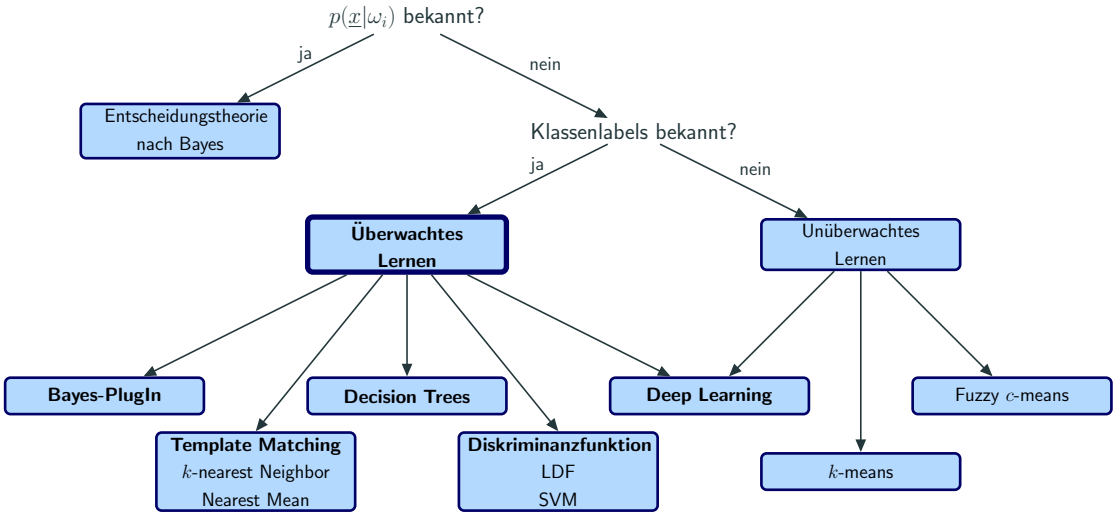
Kapitel 5 - Überwachtes Lernen

Annika Liebgott

February 27, 2022

5.1 - Einführung

Überblick


$p(\underline{x}|\omega_i)$ bekannt?
ja
nein
Entscheidungstheorie nach Bayes
Klassenlabels bekannt?
ja
nein
Überwachtes Lernen
Unüberwachtes Lernen
Bayes-PlugIn
Decision Trees
Deep Learning
Fuzzy c-means
Template Matching
k-nearest Neighbor
Nearest Mean
Diskriminanzfunktion
LDF
SVM
k-means

Wiederholung: was bedeutet überwachtes Lernen?

Training

Klassifikation

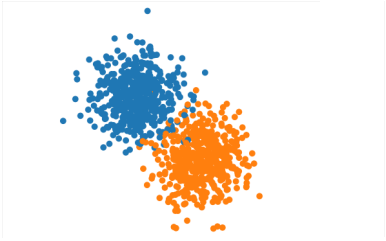
Wiederholung: was bedeutet überwachtes Lernen?

Training

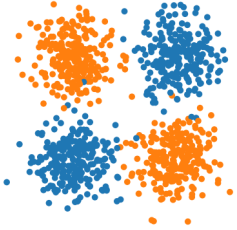
Klassifikation

Datensets

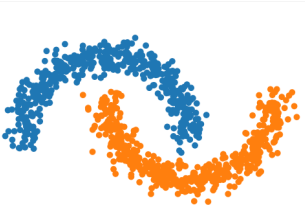
Dataset 1 (Blobs)



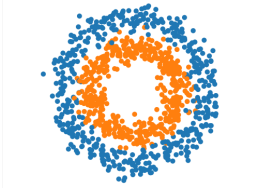
Dataset 2 (Chess)



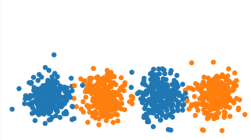
Dataset 3 (Bananas)



Dataset 4 (Donut)



Dataset 5 (Alternate)



5.2 - Bayes Plug-in

5.2 - Bayes Plug-in

5.2.1 - Naive Bayes

Spezialfall: Naiver Gauß-Klassifikator

Annahmen:

1. Gaußförmige Likelihood $p(\underline{x}|\omega_j) \sim \mathcal{N}(\underline{\mu}_j, \mathbf{C}_j)$
2. naive Annahme: $\mathbf{C}_j = \begin{pmatrix} \sigma_{1,j}^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_{1,d}^2 \end{pmatrix}$

ML-Schätzung von $\underline{\vartheta}_{ij}$:

$$\hat{\mu}_j = \frac{1}{N_j} \sum_{n=1}^{N_j} \underline{x}_n \qquad \sigma_{i,j}^2\text{: Diagonalelemente von } \hat{\mathbf{C}}_j = \frac{1}{N_j} \sum_{n=1}^{N_j} (\underline{x}_n - \hat{\mu}_j)(\underline{x}_n - \hat{\mu}_j)^{\mathrm{T}}$$

Vorteil: weniger Parameter zu schätzen $\Rightarrow$ weniger Trainingssamples nötig

