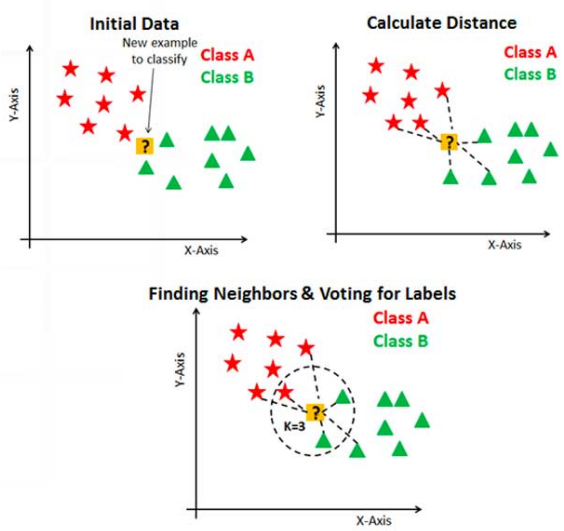
Kapitel 7

Instanzbasiertes Lernen



Inhalt

Einleitung

- Prognosen basieren auf der Annahme, dass ein funktionaler Zusammenhang zwischen der zu prognistizierenden Größe y und anderen d-dimensionalen Variablen $\mathbf{x} = (x_1, \ldots, x_d)^T \in \mathbb{R}^d$ besteht.
- Kann der funktionale Zusammenhang geschätzt werden, so kann im Prinzip für jedes $\mathbf{x}$ der zugehörige Wert y prognostiziert werden.
- Bei den meisten Anwendungen wird angenommen, dass der funktionale Zusammenhang einem niedrigdimensionalen parametrischen Modell entspricht oder durch solch einen Ansatz gut wiedergegeben wird.

  ☞ Kapitel 3
- Problem: Der wirkliche funktionale Zusammenhang entspricht u. U. nicht dem gewählten Modell.
- Folge: schlechte Prognosen

Einleitung (2)

- Nichtparametrische Verfahren gehen ebenfalls von einem funktionalen Zusammenhang aus,
- geben aber kein festes parametrisches Modell vor.
- Grundidee: Finde für ein gegebenes $\mathbf{x}$ lokal eine beste Anpassung an die vorliegenden Daten und prognostiziere damit y.
- Vorteil: Wir benötigen keine Trainingsphase mehr, da keine Parameter eines Modells vor der Prognose an die Trainingsbeispiele angepasst werden müssen.

  Die Prognose findet sofort mithilfe der abgespeicherten Trainingsbeispiele statt.

  ☞ Instanzbasiertes Lernen, Lazy Learning
- Nachteil: Wir erhalten keinen einfachen globalen funktionalen Zusammenhang.

Gewichtete Regression

- Situation wie in Kapitel 3 bei der Ausgleichsrechung: Wir wollen eine Funktion $f(x; \lambda_1, \ldots, \lambda_m)$ möglichst gut an einen Datensatz

$$(x_1, y_1), \ldots, (x_n, y_n)$$

anpassen.

- Die Funktion f ist eine Linearkombination von Ansatzfunktionen, also

$$f(x; \lambda_1, \ldots, \lambda_m) = \sum_{j=1}^{m} \lambda_i f_j(x).$$

- Aber: Für die Anpassung sollen die Abweichungen nicht (wie in Kapitel 3) mit gleichen Anteilen eingehen.

- Vielmehr wird jede Abweichung mit einem Gewicht $w_i \geq 0$, $i = 1, \ldots, n$ gewichtet.
- Damit betrachten wir das Fehlerfunktional

$$E(\lambda_1, \ldots, \lambda_m) = \sum_{i=1}^{n} w_i (y_i - f(x_i; \lambda_1, \ldots, \lambda_m))^2.$$

- Um die λ_i zu bestimmen, bilden wir wie üblich die partiellen Ableitungen und setzen diese auf Null:

$$\frac{\partial E(\lambda_1, \ldots, \lambda_m)}{\partial \lambda_i} \stackrel{!}{=} 0.$$

- Wie sieht das zugehörige Normalgleichungssystem aus?

