

# 1 Preface

## 1.1 Standardimporte

## 2 Supervised Learning

*Ensemble-Lernen*: viele verschiedene Modelle werden gleichzeitig gelernt und die Resultate bei der Anwendung geeignet aggregiert. Es kann insbesondere auch dazu genutzt werden, der Überanpassung einzelner Modelle entgegenzuwirken. Dazu werden die verschiedenen Modelle nur mit einer Teilmenge der zur Verfügung stehenden Daten trainiert (Ziehen mit Zurücklegen  $\rightarrow$  *bootstrap aggregating* oder *bagging*).

### 2.1 Lineare/Polynomiale Regression

Optimale Anpassung einer Geraden an eine gegebene Menge an Punkten, d.h. für eine Funktion

$$h_{\Theta}(x) = \Theta_0 + \Theta_1 x_1 + \dots + \Theta_n x_n$$

soll der Parametervektor  $\Theta$  gefunden werden (mit  $\Theta_0$  als Konstante), der die Summe der quadrierten Abweichungen der Funktionswerte  $h_{\Theta}(x)$  von den tatsächlichen Werten  $y$  minimiert (Methode der kleinsten Quadrate):

$$\min_{\Theta} L(D, f) = \min_{\Theta} \sum_{i=1}^m (f(x^{(i)}) - y^{(i)})^2$$
$$\min_{\Theta} L(D, \Theta) = \min_{\Theta} \|X_D \Theta - y_D\|^2$$

wobei  $X_D$  eine Matrix mit den Eingabedaten (zzgl. führende 1-Spalte) und  $y_D$  der Vektor der tatsächlichen Werte ist:

$$X_D = \begin{pmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & \dots & x_n^{(m)} \end{pmatrix}, \quad y_D = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

Lokales Minimum = Globales Minimum, da die Kostenfunktion konvex ist. Lösung numerisch oder per *Gradient Descent*  $\rightarrow$  ist bei großen Trainingsdatensätzen und/oder vielen Attributen die praktikabelste Methode (s. Skript S. 13:  $\nabla_{\Theta} L(D, \Theta) = 0 \Leftrightarrow (X_D^T X_D)^{-1} X_D^T y_D = \Theta$ , wobei inverse von  $X_D^T X_D$  sehr rechenaufwändig ist).

Erweiterung auf Polynome höheren Grades durch (Kreuz-)Multiplikation bestehender Merkmale  $\rightarrow$  Modell ist linear bzgl. des erweiterten Merkmalsraums und erscheint polynominal bei Projektion auf den ursprünglichen Merkmalsraum.

Evaluation mittels **Bestimmtheitsmaß** (=normalisierte Variante des quadratischen Fehlers):

$$R^2(D, f) = 1 - \frac{\sum_{i=1}^m (f(x^{(i)}) - y^{(i)})^2}{\sum_{i=1}^m (y^{(i)} - \bar{y})^2}$$

mit  $\bar{y} = \frac{1}{m} \sum_{i=1}^m y^{(i)}$ , wobei in der Praxis der Durchschnitt mehrerer  $R^2$  berechnet wird (*Kreuzvalidierung*).

- $R^2(D, f)$  ist maximal 1  $\rightarrow f$  modelliert  $D$  perfekt
- $R^2(D, f) = 0 \rightarrow$  naives Modell,  $f$  sagt stets den Mittelwert  $\bar{y}$  voraus
- $R^2(D, f) < 0 \rightarrow$  Modell schlechter als naives Modell
- $R^2(D^{\text{train}}, f)$  sollte relativ nahe an 1 liegen
- $R^2(D^{\text{test}}, f)$  ist üblicherweise kleiner als  $R^2(D^{\text{train}}, f)$
- Je näher  $R^2(D^{\text{test}}, f)$  an  $R^2(D^{\text{train}}, f)$ , desto besser ist das Modell generalisiert

**Überanpassung**: Modell passt sich zu stark an Trainingsdaten an, d.h. es wird zu komplex modelliert. Dies führt zu schlechterer Generalisierung auf Testdaten  $\rightarrow$  *Varianzfehler*