Nachteil: Merkmale selten unabhängig $\Rightarrow$ nur Approximation möglich

5.2 - Bayes Plug-in

5.2.2 - Gaussian Mixture Model

Gaussian Mixture Model - Beispiele

Beispiele für Anwendungen von GMM-Klassifikatoren:

- Spracherkennung[1]
- Bildklassifikation[2]
- Fehlervorhersage[3]
- Robotik[4]

[1] Patel et al.: "Emotion Recognition from Speech with Gaussian Mixture Models and via Boosted GMM", 2017

[2] Alroobaea et al.: "Bayesian inference framework for bounded generalized Gaussian-based mixture model and its application to biomedical images classification", 2019

[3] Zhang et al.: "Detection of Emerging Faults on Industrial Gas Turbines Using Extended Gaussian Mixture Models", 2017

[4] Pignat et al.: "Bayesian Gaussian Mixture Model for Robotic Policy Imitation", 2019

5.3 - Template Matching

5.3 - Template Matching

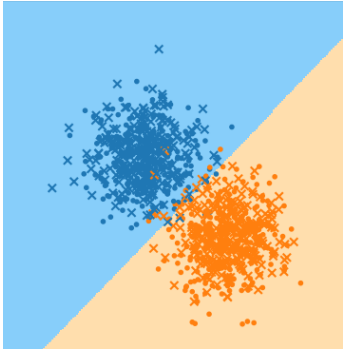
5.3.1 - Nearest Mean

Zusammenfassung Nearest Mean

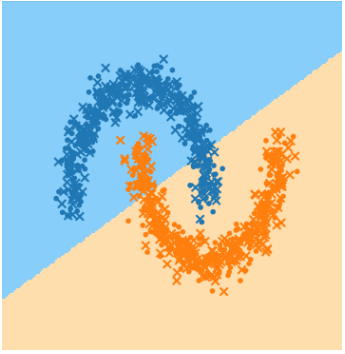
Vorteil: einfach zu implementierender Klassifikator, geringer Rechenaufwand

Nachteil: nur lineare Klassifikation möglich

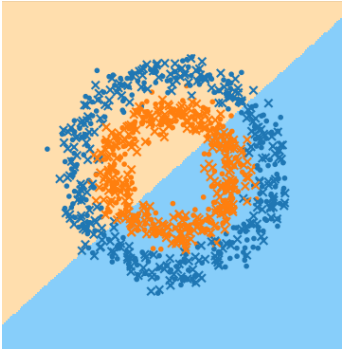
Dataset 1: $\mathrm{ACC} = 97.8\,\%$



Dataset 3: $\mathrm{ACC} = 86.8\,\%$



Dataset 4: $\mathrm{ACC} = 47.2\,\%$



5.3 - Template Matching

5.3.2 - k-Nearest Neighbor

Zusammenfassung k-Nearest Neighbor

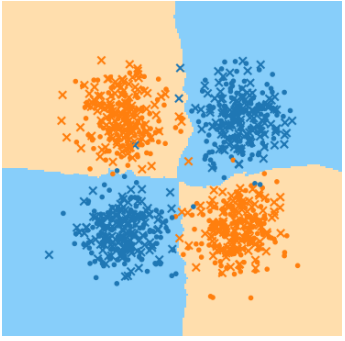
Vorteil: einfach zu implementieren, nichtlineare Trennung möglich

Nachteil: bei vielen Trainingssamples schnell hoher Rechenaufwand

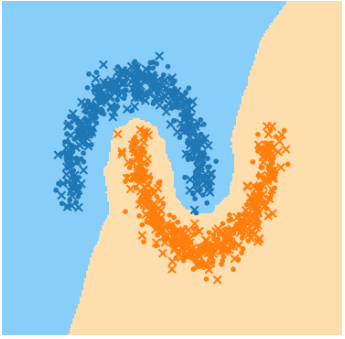
Mit $k = 5$ Nachbarn:

Dataset 1: $\mathrm{ACC} = 97.2\,\%$

Dataset 2: $\mathrm{ACC} = 99.0\,\%$



Dataset 3: $\mathrm{ACC} = 100.0\,\%$



5.4 - Support Vector Machines

5.4 - Support Vector Machines

5.4.2 - Binäre SVM für (annähernd) linear trennbare Daten

Prinzip SVM[5,6]

Grundprinzipien:

- Binärer Klassifikator $\Rightarrow y_n \in \{1, -1\}$ für ω_1, ω_2
- Nutzung von LDF: $f(\underline{x}) = \underline{w}^{\mathrm{T}}\underline{x} + w_0$
- Geschätzte Klasse $\hat{y}(\underline{x}) = \mathrm{sgn}(f(\underline{x})) \in \{1, -1\}$
- Möglichkeit zur nichtlinearen Klassifikation
- Maximum-Margin-Optimierung

[5] V. Vapnik and A. Chervonenkis, 1963

[6] C. Cortes und V. Vapnik, "Support-vector networks," in Machine Learning, 1995, S.273–297.