Lineare gewichtete Regression

Zur Minimierung von

$$E(a,b) = \sum_{i=1}^{n} w_i (y_i - (ax_i + b))^2$$

müssen die partiellen Ableitungen gleich 0 sein.

$$\begin{aligned} \frac{\partial E(a,b)}{\partial a} &= -2 \sum_{i=1}^{n} w_i ((y_i - (ax_i + b)) x_i \stackrel{!}{=} 0 \\ \frac{\partial E(a,b)}{\partial b} &= -2 \sum_{i=1}^{n} w_i ((y_i - (ax_i + b)) \stackrel{!}{=} 0 \end{aligned}$$

Umformung ergibt das LGS

$$\begin{aligned} a\sum_{i=1}^{n} w_i x_i^2 \quad + \quad b\sum_{i=1}^{n} w_i x_i \quad &= \quad \sum_{i=1}^{n} w_i x_i y_i \\ a\sum_{i=1}^{n} w_i x_i \quad + \quad b\sum_{i=1}^{n} w_i \quad &= \quad \sum_{i=1}^{n} w_i y_i \end{aligned}$$

bzw.

$$\begin{pmatrix} \sum_{i=1}^{n} w_i x_i^2 & \sum_{i=1}^{n} w_i x_i \\ \sum_{i=1}^{n} w_i x_i & \sum_{i=1}^{n} w_i \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} w_i x_i y_i \\ \sum_{i=1}^{n} w_i y_i \end{pmatrix}.$$

Normalgleichungssystem für die gewichtete Regression

Satz 7.1

Das Normgleichungssystem für das gewichtete Regressionsproblem lautet:

$$\mathbf{A}^T\mathbf{W}\mathbf{A}\boldsymbol{\lambda} = \mathbf{A}^T\mathbf{W}\mathbf{y}.$$

Dabei ist $\mathbf{W}$ *eine* $n \times n$*-Diagonalmatrix mit den Gewichten* w_i *auf der Diagonalen.*

Lokale gewichtete Regression

Gegeben: n Beobachtungen $(x_1, y_1), \ldots, (x_n, y_n)$ für die wir einen funktionalen Zusammenhang $f(x) = y$ vermuten.

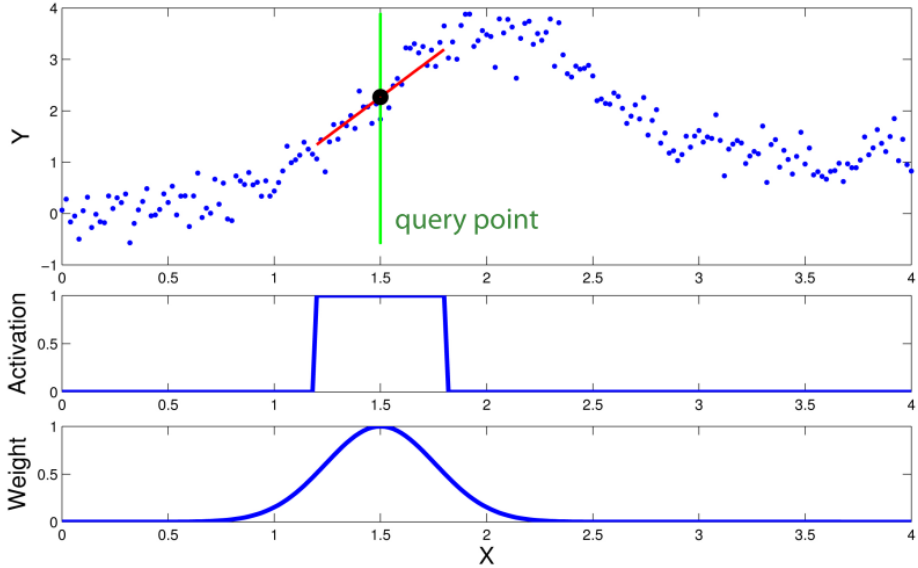
Die Funktion $f(x)$ ist uns nicht bekannt.

Gewünscht: Für einen Anfragewert $\hat{x}$ hätten wir gerne eine Prognose $\hat{y} = f(\hat{x})$.

Prinzip: Wir führen eine (lineare) Regression durch, bei der wir die x_i in Abhängigkeit von $|\hat{x} - x_i|$ gewichten.

Es sei $f_{\hat{x}}(x)$ die zugehörige Regressionfunktion.

