Maschinelles Lernen 05

Prof. Dr. Christoph Böhm

Hochschule München

3. Januar 2024

K-Nearest Neighbors

K-Nearest Neighbors

K-Nearest Neighbors

Einführung

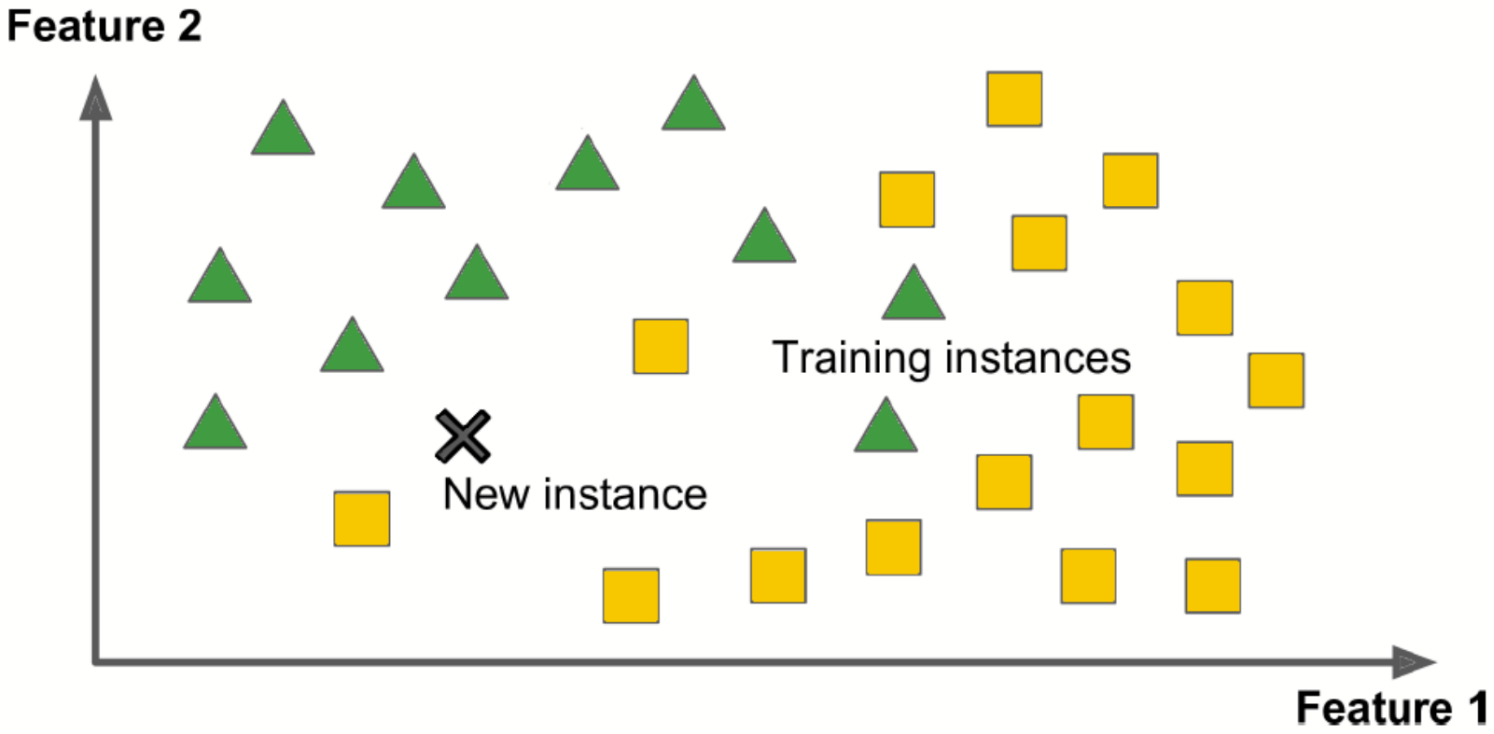


Abbildung 1: Beispiel für eine Klassifikation mit zwei Klassen.

