ordne jeden Datenpunkt dem nächsten Clusterzentrum zu.
3. Wähle die Zentroide dieser Cluster als neue Clusterzentren.

wobei die Schritte 2 und 3 bis zu einem Abbruchkriterium wiederholt werden.

k-Means Algorithmus

Beschreibung

1. Wähle k zufällige Datenpunkte als Clusterzentren.
2. Ordne jeden Datenpunkt dem nächsten Clusterzentrum zu.
3. Wähle die Zentroide dieser Cluster als neue Clusterzentren.

wobei die Schritte 2 und 3 bis zu einem Abbruchkriterium wiederholt werden.

Nachteil

Anzahl der Cluster k muss bekannt sein, oder durch Ausprobieren ermittelt werden.

k-Means Konvergenz

Datenpunkte $x^{(1)}, \ldots, x^{(m)} \in \mathbb{R}^n$
Clusterzentren $\mu^{(1)}, \ldots, \mu^{(k)} \in \mathbb{R}^n$
Clusterzuordnung $c^{(1)}, \ldots, c^{(m)} \in \{1, \ldots, k\}$

k-Means konvergiert...

k-Means Konvergenz

Datenpunkte $x^{(1)}, \ldots, x^{(m)} \in \mathbb{R}^n$
Clusterzentren $\mu^{(1)}, \ldots, \mu^{(k)} \in \mathbb{R}^n$
Clusterzuordnung $c^{(1)}, \ldots, c^{(m)} \in \{1, \ldots, k\}$

k-Means konvergiert gegen lokales Minimum von

$$J(c, \mu) = \sum_{i=1}^{m} \left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$$

Globales Minimum wird nicht immer gefunden.

Weitere Clustering-Algorithmen

Agglomeratives Clustering

1. Starte mit einem Cluster pro Datenpunkt
2. Füge dicht beieinander liegende Cluster zusammen

Schritt 2 wird bis zu einem gewissen Maximalabstand wiederholt.

Dichtebasierte Algorithmen

Cluster sind Bereiche mit hoher Datendichte getrennt durch Bereiche mit niedriger Dichte

Dimensionsreduktion

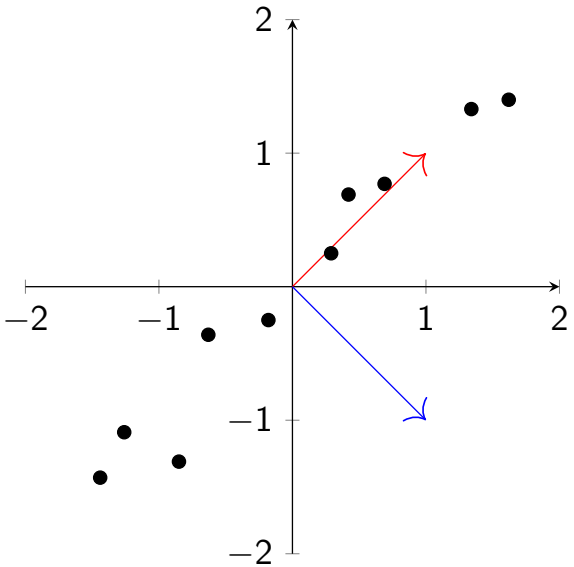
Ziel

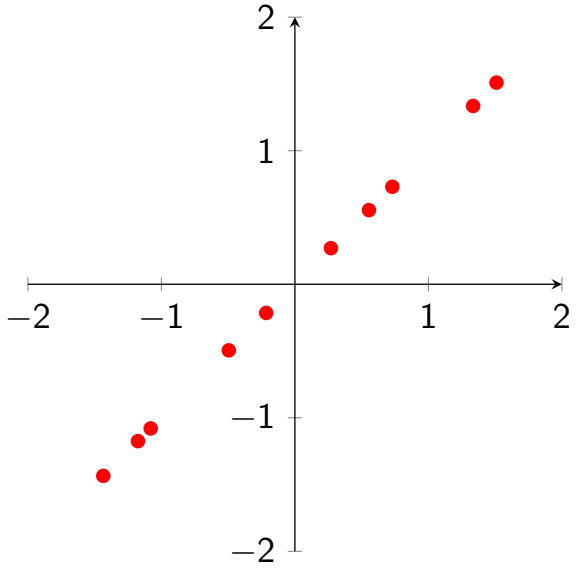
Abbildung höher-dimensionaler Daten in niedrig-dimensionale Räume ohne großen Informationsverlust.