**Unteranpassung**: Modell ist nicht ausdrucksstark genug; Trainings- und Testdaten werden unzureichend modelliert  $\rightarrow$  *Verzerrungsfehler*

Ermittlung der **optimalen Modellkomplexität** durch Betrachtung der Kostenfunktionswerte oder Bestimmtheitsmaße bei steigender Komplexität:

- Trainingsdaten: Je komplexer das Modell, desto höher die Bestimmtheit
- Testdaten
  - Bestimmtheit nimmt zunächst ebenfalls zu (das Modell ist noch unterangepasst)
  - Ab einem gewissen Punkt nimmt die Bestimmtheit ab: Das Modell ist überangepasst
- Optimaler Punkt: Modellkomplexität, bei der die Bestimmtheit bzgl. der Testdaten maximal ist

Automatische Lösung des *Verzerrungs-Varianz-Dilemmas* durch **Regularisierung**: Hinzufügen eines mit  $\lambda$  (*Regularisierungsparameter*) gewichteten Strafterms (*Tikhonov-Regularisierer*  $R_T$ ) zur Kostenfunktion, der die Größe der Parametervektoren begrenzt. Sog. *Ridge-Regression*:

$$L_T(D, \Theta) = \|X_D \Theta - y_D\|^2 + \lambda \sum_{i=1}^n \Theta_i^2$$

- Regularisierer wird ohne  $\Theta_0$  berechnet
- Je mehr  $\Theta_i \neq 0$ , desto größer wird der Tikhonov-Regularisierer  $\rightarrow$  Kosten steigend
- Einbeziehung von  $\lambda \sum_{i=1}^n \Theta_i^2$  erzwingt Fokussierung auf möglichst einfache Funktionen
- Kleines  $\lambda \rightarrow$  Überanpassung, großes  $\lambda \rightarrow$  Unteranpassung

## 2.2 Logistische Regression

Diskreter Wertebereich der Zielvariablen  $y^{(i)}$ , idR. endlich und oft auch nur binär ( $y^{(i)} \in \{0, 1\}$ ). Klassen haben idR. keine (eindeutige) Ordnung.

Einsetzen eines linearen Modells  $h_\Theta$  in eine Funktion  $g(z) = \frac{1}{1+e^{-z}}$  mit dem Zielbereich  $(0, 1)$  ergibt sog. *Sigmoid-Funktion*:

$$h_\Theta^{\text{logit}}(x) = \frac{1}{1 + e^{-(\Theta_0 + \Theta_1 x_1 + \dots + \Theta_n x_n)}}$$

Klassifikation durch Schwellwert 0.5:

$$\text{clf}_f(x) = \begin{cases} 1 & \text{falls } h_\Theta^{\text{logit}}(x) \geq 0.5 \\ 0 & \text{falls } h_\Theta^{\text{logit}}(x) < 0.5 \end{cases}$$

bzw. bei  $n$  Klassen diejenige Klasse  $k$ , für die  $h_\Theta^{\text{logit}}(x)_k$  maximal ist. Bewertung mittels der *logistischen Kostenfunktion*  $L^{\text{logit}}$ :

$$L^{\text{logit}}(D, f) = - \sum_{i=1}^m \left[ \underbrace{y^{(i)} \ln(f(x^{(i)}))}_a + \underbrace{(1 - y^{(i)}) \ln(1 - f(x^{(i)}))}_b \right]$$

Bei  $y = 1$  wird  $b = 0$ , bei  $y = 0$  wird  $a = 0$  und es ergibt sich:

$f(x)$	$y$	$L^{\text{logit}}(D, f)$
1	1	0
1	0	$\infty$
0	0	0
0	1	$\infty$

**Ziel:** Minimierung der Kostenfunktion, wobei es sich um ein konvexes Optimierungsproblem handelt (d.h. es existiert nur ein globales Minimum). Effiziente Lösung mithilfe numerischer Methoden wie *Gradient Descent* möglich.