(Quelle: `https://mlarchive.com/machine-learning/k-nearest-neighbor-knn-explained/`

K-Nearest Neighbors

Einführung

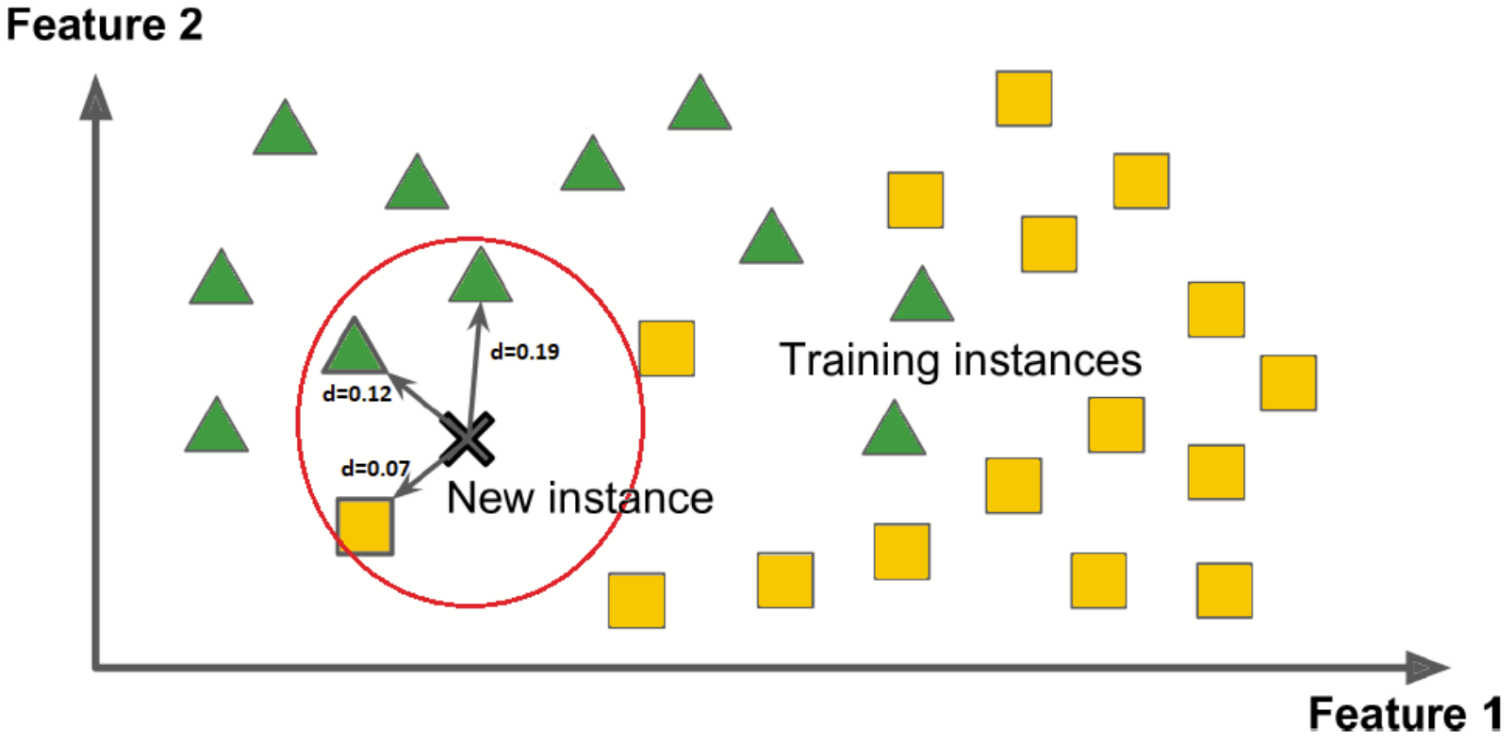


Abbildung 2: Beispiel für eine Klassifikation mit zwei Klassen und einer Nachbarschaft von $k = 3$ Punkten.