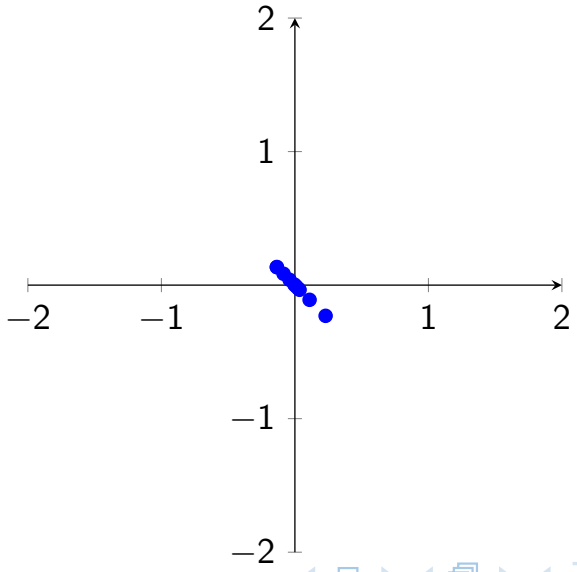
Beispiel

Voneinander abhängende Attribute:
Beschleunigung $\leftrightarrows$ *Leistung*

Principal Component Analysis (PCA)







PCA – Vorbereitung

Gegeben: Datenvektoren $x^{(1)}, \ldots, x^{(m)} \in \mathbb{R}^n$

Erwartungswert zentrieren:

1. $\mu := \frac{1}{m} \sum_{i=1}^{m} x^i$
2. $x^{(i)} \leftarrow x^{(i)} - \mu$

Varianz angleichen:

3. $\sigma_j^2 := \frac{1}{m} \sum_{i=1}^{m} \left(x_j^{(i)}\right)^2$
4. $x_j^{(i)} \leftarrow \frac{x_j^{(i)}}{\sigma_j}$

PCA - Durchführung

Empirische Kovarianzmatrix bestimmen:

5. $\Sigma := \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} \cdot x^{(i)^T})$

Σ ist symmetrisch und reell

$\Longrightarrow$ orthogonal diagonalisierbar (Hauptachsentransformation)

Ergebnis

Eigenwerte $\lambda_1 \geq \cdots \geq \lambda_n$ und
normierte Eigenvektoren $u_1, \ldots, u_n \in \mathbb{R}^n$
Je größer λ_i desto größer die Varianz entlang u_i

Self-Organizing Maps

- Abbildung hochdimensionaler Daten auf 2D oder 3D Karte.
- Ähnliche Eingaben werden auf naheliegende Punkte abgebildet
- Nutzen als Filter vor weiteren Lernverfahren

Bestärkendes Lernen

Anwendungsgebiete

Lernen durch Interaktion mit der Umgebung

Trainingsmenge

Aktionsfolge mit *Belohnungen*

Markov-Entscheidungsprozesse (MEP)

Definition

Ein MEP ist ein Tupel $(S, A, \{P_{sa}\}, \gamma, R)$

Markov-Entscheidungsprozesse (MEP)

Definition

Ein MEP ist ein Tupel $(S, A, \{P_{sa}\}, \gamma, R)$

- S ist die Zustandsmenge

Markov-Entscheidungsprozesse (MEP)

Definition

Ein MEP ist ein Tupel $(S, A, \{P_{sa}\}, \gamma, R)$

- S ist die Zustandsmenge
- A ist die Menge zulässiger Aktionen

Markov-Entscheidungsprozesse (MEP)

Definition

Ein MEP ist ein Tupel $(S, A, \{P_{sa}\}, \gamma, R)$

- S ist die Zustandsmenge
- A ist die Menge zulässiger Aktionen
- P_{sa} ist für $s \in S, a \in A$ eine Zufallsverteilung über S

Markov-Entscheidungsprozesse (MEP)

Definition

Ein MEP ist ein Tupel $(S, A, \{P_{sa}\}, \gamma, R)$

- S ist die Zustandsmenge
- A ist die Menge zulässiger Aktionen
- P_{sa} ist für $s \in S, a \in A$ eine Zufallsverteilung über S
- $\gamma \in [0, 1]$ heißt Diskontfaktor. Gibt es unendliche Zustandsfolgen, so muss $\gamma < 1$ sein.