Dann prognostizieren wir $\hat{y} = f_{\hat{x}}(\hat{x})$.


Y
query point
Activation
Weight
X

Gewichtung und Auswahl

Fragen:

1. Wie sollen wir die x_i in Abhängigkeit von $\hat{x}$ gewichten?
   ☞ Weight
2. Welche x_i sollen überhaupt für die Prognose genutzt werden?
   ☞ Activation

Zu 1.:

- Zunächst berechnet man den Abstand $d(x_i, \hat{x})$.
- Diesen Abstand setzt man dann in eine Gewichtsfunktion $K(\cdot)$, auch Kernfunktion genannt, ein.

Zu 2.:

- Üblicherweise berücksichtigt man nur Beobachtungen (x_i, y_i) für die x_i nicht zu weit von $\hat{x}$ entfernt ist.
- Dies drückt man durch eine Indikatorfunktion $I(\cdot)$ aus, z. B.

$$I_{\hat{x}}(x) = \begin{cases} 1 & \text{für } |x - \hat{x}| \leq 1 \\ 0 & \text{sonst} \end{cases}$$

Die Gewichtung erfolgt dann als Kombination von Kern- und Indikatorfunktion:

$$w_i = K(d(x_i, \hat{x})) \cdot I_{\hat{x}}(x_i).$$

Typische Kernfunktionen

- Gaußkern: $K(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}t^2\right)$
- Cauchy-Kern: $K(t) = \frac{1}{\pi(1+t^2)}$
- Picard-Kern: $K(t) = \frac{1}{2}\exp(-|t|)$

Für eine Aussage $B(t)$ sei die Indikatorfunktion $I(B(t))$ definiert durch

$$I(B(t)) = \begin{cases} 1 & \text{falls } B(t) \text{ wahr ist} \\ 0 & \text{sonst.} \end{cases}$$

- Epanechnikov-Kern: $K(t) = \frac{3}{4}(1-t^2)I(|t| \leq 1)$.
- Tricube-Kern: $K(t) = \frac{70}{81}\left(1-|t|^3\right)^3 I(|t| \leq 1)$.

Diskussion: Kernfunktionen

- Für t ist der Abstand $d(x_i, \hat{x})$ einzusetzen.
- Die Kernfunktionen sind hier als Wahrscheinlichkeitsdichten angegeben (Probability Density Function, PDF).
- Wenn wir die Gewichte aber ausschließlich für eine gewichtete Regression nutzen, spielen die konstanten Faktoren keine Rolle.
- Beim Gauß-, Cauchy- und Picard-Kern können prinzipiell alle Beobachtungen in das Ergebnis einfließen.
- Wir können diese Kernfunktionen aber auch mit einer zusätzlichen Indikatorfunktion kombinieren.
- Beim Epanechnikov- und Tricube-Kern fließen nur die Beobachtungen ein, die von $\hat{x}$ einen Abstand ≤ 1 haben.
- Dieser erlaubte Abstand kann natürlich angepasst werden.

Diskussion: Lokale gewichtete Regression

- Bei reiner linearer Regression (lokal) werden Spitzen in den Daten evtl. nicht gut erkannt.
- Lokale gewichtete Regression ist aber auch mit Polynom-Modellen möglich. Damit ist eine bessere Anpassung an Spitzen möglich.
- Wichtig: Es sollten genügend Daten vorhanden sein.
- Kern- und Indikatorfunktion müssen gut eingestellt sein, um Über- und Unteranpassung zu vermeiden.
- Mit einer zusätzlichen gewichteten zeitlichen Komponente kann man auch Änderungen in den Daten lernen.

kNN-Verfahren

- kNN-Verfahren können sowohl zur Regression als auch zur Klassifikation eingesetzt werden.
- Sie basieren auf den Ideen der lokal gewichteten Regression.
- Zu einem Anfragepunkt werden die k nächsten Nachbarn genutzt, um einen Funktionswert zu prognostizieren (Regression) oder das Objekt einer Klasse zuzuordnen (Klassifikation).
- $k = 1$ führt meist zu einer starken Überanpassung, es sollte ein größerer Wert für k gewählt werden.
- Meistens gehen die k nächsten Nachbarn gewichtet in das Ergebnis ein. Dann können für die Gewichtung die bekannten Kernfunktionen genutzt werden.