(Quelle: `https://mlarchive.com/machine-learning/k-nearest-neighbor-knn-explained/`

K-Nearest Neighbors

Einführung

Das Grundprinzip hinter K-Nearest Neighbors (KNN) kann sowohl für Klassifikation als auch Regression verwendet werden, wir betrachten jedoch nur die Klassifikation. Wir beginnen wieder beim optimalen Bayes-Klassifikator mit den Klassen $C_1, \ldots, C_m$. Wie gewohnt entscheiden wir uns bei gegebenem $\mathbf{x} \in \mathcal{X}$ für die Klasse mit der größten Wahrscheinlichkeit, also

$$f(\mathbf{x}) = \arg \max_{C_j \in \{C_1, \ldots, C_m\}} \Pr(y = C_j \mid X = \mathbf{x}).$$

K-Nearest Neighbors

Einführung

Leider ist die tatsächliche Funktion

$$\Pr(y = C_j \mid X = \mathbf{x})$$

meist nicht bekannt. Im Fall der binären logistischen Regression zum Beispiel hatten wir daher angenommen, dass die Funktion für Features $\mathbf{x} \in \mathbb{R}^d$ ausreichend gut approximiert werden kann durch

$$\Pr(y = 1 \mid X = \mathbf{x}) = \frac{e^{\mathbf{w}^T \mathbf{x}}}{1 + e^{\mathbf{w}^T \mathbf{x}}}$$

mit Hilfe der Parameter $\mathbf{w} \in \mathbb{R}^{d+1}$.

K-Nearest Neighbors

Algorithmus

Bei der KNN-Klassifikation wollen wir keine direkte Formel für die Wahrscheinlichkeiten $\Pr(y = C_j | X = \mathbf{x})$ angeben und somit auch ohne Parameter auskommen, welche die Form dieser Funktionen bestimmt. KNN-Klassifikation ist somit eine nicht-parametrische, lokale Methode.

K-Nearest Neighbors

Algorithmus

Die Idee ist, dass wir uns die alle Punkte $\mathbf{x}^{(i)}$ zusammen mit ihrer Klasse $y^{(i)}$ welche im Trainingsdatensatz

$$\mathcal{T} = \{(\mathbf{x}^{(i)}, y^{(i)}) \mid 1 \leq i \leq n\}$$

enthalten sind, direkt merken (an Stelle einer Trainingsphase wie in den anderen Methoden üblich).

K-Nearest Neighbors

Algorithmus

Für die eigentliche Klassifikation nehmen wir nun an, dass ein zu klassifizierender Datenpunkt **x** die gleiche Klasse hat, wie die Trainingsdatenpunkte in seiner Nähe.

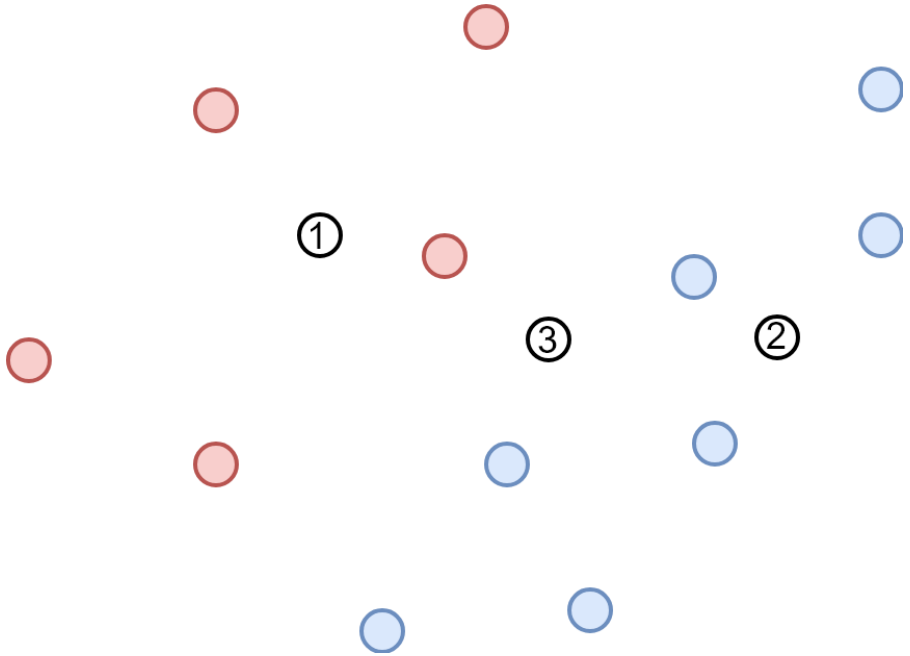


Abbildung 3: KNN-Klassifikation: Punkt 1 gehört wohl wahrscheinlich zur roten Klasse, Punkt 2 zur blauen Klasse und bei Punkt 3 ist die Klasse nicht deutlich erkennbar.