Prinzip SVM[5,6]

Grundprinzipien:

- Binärer Klassifikator $\Rightarrow y_n \in \{1, -1\}$ für ω_1, ω_2
- Nutzung von LDF: $f(\underline{x}) = \underline{w}^{\mathrm{T}}\underline{x} + w_0$
- Geschätzte Klasse $\hat{y}(\underline{x}) = \mathrm{sgn}(f(\underline{x})) \in \{1, -1\}$
- Möglichkeit zur nichtlinearen Klassifikation
- Maximum-Margin-Optimierung

Warum Maximum-Margin-Optimierung?

[5] V. Vapnik and A. Chervonenkis, 1963

[6] C. Cortes und V. Vapnik, "Support-vector networks," in Machine Learning, 1995, S.273–297.

Prinzip SVM[5,6]

Grundprinzipien:

- Binärer Klassifikator $\Rightarrow y_n \in \{1, -1\}$ für ω_1, ω_2
- Nutzung von LDF: $f(\underline{x}) = \underline{w}^{\mathrm{T}}\underline{x} + w_0$
- Geschätzte Klasse $\hat{y}(\underline{x}) = \mathrm{sgn}(f(\underline{x})) \in \{1, -1\}$
- Möglichkeit zur nichtlinearen Klassifikation
- Maximum-Margin-Optimierung

Warum Maximum-Margin-Optimierung?

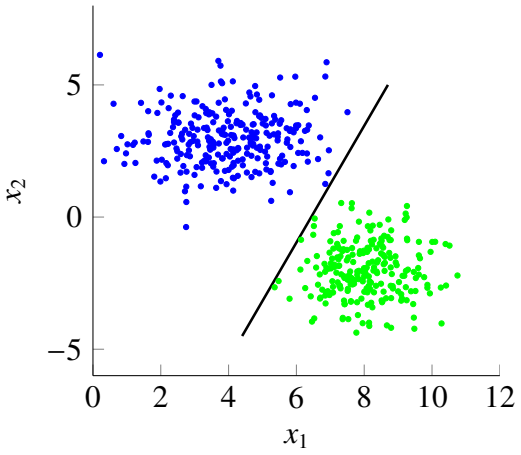
Annahme: Datenset $\mathcal{D}$ linear trennbar

$\Rightarrow$ unendliche Anzahl möglicher LDFs. Welche ist die beste?

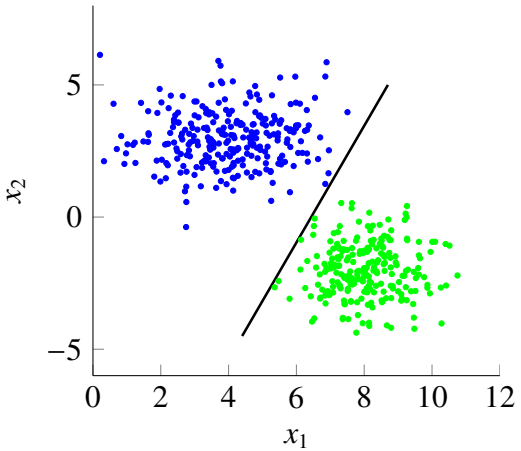
[5] V. Vapnik and A. Chervonenkis, 1963

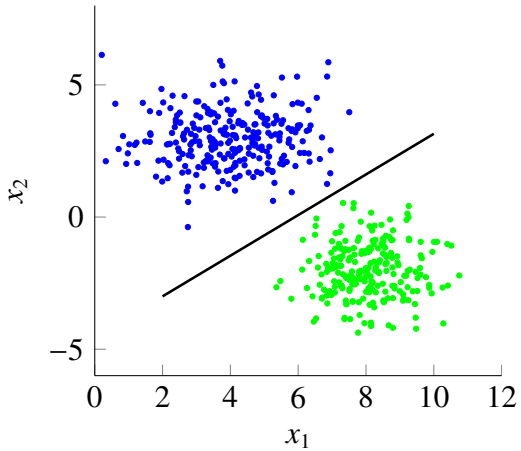
[6] C. Cortes und V. Vapnik, "Support-vector networks," in Machine Learning, 1995, S.273–297.