Markov-Entscheidungsprozesse (MEP)

Definition

Ein MEP ist ein Tupel $(S, A, \{P_{sa}\}, \gamma, R)$

- S ist die Zustandsmenge
- A ist die Menge zulässiger Aktionen
- P_{sa} ist für $s \in S, a \in A$ eine Zufallsverteilung über S
- $\gamma \in [0, 1]$ heißt Diskontfaktor. Gibt es unendliche Zustandsfolgen, so muss $\gamma < 1$ sein.
- $R : S \to \mathbb{R}$ ist die Rewardfunktion ($R(s)$ ist Belohnung, wenn Zustand s erreicht wird)

MEP – Ablauf

1. Aktueller Zustand ist s_i
2. Wahl einer Aktion a_i
3. Zufälliger Übergang in einen Zustand $s_{i+1} \sim P_{s_i, a_i}$

Bildlich dargestellt:

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots$$

MEP – Ablauf

1. Aktueller Zustand ist s_i
2. Wahl einer Aktion a_i
3. Zufälliger Übergang in einen Zustand $s_{i+1} \sim P_{s_i,a_i}$

Bildlich dargestellt:

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots$$

Der gesamte Gewinn ist dann

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$$

Ziel: Wahl der Aktionen, die erwarteten Gewinn maximieren

MEP – Ansatz

- Eine Abbildung $\pi : S \rightarrow A$ heißt *Strategie*

MEP – Ansatz

- Eine Abbildung $\pi : S \to A$ heißt *Strategie*
- In einem Zustand s wird Aktion $\pi(s)$ ausgeführt

MEP – Ansatz

- Eine Abbildung $\pi : S \to A$ heißt *Strategie*
- In einem Zustand s wird Aktion $\pi(s)$ ausgeführt
- Wertfunktion einer Strategie

$$V^{\pi}(s) = \mathbb{E}\left[\sum_i \gamma^i R(s_i) \middle| s_0 = s, \pi\right]$$

MEP – Ansatz

- Eine Abbildung $\pi : S \to A$ heißt *Strategie*
- In einem Zustand s wird Aktion $\pi(s)$ ausgeführt
- Wertfunktion einer Strategie

$$V^{\pi}(s) = \mathbb{E}\left[\sum_i \gamma^i R(s_i) \middle| s_0 = s, \pi\right]$$

- V^{π} erfüllt die **Bellman Gleichung**

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')$$

Value Iteration

Algorithmus

1. Initialisiere $V_0(s) := 0$
2. Wiederhole bis Konvergenz:

$$V_{t+1}(s) := R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V_t(s')$$

Value Iteration

Algorithmus

1. Initialisiere $V_0(s) := 0$
2. Wiederhole bis Konvergenz:

$$V_{t+1}(s) := R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V_t(s')$$

Die V_t konvergieren gegen $V^*(s) = \max_\pi V^\pi(s)$
Die optimale Strategie $\pi^* : S \to A$ ergibt sich durch

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s') V_t(s')$$

und es gilt $V^*(s) = V^{\pi^*}(s)$ für alle $s \in S$

Unbekanntes Modell

Übergangsmodell $\{P_{sa}\}$ oft unbekannt

Unbekanntes Modell

Übergangsmodell $\{P_{sa}\}$ oft unbekannt

Algorithmus

1. Beginn mit zufälliger Strategie π_0
2. Wiederhole für $t = 0, 1, \ldots$
   1. Exploration des Zustandsraumes mittels π_t
   2. Aktualisierung der Schätzung von $\{P_{sa}\}$
   3. Bestimme π_{t+1} mittels Value Iteration

Zusammenfassung I

Überwachtes Lernen

In dieser Präsentation:

- Support-Vektor-Maschinen
- Naive Bayes

Außerdem:

- Perzeptron
- Neuronale Netzwerke
- Entscheidungsbäume
- Regression

Zusammenfassung II

Unüberwachtes Lernen

Clustering

- k-Means
- Agglomerative Verfahren (z.B. Single-Linkage Clustering)
- Dichtebasierte Verfahren (z.B. DBSCAN, OPTICS)

Dimensionsreduktion

- Principal Component Analysis
- Self-organizing maps

Zusammenfassung III

Bestärkendes Lernen

Markov-Entscheidungsprozesse

- Bekanntes Modell: Value-Iteration
- Unbekanntes Modell: Exploration + Schätzung + Value-Iteration

Noch Fragen?

Literatur

[1] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Bd. 4. 2006, S. 738. arXiv: 0-387-31073-8.

[2] Andrew Ng. „Machine Learning“.

[3] John C. Platt. „Fast Training of Support Vector Machines Using Sequential Minimal Optimization“. In: *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Jan. 1998.

[4] S J Russell und P Norvig. *Artificial Intelligence: A modern approach (2nd ed.* 2002.