K-Nearest Neighbors

Algorithmus

Um genauer zu definieren, was mit Nähe gemeint ist, wollen als Distanzmetrik den euklidischen Abstand zwischen zwei Punkten $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ definieren als

$$\text{dist}(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_2 = \sqrt{\sum_{i=1}^{d} (\mathbf{x}_i - \mathbf{x}'_i)^2}.$$

Auch andere Distanzen sind möglich, z.B. die Manhatten-Distanz.

K-Nearest Neighbors

Algorithmus

Gegeben eine Menge von Trainingspunkten

$$\mathcal{T} = \{(\mathbf{x}^{(i)}, y^{(i)}) \mid 1 \leq i \leq n\}$$

und einen Punkt $\mathbf{x}$ definieren wir mit

$$\mathcal{N}^k(\mathbf{x}) \subseteq \mathcal{T}$$

die Menge der k Trainingspunkte $(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{T}$ mit der geringsten Distanz $\text{dist}(\mathbf{x}, \mathbf{x}^{(i)})$ zum Punkt $\mathbf{x}$. Logischerweise gilt

$$|\mathcal{N}^k(\mathbf{x})| = k.$$

K-Nearest Neighbors

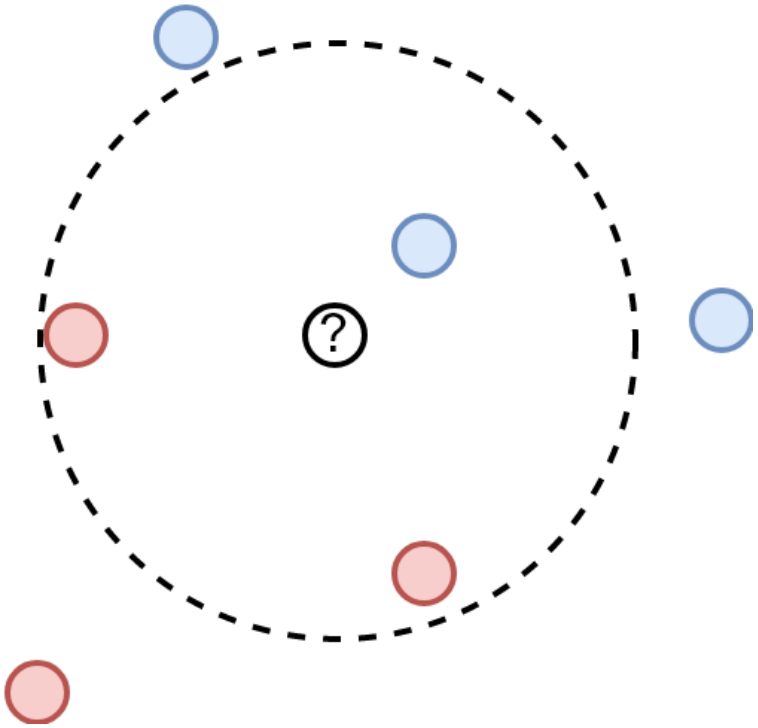
Algorithmus

Wir können nun die Wahrscheinlichkeit der Klassenzugehörigkeit eines Punktes zu einer Klasse im Sinne der KNN-Klassifikation definieren als

$$\Pr\left(y = C_j \middle| \; X = \mathbf{x}\right) = \frac{\left|\left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{N}^k(\mathbf{x}) \mid y^{(i)} = C_j\right\}\right|}{k}.$$