Veranschaulichung Maximum-Margin-Optimierung

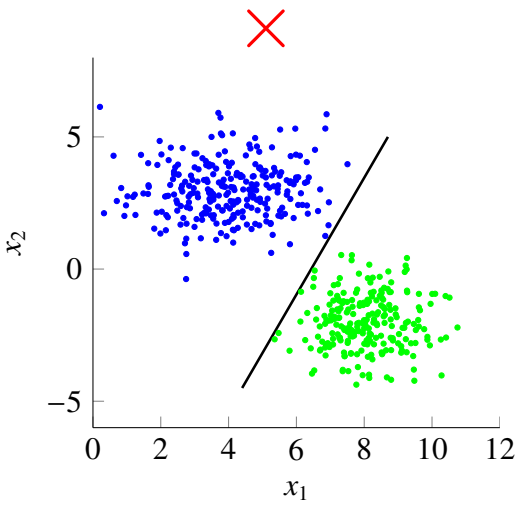


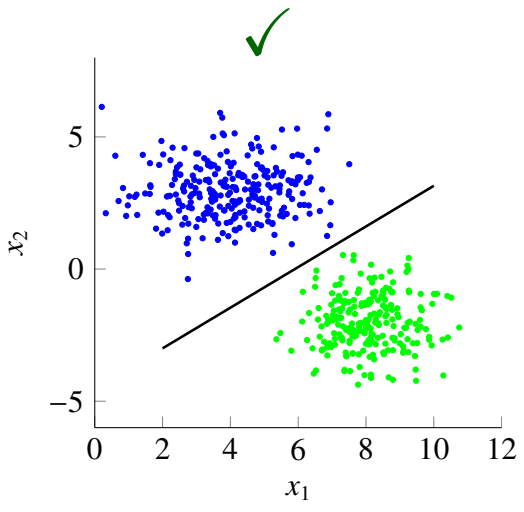
Veranschaulichung Maximum-Margin-Optimierung



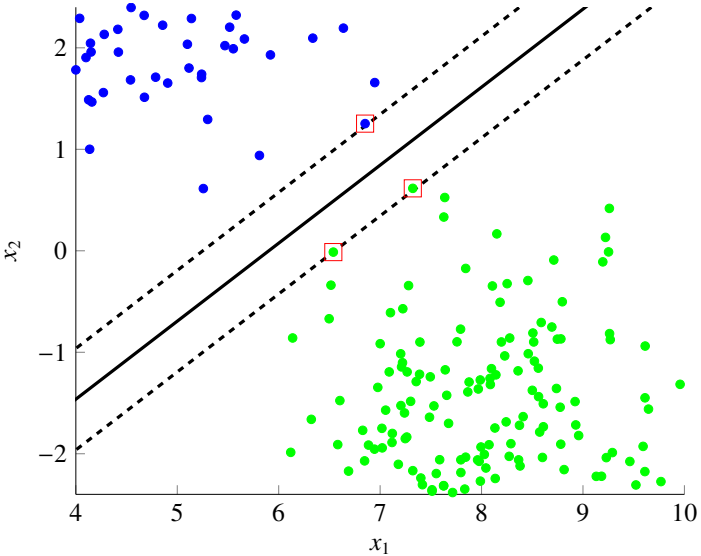


Veranschaulichung Maximum-Margin-Optimierung





Support-Vektoren



□ ⇒ Support-Vektoren:

- Trainingssamples mit geringstem Abstand zur gelernten Trennebene
- reichen aus, um Entscheidungsgrenze vollständig zu beschreiben
- oft reicht 1 Support-Vektor pro Klasse

Prinzip Soft-Margin-SVM

Idee: Einführung einer Schlupfvariablen ξ und eines Fehlergewichts C:

$$\min_{\underline{w},w_0} \frac{1}{2}\|\underline{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$

ξ_n : bestimmt den Outlier-Grad eines Samples $\underline{x}_n$

$\xi_n = 0$: $\underline{x}_n$ liegt auf β_1 oder β_{-1}
$0 < \xi_n < 1$: $\underline{x}_n$ liegt auf der richtigen Seite von β_{12}, aber in Margin
$\xi_n = 1$: $\underline{x}_n$ liegt auf β_{12}
$\xi_n > 1$: $\underline{x}_n$ liegt auf der falschen Seite von β_{12}

C : bestimmt, wie stark Outlier in der Optimierung gewichtet werden sollen

Optimierungsproblem Soft-Margin-SVM

Das Gesamte Optimierungsproblem mit Nebenbedingungen wird dadurch zu:

$$\min_{\underline{w},w_0} \frac{1}{2}\|\underline{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$

$$\text{s.t. } \xi_n \geq 0$$

$$\text{s.t. } y_n(\underline{w}^{\mathrm{T}}\underline{x}_n + w_0) \geq 1 - \xi_n$$

Zusammenfassung Soft-Margin-SVM

Wunsch: Margin maximieren und gleichzeitig $\sum \xi_n$ gering halten

$\Rightarrow$ Optimierung $\widehat{=}$ Kompromiss finden

5.4 - Support Vector Machines

5.4.3 - Nichtlineare Klassifikation mit SVMs

Beliebte Kernelfunktionen

Häufig verwendete Kernelfunktionen:

$$\text{Linearer Kernel: } k(\underline{x}_i, \underline{x_j}) = \underline{x}_i^{\mathrm{T}} \underline{x}_j$$

$$\text{Polynomieller Kernel: } k(\underline{x}_i, \underline{x_j}) = (\underline{x}_i^{\mathrm{T}} \underline{x}_j)^d$$

$$\text{Radiale Basisfunktion (RBF)-Kernel: } k(\underline{x}_i, \underline{x_j}) = \exp(-\gamma \|\underline{x}_i - \underline{x}_j\|^{\mathrm{T}})$$