**Polynomiale Erweiterung** sowie **Regularisierung** analog zur *linearen Regression*.

**Evaluation:** Berechnung der *Konfusionsmatrix* und Einsetzen in die *Klassifikationsmetriken*:

	$y = 1$	$y = 0$
$\text{clf} = 1$	TP	FP
$\text{clf} = 0$	FN	TN

- *Accuracy/Genauigkeit*:  $\text{acc}(D, \text{clf}) = \frac{TP+TN}{TP+TN+FP+FN} \rightarrow$  Verhältnis der korrekt klassifizierten Instanzen zu allen Instanzen; bei ungleicher Klassenrepräsentation nicht geeignet
- *Precision*:  $\text{prec}(D, \text{clf}) = \frac{TP}{TP+FP} \rightarrow$  wie viele der positiv klassifizierten Instanzen sind tatsächlich positiv?
- *Recall/Sensitivität*:  $\text{rec}(D, \text{clf}) = \frac{TP}{TP+FN} \rightarrow$  wie viele Ist-positive Instanzen wurden korrekt klassifiziert?
- *F1-Score*:  $F1(D, \text{clf}) = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \rightarrow$  harmonisches Mittel von Precision und Recall (Gesamtqualität)

## 2.3 Support Vector Machines

Test

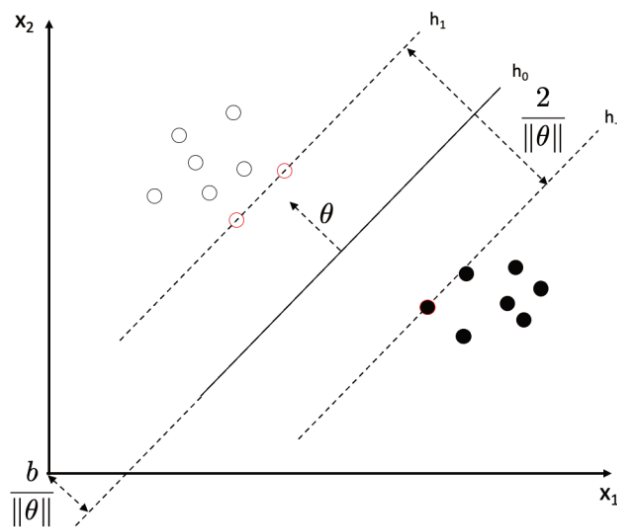


Abbildung 1: Support Vector Machines

## 2.4 K-Nearest Neighbours

Grundidee: Klassifizierung eines Objekts anhand der Klassenzugehörigkeit seiner  $k$  nächsten Nachbarn (aus  $X_{\text{Train}}$ ). Bei mehr als  $k$  nächsten Nachbarn wird eine zufällige Auswahl der möglichen Kandidaten gewählt. Wenn Klassenzugehörigkeit nicht eindeutig ist (z.B.  $k = 3$  und alle Elemente haben unterschiedliche Klassen) wird eine zufällige gewählt. Parametrisierung:

- Abstandsmessung z.B. per *euklidischer Norm*  $\|\cdot\| = \sqrt{(x_1^i - x_1^j)^2 + \dots + (x_n^i - x_n^j)^2}$  oder *Manhattan-Norm*  $= \sum_{l=1}^n |x_l^i - x_l^j|$  (o.a.).
- Selektion der Klasse anhand der Mehrheitsentscheidung der  $k$  nächsten Nachbarn (*maj*) oder anderen (z.B. mit Gewichtung).
- Typische Werte für  $k$  liegen im Bereich 1 bis 10, wobei kleinere  $k$  zu *Überanpassung* und größere  $k$  zu *Unteranpassung* neigen.

Regression: Statt Klassenzugehörigkeit wird der Mittelwert der  $k$  nächsten Nachbarn als Schätzung für den Wert des Objekts verwendet.

Vor- und Nachteile:

- + Einfach und intuitiv
- + Keine Annahmen über die Verteilung der Daten
- Langsam bei großen Trainingsdatensätzen
- Sensibel gegenüber Ausreißern
- Wahl von  $k$  und Abstandsmessung nicht trivial

**Merkmalskalierung:** Wichtig bei den meisten ML-Verfahren, da sonst Merkmale mit größeren Werten (Skalen) stärker gewichtet werden (insb. bei Verwendung der Euklidischen Norm). Wichtiger Schritt in der *Datenvorverarbeitung*: Es geht darum Merkmalsausprägungen zu *normieren*. Gebräuchlichste Variante: *z-Transformation* (bzgl. *Standardisierung*):

$$\text{normiertes Merkmal} \rightarrow \hat{x}_j^{(i)} = \frac{x_j^{(i)} - \bar{x}_j}{\sigma_j}$$

$$\text{Mittelwert} \rightarrow \bar{x}_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$$

$$\text{Standardabweichung} \rightarrow \sigma_j = \sqrt{\frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \bar{x}_j)^2}$$

→ Merkmale erhalten eine mittlere Ausprägung von 0 und eine Standardabweichung von 1.