K-Nearest Neighbors

Algorithmus

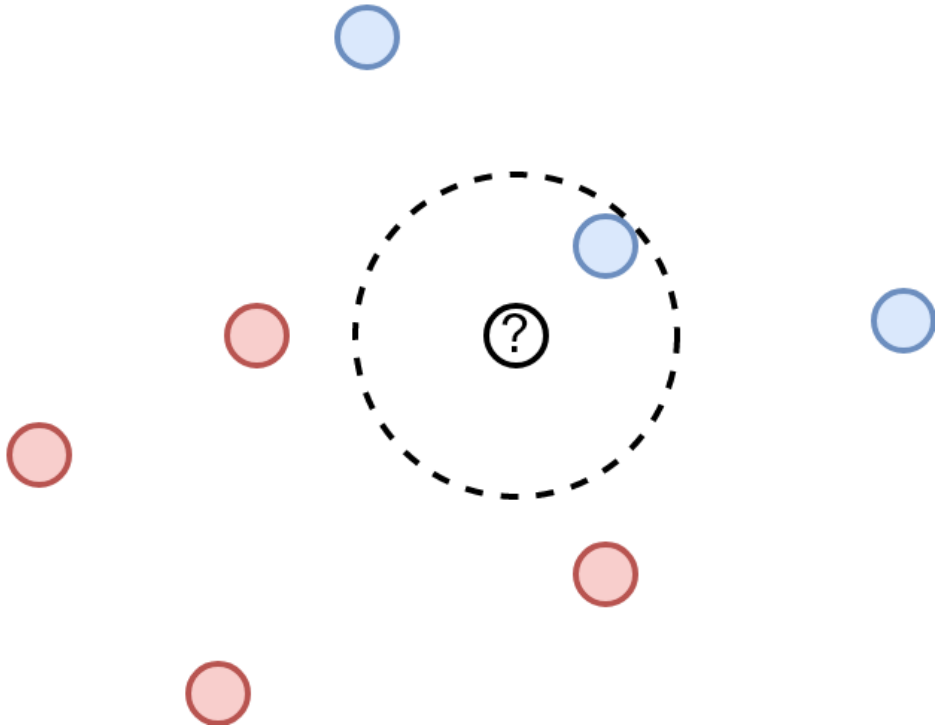


$$\Pr(y = C_r | X = \mathbf{x}) = \frac{2}{3}, \Pr(y = C_b | X = \mathbf{x}) = \frac{1}{3}$$

Abbildung 4: Beispiel KNN-Klassifikation mit zwei Klassen und $k = 3$.

K-Nearest Neighbors

Algorithmus

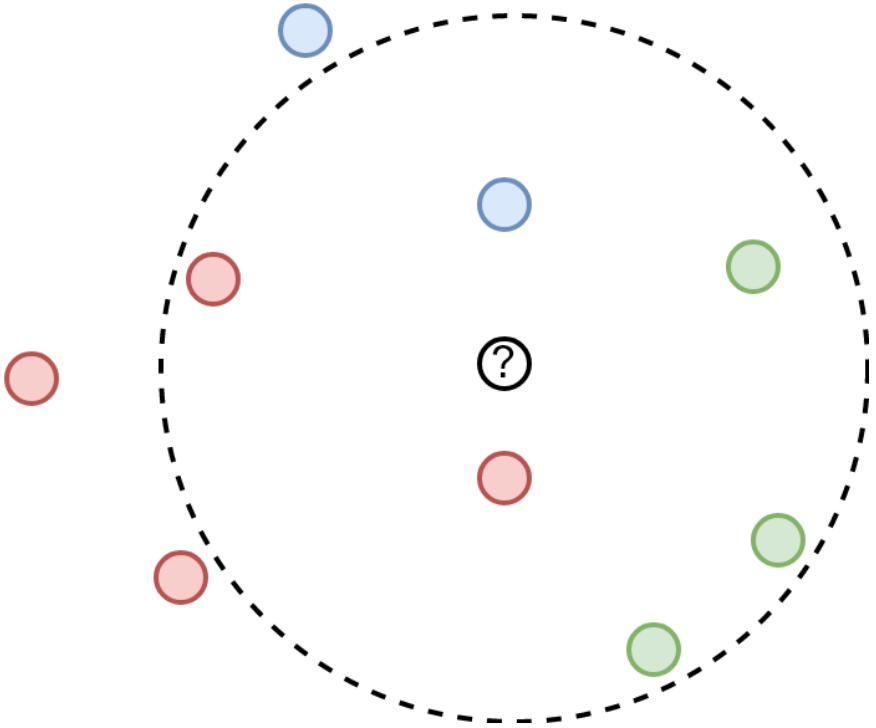


$$\Pr(y = C_r | X = \mathbf{x}) = 0, \Pr(y = C_b | X = \mathbf{x}) = 1$$

Abbildung 5: Beispiel KNN-Klassifikation mit zwei Klassen und $k = 1$.