$\gamma > 0$: Konstante, die RBF-Radius bestimmt

Hinweis: um den Kernel-Trick anwenden zu können, muss das Optimierungsproblem in seine duale Form überführt werden.[7]

$$\tilde{L}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\underline{x}_i, \underline{x}_j)$$

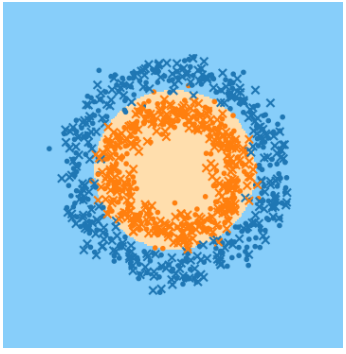
[7] Scikit-learn und andere SVM-Frameworks nutzen meist standardmäßig das duale Problem.

Beispiel 5.3: Wahl der Hyperparameter für RBF-Kernel

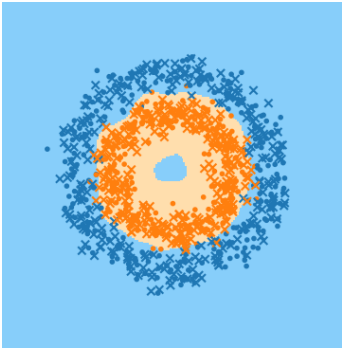
RBF-Kernel sind beliebt. Aber: Hyperparameter γ spielt große Rolle für Performance

Beispiel: Datenset 4 (Donut)

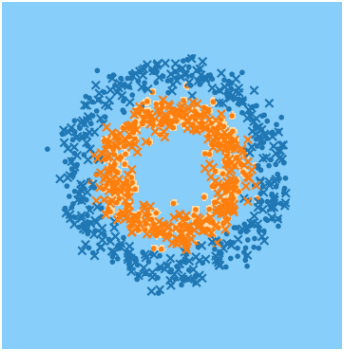
$\gamma = 0.01$: $\mathrm{ACC} = 97.6\,\%$



$\gamma = 10$: $\mathrm{ACC} = 94.6\,\%$



$\gamma = 500$: $\mathrm{ACC} = 77.0\,\%$



$\Rightarrow$ zu hoher Wert für γ führt zu Overfitting

5.4 - Support Vector Machines

5.4.4 - Multiclass-SVM

Multiclass-SVM (1)

Problem: SVMs sind grundsätzlich binäre Klassifikatoren. Wie kann zwischen c Klassen unterschieden werden?

Lösung: Umformulierung eines c-Klassen-Problems in mehrere binäre Probleme

Möglichkeit 1: "One vs. One"-Klassifikation

Vorgehen:

- Trainiere eine SVM für jede mögliche Kombination zweier Klassen
  $\Rightarrow \frac{c(c-1)}{2}$ SVMs nötig
- Klassifikation: bestimme $\hat{y}$ für jede trainierte SVM
- Wähle die Klasse, für die sich die meisten SVMs entschieden haben

Multiclass-SVM (2)

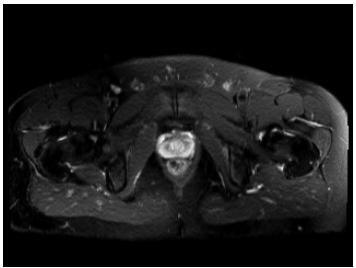
Möglichkeit 2: "One vs. All"-Klassifikation

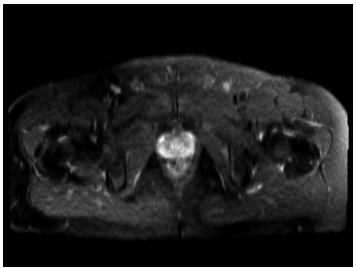
Vorgehen:

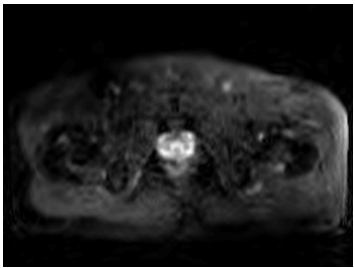
- Trainere jeweils eine SVM, um eine Klasse von allen anderen zu unterscheiden
  $\Rightarrow c$ SVMs nötig
- Eine gelernte Entscheidungsfunktion pro Klasse: $f_k(\underline{x}) = \underline{w}_k^{\mathrm{T}} \underline{x} + w_0$
  Trainingssamples aus Klasse k: $f_k > 0$
  Trainingssamples aus anderen Klassen: $f_k < 0$
- Klassifikation: bestimme Wert von f_k für jede trainierte SVMs
- Wähle die Klasse, für die der höchste Wert f_k erreicht wird

Beispiel 5.4: Automatische Qualitätsbewertung von MRT-Bildern (1)

Klassifikationsaufagabe: Bewertung von Magnetresonanztomografie-Bildern auf einer Skala von 1 (sehr gut) bis 5 (sehr schlecht)[8]