## 2.5 Bayes-Klassifikator

Optimalität: Gegeben einem beobachteten Datensatz  $D$ , suche die wahrscheinlichste Hypothese  $h$ , die den Wert  $P(h|D)$  maximiert.

- **Bayes Theorem:**  $P(h|D) = \frac{P(D|h)P(h)}{P(D)}$
- **a posteriori** Wkt. von  $h$ :  $P(h|D) \rightarrow$  kann im allg. nicht direkt bestimmt werden
- **a priori** Wkt von  $h$ :  $P(h) \rightarrow$  wie wahrscheinlich ist eine Hypothese  $\rightarrow$  Hintergrundwissen oder Annahme von Gleichverteilung
- $P(D) \rightarrow$  Wkt, dass  $D$  beobachtet wurde
- $P(D|h) \rightarrow$  Wkt,  $D$  zu beobachten, gegeben dass  $D$  von  $h$  *generiert* wurde  $\rightarrow$  wie gut erklärt  $h$  den Datensatz  $D$ ? Ist einfacher abzuschätzen als  $P(h|D)$
- **Maximum-A-Posteriori (MAP)**-Hypothese  $\rightarrow h^* = \operatorname{argmax}_{h \in H} P(h|D)$
- **Maximum-Likelihood (ML)**-Hypothese  $\rightarrow h^* = \operatorname{argmax}_{h \in H} P(D|h)$

Es lässt sich zeigen, dass das Problem der **(linearen) Regression** mit  $\Theta^* = \arg \min_{\Theta} \sum_{i=1}^m (\Theta^T x^{(i)} - y^{(i)})^2$  (d.h. Nutzung der quadratischen Kostenfunktion) eine ML-Hypothese bestimmt, die lineare Regression also nach Bayes'schen Grundsätzen plausibel ist. Dabei wird angenommen, dass die Werte  $y^{(i)}$  eines Datensatzes  $D$  durch den wahren Wert der Merkmalsausprägung  $\hat{y}^{(i)}$  und einen Fehler  $\epsilon^{(i)}$  bestimmt sind. Bzgl.  $\epsilon^{(i)}$  wird üblicherweise Normalverteilung angenommen (Herleitung: Skript S. 59).

**Nachteile** des (normalen) Bayes Klassifikators:

- jede mögliche Merkmalsausprägung (Kombination aus mehreren Merkmalen) muss im Trainingsdatensatz vorhanden sein, da Wahrscheinlichkeit sonst undefiniert ( $\frac{0}{0}$ ) ist
- der gesamte Datensatz  $D$  muss vorgehalten werden  $\rightarrow$  Höherer Speicherbedarf und längere Berechnungszeiten

Diese Nachteile lassen sich mittels **Naivem Bayes Klassifikators** lösen. Dieser geht davon aus, dass die einzelnen Merkmalsausprägungen bedingt unabhängig voneinander sind:

$$\text{clf}_D^{\text{NaiveBayes}}(x) = \arg \max_{c \in Z} P(c|D)P(x_1|c, D)P(x_n|c, D) \dots P(x_n|c, D)$$

- $P(c|D) \rightarrow$  Für jede Klasse  $c$ : Wie oft kommt diese in  $D$  vor (Wkt)?
- $P(x_i|c, D) \rightarrow$  Für eine gegebene Klasse  $c$ : Einzelwkt. für  $x_i = \dots$

Verwendung *kontinuierlicher Merkmale* möglich, indem die direkt aus den Daten berechnete Verteilung  $P(x_i|c, D)$  durch eine abgeschätzte Dichte  $p(x_i|c, D) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$  ersetzt wird.