Regression mit kNN

Gegeben:

- Beobachtungen $(\mathbf{x}^{(1)}, y_1), \ldots, (\mathbf{x}^{(n)}, y_n)$. mit $\mathbf{x}^{(i)} \in \mathbb{R}^d$ und $y_i \in \mathbb{R}$.
- Ein Anfragepunkt $\hat{\mathbf{x}} \in \mathbb{R}^d$.

Gesucht: Ein prognostizierter Funktionswert $\hat{y} \in \mathbb{R}$ für $\hat{\mathbf{x}}$.

1. Bestimme die k nächsten Nachbarn $\mathbf{x}^{(j_1)}, \ldots, \mathbf{x}^{(j_k)}$ des Anfragepunkts $\hat{\mathbf{x}}$.
2. Ordne jedem der k nächsten Nachbarn $\mathbf{x}^{(j_i)}$ ein Gewicht w_i zu, abhängig von der Entfernung zwischen $\mathbf{x}^{(j_i)}$ und $\hat{\mathbf{x}}$.
3. Prognostiziere

$$\hat{y} = \frac{\sum_{i=1}^{k} w_i y_{j_i}}{\sum_{i=1}^{k} w_i}$$

als Funktionswert für $\hat{x}$.

Klassifikation mit kNN

Gegeben:

- Eine Menge $\mathcal{K} = \{K_1, \ldots, K_r\}$ von Klassen.
- Beobachtungen $(\mathbf{x}^{(1)}, k_1), \ldots, (\mathbf{x}^{(n)}, k_n)$ mit $\mathbf{x}^{(i)} \in \mathbb{R}^d$ und $k_i \in \mathcal{K}$.
- Ein Anfragepunkt $\hat{\mathbf{x}} \in \mathbb{R}^d$.

Gesucht: Eine Klassenzuordnung $\hat{k} \in \mathcal{K}$ für $\hat{\mathbf{x}}$.

1. Bestimme die k nächsten Nachbarn $\mathbf{x}^{(j_1)}, \ldots, \mathbf{x}^{(j_k)}$ des Anfragepunkts $\hat{\mathbf{x}}$.
2. Ordne jedem der k nächsten Nachbarn $\mathbf{x}^{(j_i)}$ ein Gewicht w_i zu, abhängig von der Entfernung zwischen $\mathbf{x}^{(j_i)}$ und $\hat{\mathbf{x}}$.
3. $$\hat{k} = \text{argmax}_{r \in \mathcal{K}} \frac{\sum_{i=1}^{k} w_i \cdot I(\mathbf{x}^{(j_i)} \text{ hat Klasse } r)}{\sum_{i=1}^{k} w_i}$$

Suche der k nächsten Nachbarn

- Problem: Suche in einem evtl. hochdimensionalem Raum
- Unterscheidung zwischen Vektorraum und metrischem Raum
- Vektorraum: Die Beobachtungen sind Punkte im $\mathbb{R}^d$.

  Uns stehen alle Operationen eines Vektorraums zur Verfügung, wir können z. B. Mittelpunkte bilden oder die Punktemenge mit einer Hyperebene aufteilen.
- Spezielle Datenstrukturen für die Suche: k-d-Baum, R-Baum
- Metrischer Raum: Wir haben nur die Objekte und eine Abstandsfunktion, aber keine Vektorraumoperationen.
- Es gibt spezielle Datenstrukturen für die Suche in metrischen Räumen, z. B. M-Tree.
- Hauptproblem: Der Datenraum kann i. d. R. nicht disjunkt aufgeteilt werden.
- Die Effizienz der Suche ist dadurch (teilweise deutlich) geringer als im eindimensionalen Fall (B-Baum, AVL-Baum).

Zusammenfassung

- Bei instanzbasiertem Lernen wird kein globales Modell gelernt,
- sondern zu einem Anfragepunkt erfolgt eine lokale Anpassung an die vorliegenden Beobachtungen.
- Anpassung mit linearer Funktion oder einem Polynom mit kleinem Grad
- Gewichtung der Beobachtungen mit Kernfunktionen
- einsetzbar für Regressions- und Klassifikationsprobleme