[8] Liebgott et al.: "Active Learning for Magnetic Resonance Image Quality Assessment", ICASSP 2016

Beispiel 5.4 Automatische Qualitätsbewertung von MRT-Bildern (2)

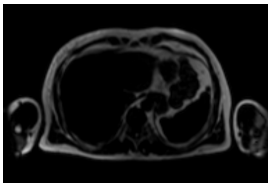
Klassifikations-Setup:

- Input-Daten: 2911 2D-Bilder von 100 Patienten, gelabelt durch Radiologen
- 2871 Merkmale basierend auf Graustufenvarianzen
- Merkmalsreduktion: PCA, beste Performance bei 36 Komponenten
- Klassifikator: Soft-Margin Multi-Class Support Vector Machine mit RBF-Kernel
- 70 % Trainings-, 30 % Testdaten

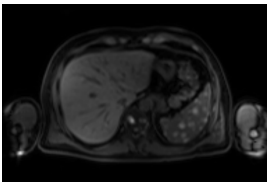
Mittlere Accuracy bei Evaluation mit 10-facher Kreuzvalidierung: $\mathrm{ACC} = 91.2\,\%$

Beispiel 5.5: Vorhersage des Therapieansprechens auf Immuntherapie (1)

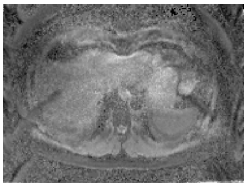
Klassifikationsaufgabe: Vorhersage des Therapieansprechens auf Immuntherapie von Melanom-Patienten aus multiparametrischen PET/MRT-Aufnahmen[9]



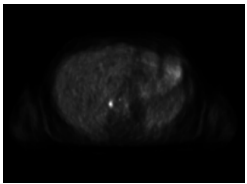
Fett-gewichtetes MRT-Bild



Wasser-gewichtetes MRT-Bild



Diffusions-gewichtetes MRT-Bild

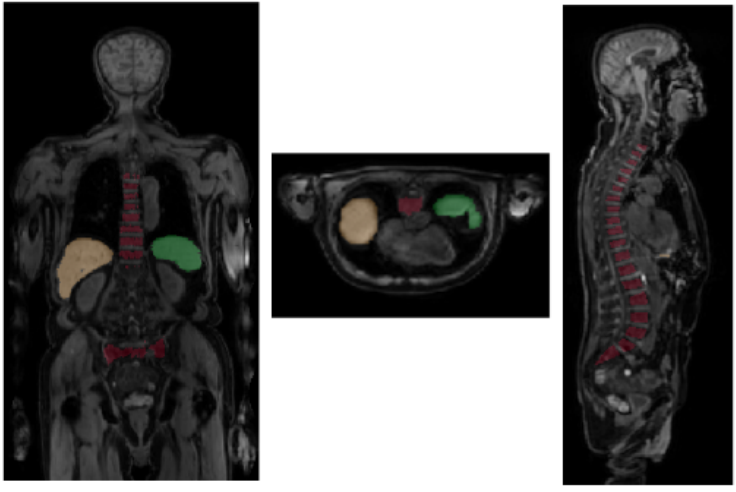


PET-Bild

[9] Liebgott et al.: "Feature-based Response Prediction to Immunotherapy of late-stage Melanoma Patients Using PET/MR Imaging", EUSIPCO 2020

Beispiel 5.5: Vorhersage des Therapieansprechens auf Immuntherapie (2)

Klassifikation: anhand mit Immunsystem assoziierter Organe Leber, Milz, Wirbelsäule



Beispiel 5.5: Vorhersage des Therapieansprechens auf Immuntherapie (3)

Klassifikations-Setup:

- Input-Daten: 3D-Aufnahmen von 24 Patienten zu jeweils 3 Therapie-Zeitpunkten $\Rightarrow$ 72 Samples
- 810 Graustufenvarianz-Merkmale, 1 Merkmal je Organ und Aufnahmetyp $\Rightarrow 810 \cdot 3 \cdot 4 = 3240$ Merkmale je Sample
- Merkmalsreduktion: DL-basiertes Selektionsverfahren über Concrete Autoencoder, beste Performance mit 14 selektierten Merkmalen
- Klassifikator: Soft-Margin Support Vector Machine with RBF-Kernel
  Hyperparameter: $C = 2^9$, $\gamma = 2^{-7}$
- 70 % Trainings-, 30 % Testdaten

Mittlere Balanced Accuracy mit 5-facher Kreuzvalidierung: $\mathrm{BACC} = 91.11\,\%$

5.5 - Baumbasierte Klassifikation

5.5 - Baumbasierte Klassifikation

5.5.1 - Entscheidungsbäume

Was sind Entscheidungsbäume?