K-Nearest Neighbors

Algorithmus



$$\Pr(y = C_r | X = \mathbf{x}) = \frac{1}{3}, \Pr(y = C_b | X = \mathbf{x}) = \frac{1}{6},$$

$$\Pr(y = C_g | X = \mathbf{x}) = \frac{1}{2}$$

Abbildung 6: Beispiel KNN-Klassifikation mit drei Klassen und $k = 6$.

K-Nearest Neighbors

Algorithmus

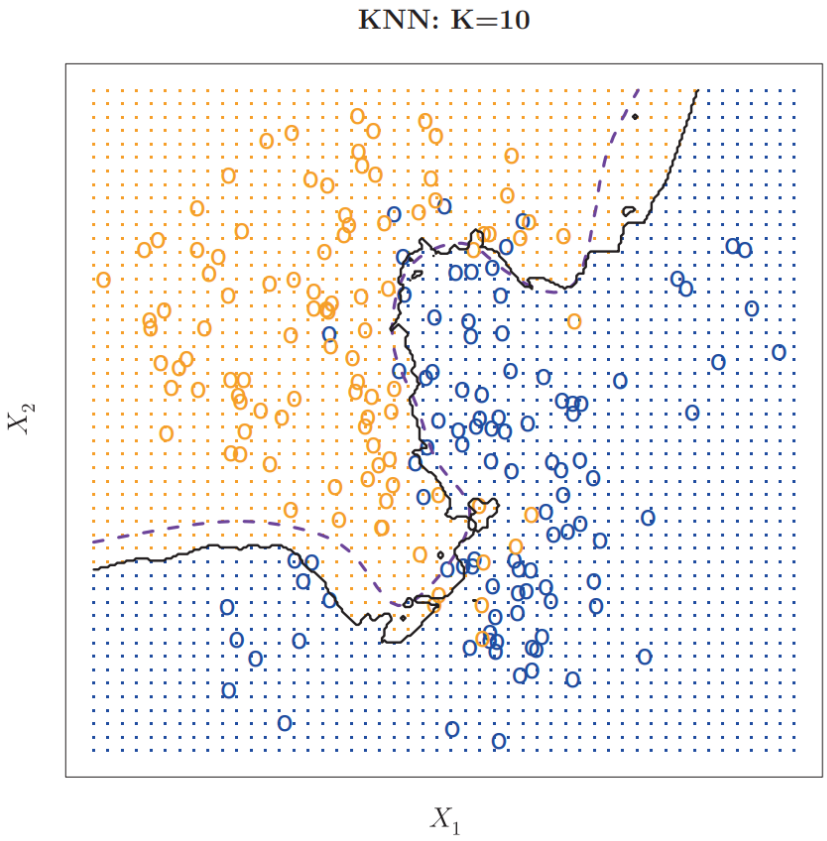


Abbildung 7: Beispiel einer binären Klassifikation mit Hilfe KNN ($k = 10$). Die gelernte KNN-Entscheidungsgrenze approximiert die des optimalen Bayes Klassifizierer relativ gut. Abbildung entnommen aus [JWHT14].

K-Nearest Neighbors

Algorithmus

K-Nearest Neighbors Methoden besitzen normalerweise einen einzigen Hyperparameter $k \in \mathbb{N}$, welcher maßgeblich die Leistung beeinflusst. Ziel sollte daher sein, den Hyperparameter k zu optimieren, um

- die Fehlerrate auf dem Testdatensatz zu minimieren bzw. äquivalent dazu,
- die Genauigkeit zu maximieren.

