1 Preface

1.1 Standardimporte

2 Supervised Learning

2.1 Lineare/Polynominelle Regression

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

$$h_\Theta(x) = \Theta_0 + \Theta_1 x_1 + \cdots + \Theta_n x_n$$

soll der Parametervektor Θ gefunden werden (mit Θ_0 als Konstante), der die Summe der quadrierten Abweichungen der Funktionswerte $h_\Theta(x)$ von den tatsächlichen Werten y minimiert (Methode der kleinsten Quadrate):

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

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

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

Lokales Minimum = Globales Minimum, da die Kostenfunktion konvex ist. Lösung numerisch oder per *Gradient Descent* → 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 → Modell ist linear bzgl. des erweiterten Merkmalsraums und erscheint polynominell 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 → *Varianzfehler*

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

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

- Trainingsdten: 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 λ (*Regularisierungsparamter*) gewichteten Strafterms (*Tikhonov-Regularisierer* R_T) zur Kostenfunktion, der die Größe der Parametervektoren begrenzt. Sog. *Ridge-Regression*:

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

- Regularisierer wird ohne Θ_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_Θ in eine Funktion $g(z) = \frac{1}{1+e^{-z}}$ mit dem Zielbereich $(0, 1)$ ergibt sog. *Sigmoid-Funktion*:

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

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^{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	∞
0	0	0
0	1	∞

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

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

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

	$y = 1$	$y = 0$
clf $= 1$	TP	FP
clf $= 0$	FN	TN

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

2.3 Support Vector Machines

Test

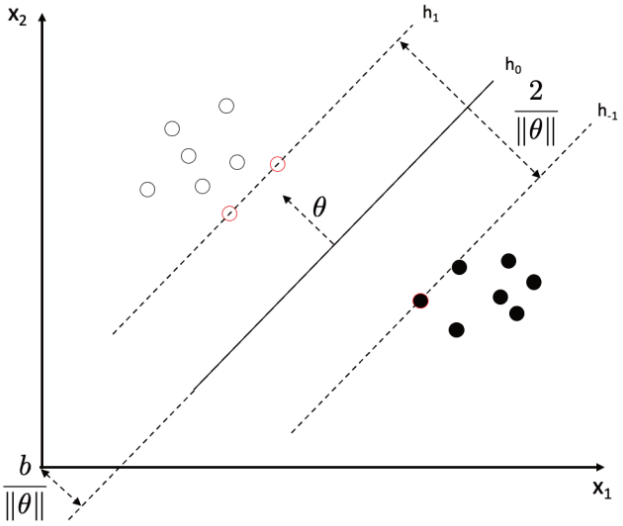

Abbildung 1: Support Vector Machines

2.4 K-Nearest Neighbours

Grundidee: Klassifizierung eines Objekts anhand der Klassenzugehörigkeit seiner k nächsten Nachbarn (aus X_{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* $\|\cdot\| = \sqrt{(x_1^i - x_1^j)^2 + \cdots + (x_n^i - x_n^j)^2}$ 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 Trainingsdatenmengen

- Sensibel gegenüber Ausreißern
- Wahl von k und Abstandsmessung nicht trivial

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

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

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

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

$\rightarrow$ Merkmale erhalten eine mittlere Ausprägung von 0 und eine Standardabweichung von 1.

2.5 Bayes-Klassifikator

Test

2.6 Entscheidungsbäume

Test

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