Grundidee: Darstellung hierarchischer Abfolge von Entscheidungen

Anwendungsgebiete:

- Stochastik (z.B. Wahrscheinlichkeitsbäume zur Veranschaulichung bedingter Wahrscheinlichkeiten)
- Übersichtliche Darstellung von Entscheidungen (z.B. Notfallpläne, Bussinesmodelle, ärztliche Entscheidungen)
- Programmierung (z.B. hierarchische Abfolge von Methoden)
- Entscheidungstheorie und maschinelles Lernen

Eigenschaften von Entscheidungsbäumen im ML

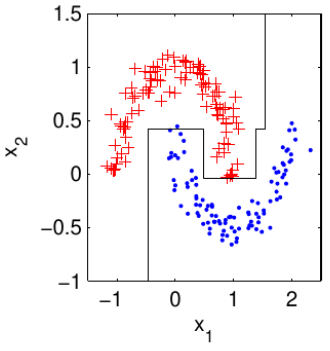
Allgemein:

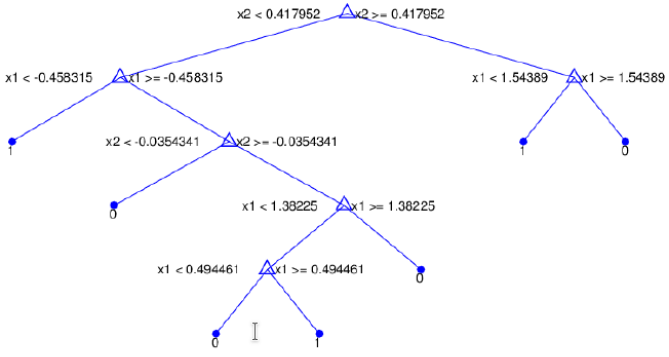
- Entscheidungsbäume können binäre oder multinomiale Entscheidungen beinhalten, die auf ja/nein-Fragen, Zahlen, Kategorien,... basieren.
- Jeder Entscheidungsbaum lässt sich in einen rein binären Baum umformulieren

Annahmen im maschinellen Lernen:

- Rein binäre Entscheidungsbäume: jeder Elternknoten besitzt 2 Kindknoten
- Jede Entscheidung ist eine ja/nein-Frage
- Jede ja/nein-Frage bezieht sich auf exakt ein Merkmal
  (z.B. "$x_1 < 0$?", nicht "$x_1 < x_2$?")
  $\Rightarrow$ Entscheidungsgrenzen verlaufen orthogonal zu Merkmals-Achsen

Beispiel 5.7: Entscheidungsgrenze für Banana-Datenset





Training eines Entscheidungsbaums

Annahme: Trainingsset $\mathcal{D}_{\text{train}}$, Featureset $\mathcal{X}$ mit d Merkmalen x

Design des Trainings:

1. An jedem Elternknoten: Ein Merkmal x wählen, das getestet wird
2. Für dieses Merkmal entscheiden, welcher Schwellwert für den Test genutzt wird
3. Regelungen für Splitting-Stop festlegen
4. Blättern die passenden Klassen zuweisen

Ziel: "Reine" Blätter, d.h. alle Samples in diesem Blatt haben idealerweise das selbe Klassenlabel

Frage: Wie wähle ich Merkmale und Schwellwerte?

Maß für “gute” Klassenverteilung

Maß für “gute” Klassenverteilung: “Unreinheit” der Sets, z.B. durch

- Entropie: $H(\mathcal{D}) = -\sum_{i=1}^{N_c} p_i \log p_i$
- Gini-Index: $G(\mathcal{D}) = 1 - \sum_{i=1}^{N_c} p_i^2$

Bei beiden gilt:

Kleiner Wert $\rightarrow$ geringe Unreinheit

Großer Wert $\rightarrow$ hohe Unreinheit

Beispiel:

$H(\mathcal{D}) = 0$ und $G(\mathcal{D}) = 0$, falls nur eine Klasse in $\mathcal{D}$ vorkommt

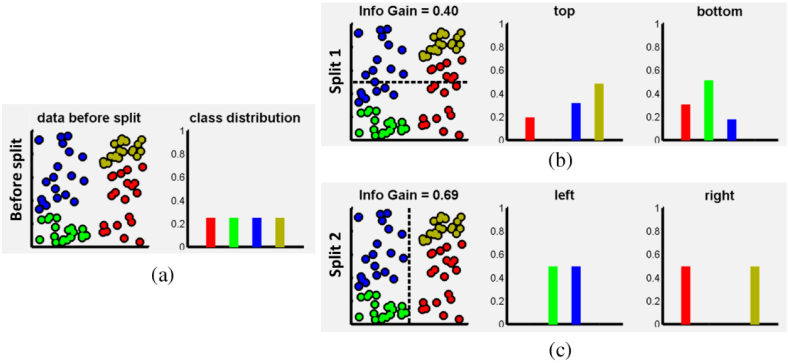
$H(\mathcal{D}) = 1$ und $G(\mathcal{D}) = 1 - \frac{1}{N_c}$, falls die Klassen in $\mathcal{D}$ gleichverteilt sind

Wie gut ist ein Split?