K-Nearest Neighbors

Algorithmus

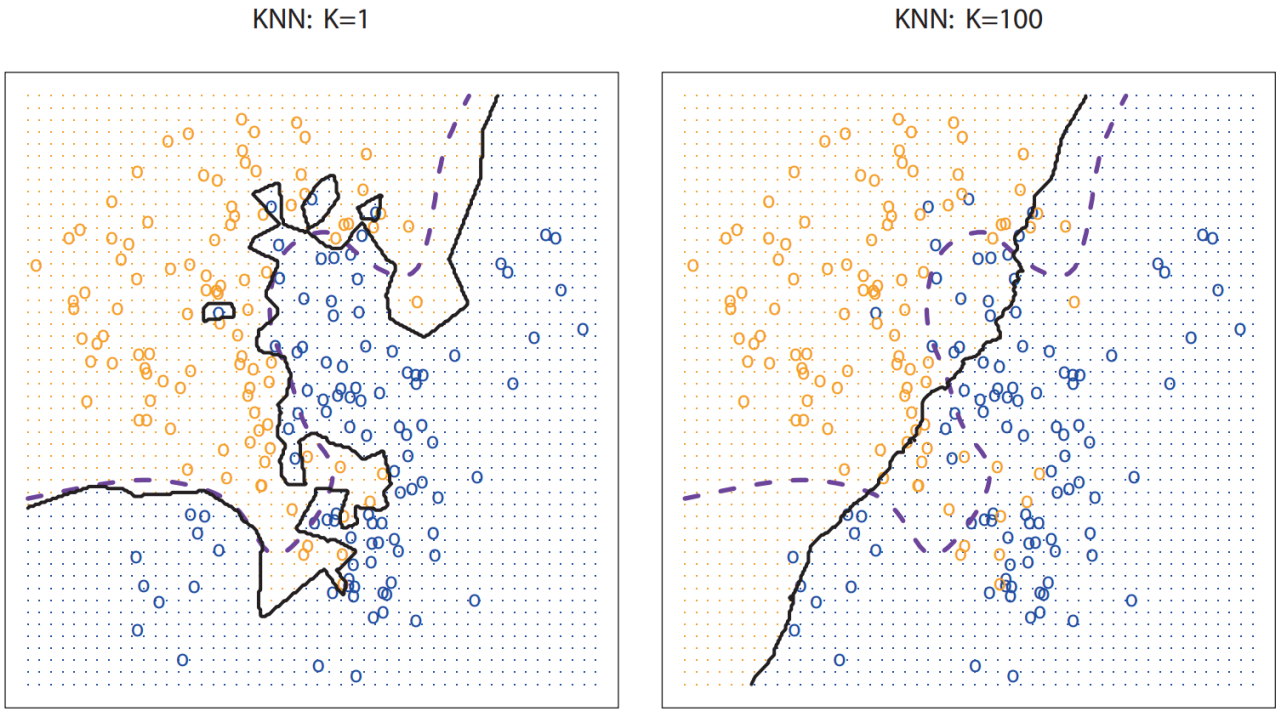


Abbildung 8: Auswirkungen der Wahl des Hyperparameters k in KNN. Ist k zu gering, werden die Trainingsdaten zu stark *auswendig gelernt* und es kommt zur Überanpassung. Ist k zu hoch, ist das Model nicht flexibel genug und es kommt zur Unteranpassung. Abbildung entnommen aus [JWHT14].