## 2.6 Entscheidungsbäume

Eigenschaften:

- Intuitiv und regelbasiert nachvollziehbar
- *Entscheidungsbaum*  $T$  mit Blattknoten (auch: *Klassifikationsknoten*), inneren Knoten (auch: *Entscheidungsknoten*) und Kanten (E) stellt einen (abwärts) gerichteten Baum mit Wurzel  $r$  dar.
- Jeder Knoten stellt eine Fallunterscheidung für den Wert eines Merkmals eines gegebenen Datenpunktes dar.
- Für jeden Fall gibt es einen eindeutigen Nachfolgeknoten und die Blätter des Baumes enthalten die möglichen Klassifikationen des betrachteten Datenpunktes.

Lösung üblicherweise nicht durch Lösen eines Optimierungsproblems sondern durch dedizierte prozedurale Algorithmen wie **ID3** (Iterative Dichotomiser 3) oder **C4.5**:

- Basieren auf dem Prinzip der *top-down induction of decision trees* (TDIDT)  $\rightarrow$  baut Entscheidungsbaum rekursiv von der Wurzel beginnend auf
- Abbruchbedingungen  $\rightarrow$  alle Beispiele in der betrachteten (Teil-)Menge
  - sind von derselben Klasse
  - haben identische Merkmalsausprägungen  $\rightarrow$  wähle am häufigsten vorkommende Klasse
- *C4.5-Algorithmus* ist eine Weiterentwicklung von ID3, enthält jedoch folgende Optimierungen:
  - ID3 tendiert bei leicht verrauschten Daten zur Überanpassung (d.h. sehr lange Pfade); C4.5 enthält einen Nachverarbeitungsschritt, der weniger relevante Entscheidungsregeln erkennt und den Baum kürzt (sog. *pruning*)
  - bessere Behandlung kontinuierlicher Merkmale
  - optimierte (skalierte) Version des *Informationsgewinns* (s.u.)  $\rightarrow$  unten dargestellte Version bevorzugt Merkmale mit einer hohen Anzahl an Merkmalsausprägungen

Merkmal, nach dem die jeweils nächste Aufteilung erfolgt soll so gewählt werden, dass der Baum möglichst klein wird  $\rightarrow$  wähle Merkmal mit höchstem Informationsgewinn (IG) unter Verwendung der *Entropie*.

- **Entropie:**  $H(D) = -\sum_i^k \text{relative Häufigkeit}_i \times \log_2(\text{relative Häufigkeit}_i) \rightarrow$  je höher  $H(D)$ , desto gleichverteilter treten die Klassen auf (z.B. = 0, wenn alle Element einer einzelnen Klasse angehören; dann steht Klassifikation fest).
- **bedingte Entropie:** Was passiert mit der Entropie, wenn ein bestimmtes Merkmal ausgewählt wird?  $H(D, i) = \sum \text{rel. Häufigkeit der Merkmalsausprägung}_i \times \text{Entropie der Teilmenge}_i$
- **Informationsgewinn:**  $IG(D, i) = H(D) - H(D|i) \rightarrow$  gibt an, wie sehr die Merkmalsauswahl  $i$  die Daten in  $D$  nach den Klassen (vor-)sortiert. Je höher der Informationsgewinn, desto besser klassifiziert das Merkmal  $i$  den Datensatz  $D$

## **3 Unsupervised Learning**

### **3.1 K-Means Clustering**

Test

### **3.2 Hierarchisches Clustering**

Test

### **3.3 Assoziationsregeln**

Test

### **3.4 Anomalieerkennung**

Test

### **3.5 Hauptkomponentenanalyse/Principal Component Analysis (PCA)**

Test

## **4 Reinforcement Learning**

### **4.1 Markov-Entscheidungsprozesse**

Test

### **4.2 Passives Reinforcement-Learning**

Test

### **4.3 Aktives Reinforcement-Learning**

Test



## **5 Deep Learning**

### **5.1 Künstliche Neuronale Netze**

Test

### **5.2 Convolutional Neutral Networks**

Test

### **5.3 Recurrent Neutral Networks**

Test

### **5.4 Recurrent Neutral Networks**

Test