Aufgabe: Teile Set $\mathcal{D}$ so in zwei Teile, dass die Klassen "besser" verteilt sind

Ziel: Reduzierung der Unreinheit $\hat{=}$ Informationsgewinn

Beispiel 5.8:[10]



[10] Quelle: A. Criminisi, J. Shotton, E. Konukoglu et al., "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning", Foundations and Trends in Computer Graphics and Vision, vol. 7, pp. 81-227, 2012.

Wie lange wird gesplittet?

Zur Erinnerung: das Ziel sind “reine” Blätter

Wann wird das Splitten gestoppt?

- Splitten zu früh gestoppt: Blätter sind nicht rein genug, großer Trainingsfehler → Underfitting
- Splitten zu spät gestoppt: Blätter enthalten nur ein oder wenige Samples, der Baum entspricht einer Lookup Table für das Trainingsset → Overfitting

Wie kann Overfitting verhindert werden?

- Testfehler im Auge behalten
- Splitten beenden, wenn der Informationsgewinn zu gering ist
- Splitten beenden, wenn zu wenig Samples in einem Blatt sind
- Zurückschneiden des Baums (“pruning”): Äste/Teilbäume mit zu geringem Informationsgewinn werden abgeschnitten

5.5 - Baumbasierte Klassifikation

5.5.2 - Random Forest

Random Forests (1)

Idee: Training von N unterschiedlichen Entscheidungsbäumen und Kombination der Ergebnisse $\rightarrow$ Vermeidung von Overfitting und Erhöhung der Robustheit durch geringe Korrelation zwischen den Bäumen

Erzeugung der Bäume:

- "Bagging" (*b*ootstrap *agg*regat*ing*): Aus dem Trainingsset $\mathcal{D}_{\text{train}}$ werden zufällig N verschiedene Sets $\mathcal{D}_n$ gezogen (mit Zurücklegen)
- Für jedes Datenset wird ein Baum erzeugt, dabei werden für jeden Knoten nur k Merkmale aus dem Merkmalsset $\mathcal{X}$ zufällig gewählt, die für den Split verwendet werden können
- Kein Pruning

Random Forests (2)

Wie werden die Ergebnisse kombiniert?

- Klassifikation: Mehrheitsentscheid
- Regression: Mittelwert oder Median

Wie groß soll k sein?

- k klein: geringe Korrelation zwischen den Bäumen, aber hohe Fehlerrate der einzelnen Bäume $\rightarrow$ hohe Fehlerrate des Random Forests
- k groß: geringe Fehlerrate der einzelnen Bäume, aber hohe Korrelation zwischen den Bäumen $\rightarrow$ schlechte Generalisierung und damit hohe Testfehlerrate des Random Forests

$\rightarrow$ Trade-Off. Generell gilt aber: $k \ll |\mathcal{X}|$

Beispiel 5.9: Vorhersage des Therapieansprechens auf Immuntherapie[11]

Aus Beispiel 5.5: Ergebnis nach SVM-Klassifikation bei $\mathrm{BACC} = 91.11\,\%$

Ergebnisse für Klassifikation mit Random Forest, kombiniert mit Feature-Selektion über Concrete Autoencoders:

$\mathrm{BACC} = 94.40\,\%$, Anzahl benötigter Merkmale: 14

[11] Liebgott et al.: "Feature-based Response Prediction to Immunotherapy of late-stage Melanoma Patients Using PET/MR Imaging", EUSIPCO 2020

5.6 - Allgemeine Anmerkungen zu Supervised Learning

Vergleich von Klassifikatoren (1)

	Vorteile	Nachteile
kNN	• einfach zu verstehen	• keine glatte Entscheidungsgrenze • empfindlich in Bezug auf k • hoher Rechenaufwand für Klassifikation • keine Klassenwahrscheinlichkeiten
GMM	• Klassenwahrscheinlichkeiten	• empfindlich bezüglich Modellannahmen • empfindlich bezüglich Modenanzahl • viele Parameter • nichtkonvexe Optimierung

Vergleich von Klassifikatoren (2)

	Vorteile	Nachteile
SVM	• robust ggü. Ausreißern • konvexe Optimierung • Kernel-Trick	• hohe Komplexität • Wahl von Hyperparameter und Kernel • keine Klassenwahrscheinlichkeiten
RF	• Interpretierbarkeit • nicht-numerische Merkmale möglich	• orthogonale Entscheidungsgrenzen • keine Klassenwahrscheinlichkeiten

Vergleich von Klassifikatoren (2)

	Vorteile	Nachteile
SVM	• robust ggü. Ausreißern • konvexe Optimierung • Kernel-Trick	• hohe Komplexität • Wahl von Hyperparameter und Kernel • keine Klassenwahrscheinlichkeiten
RF	• Interpretierbarkeit • nicht-numerische Merkmale möglich	• orthogonale Entscheidungsgrenzen • keine Klassenwahrscheinlichkeiten

$\Rightarrow$ **Jeder Klassifikator hat Vor- und Nachteile, welcher am besten geeignet ist hängt von der Anwendung ab!**