K-Nearest Neighbors

Algorithmus

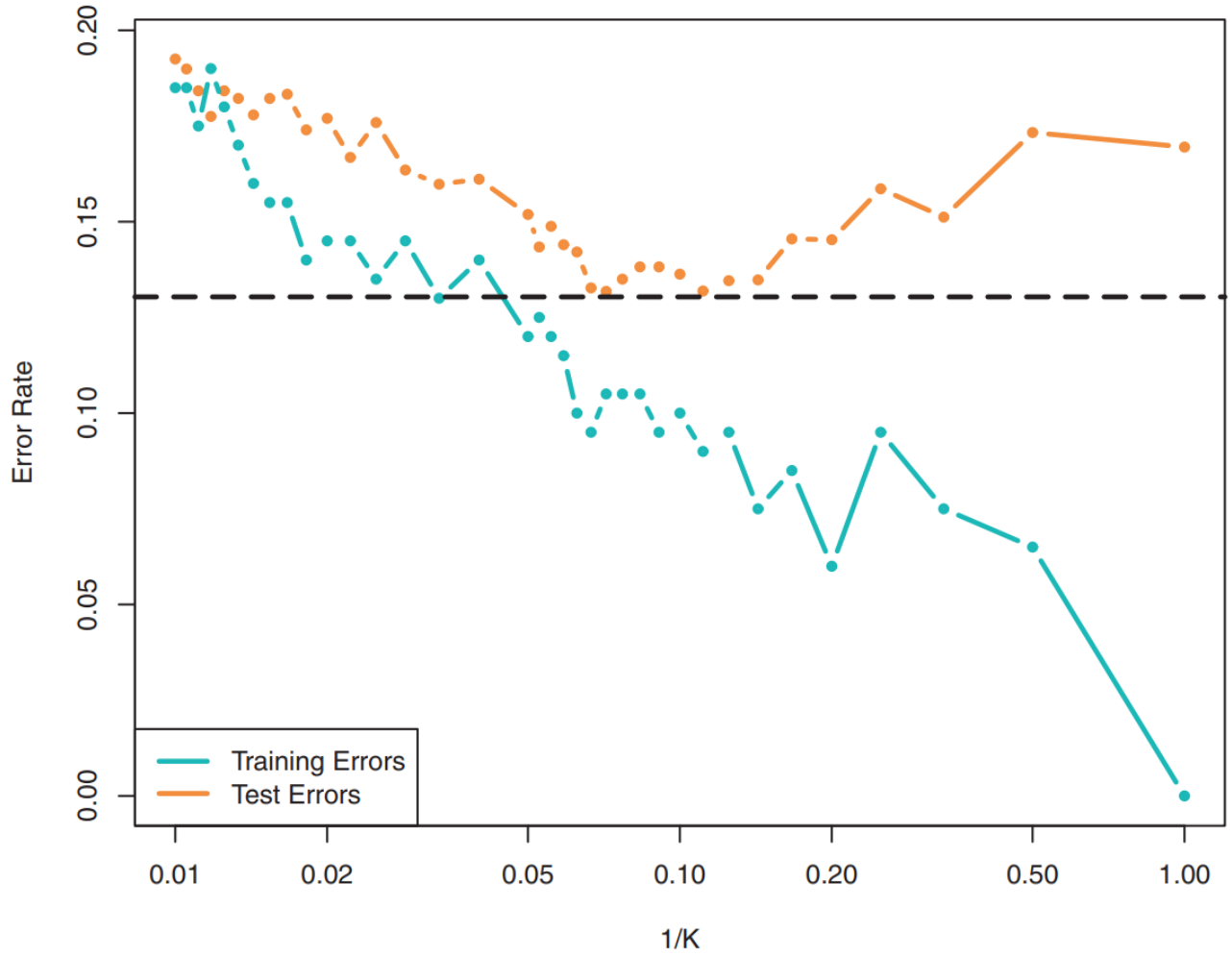


Abbildung 9: Die Fehlerrate auf den Test- und Trainingsdaten in Abhängigkeit von k. Abbildung entnommen aus [JWHT14].

K-Nearest Neighbors

Algorithmus

Eine effiziente Implementierung von KNN ist algorithmisch herausfordernd und ist immer noch Stand der Forschung. Ein naiver Ansatz ist es bei jeder von t Klassifikationen, die euklidischen Abstände zu allen n Trainingsdaten zu berechnen und danach zu ordnen, um die k nächsten Punkte zu berechnen. Angenommen, t ist in der Größenordnung von n, dann hat ein solcher Algorithmus eine Laufzeitkomplexität von

$$\mathcal{O}(n^2 \cdot \log n)$$

was meist durch effiziente Datenstrukturen und Algorithmen in der Praxis reduziert wird.

K-Nearest Neighbors

Zusammenfassung

- Bei KNN passiert keine wirkliche Komprimierung der Trainingsdaten; es speichert alle Daten und braucht diese auch für Vorhersagen auf neuen Beobachtungen
- KNN basiert auf keiner Verteilungsannahme und keiner Annahme einer funktionalen Struktur; theoretisch kann KNN beliebig komplexe Muster lernen
- **Standardisierung**: meist werden alle Features standardisiert; wenn zwei Features Werte auf unterschiedlichen Skalen annehmen, dann erhält das Feature mit der größeren Skala die größere Wichtigkeit (das gilt für die Euklidische Distanz und auch für die meisten anderen Distanzen)
- Je kleiner k, umso weniger weniger glatt sind die Entscheidungsgrenzen
- Die Genauigkeit von KNN kann stark verschlechtert werden durch Hinzufügen verrauschter oder irrelevanter Einflussgrößen

K-Nearest Neighbors

Ausblick

Ausblick: KNN für Regressionprobleme

- man kann KNN auch zur Prognose einer kontinuierlichen Zielgröße verwenden
- dazu berechnet man den Mittelwert der Zielgröße aller Beobachtungen in der Nachbarschaft

K-Nearest Neighbors

References

G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning: With applications in R*, Springer Publishing Company, Incorporated, 2014.