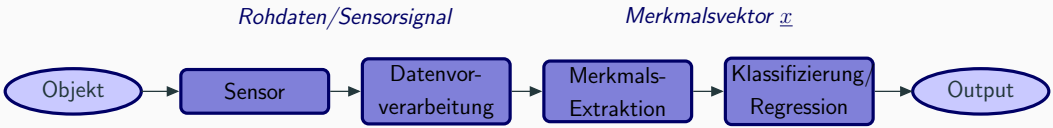
Kapitel 3 - Grundlagen für Klassifikation und Regression

Annika Liebgott

October 17, 2025

3.1 - Aufbau von ML-Systemen

Allgemeiner Aufbau von ML-Systemen



z.B. Mensch, Tier, Planze, Sprache, Temperatur, Wassertropfen, Strom,...

z.B. Kamera, Temperatur-sensor, CT, MRT, Strom-messgerät, Radar, Mikrofon,...

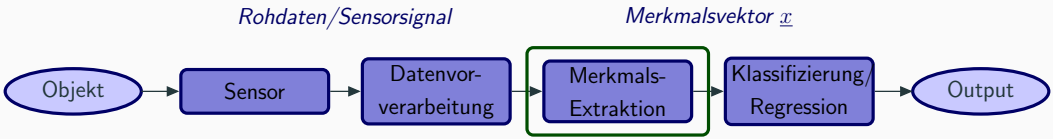
z.B. Segmen-tierung, Fil-ter, Transfor-mation, Farb-korrektur, La-beln,...

Gegebenenfalls inklusive an-schließender Merkmalsre-duktion oder -selektion

Anhand zuvor trainiertem Modell

Geschätzte Klasse bzw. Modell-Output

Allgemeiner Aufbau von ML-Systemen



z.B. Mensch, Tier, Planze, Sprache, Temperatur, Wassertropfen, Strom,...

z.B. Kamera, Temperatur-sensor, CT, MRT, Strom-messgerät, Radar, Mikrofon,...

z.B. Segmentierung, Filter, Transformation, Farbkorrektur, Labeln,...

Gegebenenfalls inklusive anschließender Merkmalsreduktion oder -selektion

Bei DL in der Regel nicht explizit nötig

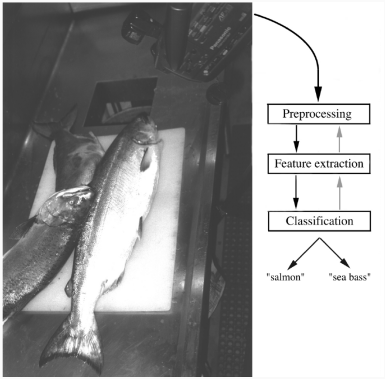
Anhand zuvor trainiertem Modell

Geschätzte Klasse bzw. Modell-Output

Beispiel 3.1: Fisch-Sortierung (2)

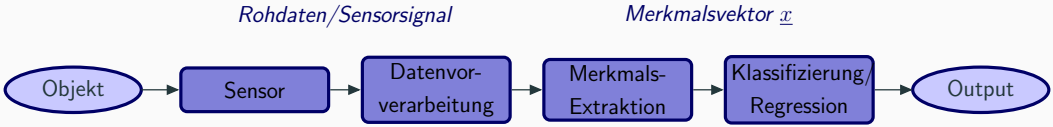
Beispiel aus dem Buch "Pattern Classification"[1]:

Fiktive Fisch-Sortieranlage, die aus Basis von Kamera-Aufnahmen zwischen gefangenen Lachsen und Seebarschen unterscheiden soll.



[1] Richard O. Duda, Peter E. Hart, David G. Stork: "Pattern Classification", Wiley-Verlag

Beispiel 3.1: Fisch-Sortierung (2)



Objekt:

Sensor:

Rohdaten:

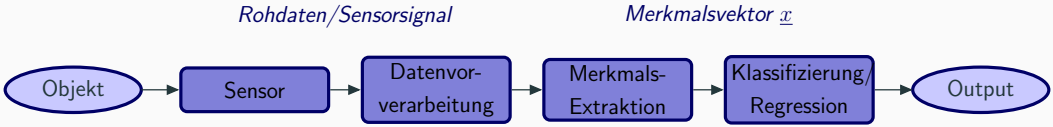
Datenvorverarbeitung:

Merkmalsextraktion:

Klassifizierung:

Output:

Beispiel 3.1: Fisch-Sortierung (2)



Objekt: Fisch unbekannter Spezies

Sensor:

Rohdaten:

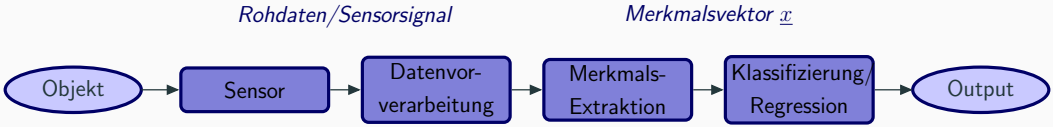
Datenvorverarbeitung:

Merkmalsextraktion:

Klassifizierung:

Output:

Beispiel 3.1: Fisch-Sortierung (2)



Objekt: Fisch unbekannter Spezies

Sensor: Kamera

Rohdaten:

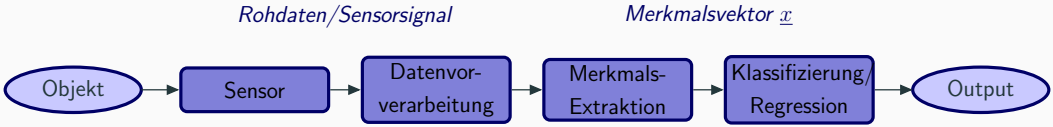
Datenvorverarbeitung:

Merkmalsextraktion:

Klassifizierung:

Output:

Beispiel 3.1: Fisch-Sortierung (2)



Objekt: Fisch unbekannter Spezies

Sensor: Kamera

Rohdaten: Bild

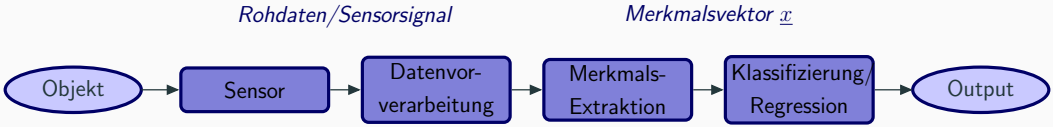
Datenvorverarbeitung:

Merkmalsextraktion:

Klassifizierung:

Output:

Beispiel 3.1: Fisch-Sortierung (2)



Objekt: Fisch unbekannter Spezies

Sensor: Kamera

Rohdaten: Bild

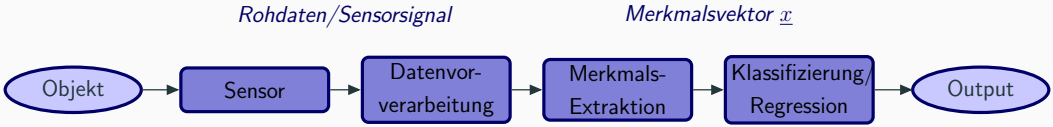
Datenvorverarbeitung: Segmentierung, Farbkorrektur, Skalierung, Rotieren,...

Merkmalsextraktion:

Klassifizierung:

Output:

Beispiel 3.1: Fisch-Sortierung (2)



Objekt: Fisch unbekannter Spezies

Sensor: Kamera

Rohdaten: Bild

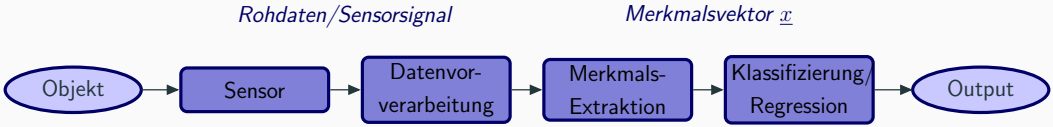
Datenvorverarbeitung: Segmentierung, Farbkorrektur, Skalierung, Rotieren,...

Merkmalsextraktion: Eigenschaften, z.B. Länge, Farbton, Breite, Anzahl Flossen,...

Klassifizierung:

Output:

Beispiel 3.1: Fisch-Sortierung (2)



Objekt: Fisch unbekannter Spezies

Sensor: Kamera

Rohdaten: Bild

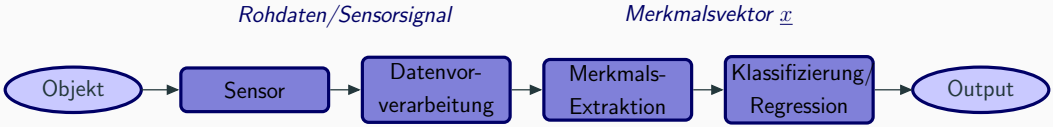
Datenvorverarbeitung: Segmentierung, Farbkorrektur, Skalierung, Rotieren,...

Merkmalsextraktion: Eigenschaften, z.B. Länge, Farbton, Breite, Anzahl Flossen,...

Klassifizierung: Binärer Klassifikator

Output:

Beispiel 3.1: Fisch-Sortierung (2)



Objekt: Fisch unbekannter Spezies

Sensor: Kamera

Rohdaten: Bild

Datenvorverarbeitung: Segmentierung, Farbkorrektur, Skalierung, Rotieren,...

Merkmalsextraktion: Eigenschaften, z.B. Länge, Farbton, Breite, Anzahl Flossen,...

Klassifizierung: Binärer Klassifikator

Output: Klassenlabel, das Spezies des Fischs repräsentiert

Klassifikation vs. Regression

Klassifikation

Das ML-System lernt, jedem Datenpunkt x aus Datenset $\mathcal{D}$ eine Klasse c, repräsentiert durch eine natürliche Zahl, zuzuordnen.

Regression

Parameter aus dem Bereich der Reellen Zahlen oder ganze Signale (also Abfolgen reeller Zahlen) werden gelernt.

3.2 - Daten: Anforderungen und Eigenschaften

Anforderungen

Frage: Welche Anforderung müssen wir an Daten stellen, um ein erfolgreiches ML-System zu implementieren?

Geeignete Daten müssen...

- repräsentativ für das zu lösende Klassifikationsproblem sein
- divers genug sein, um möglichst allgemeine Klassifikationsregeln zu lernen
- in ausreichender Menge vorliegen, um genug "Lernmaterial" zu bieten
- von Experten erstellte ("saubere") Labels besitzen (für überwachtes Lernen)

Merke:
Ein ML-System kann immer nur so gut lernen, wie die Daten sind, die ihm dafür zur Verfügung gestellt werden!

Überwachtes vs. unüberwachtes Lernen

Überwachtes Lernen (engl. supervised learning)
Neben den Datenbeispielen steht dem ML-System Informationen über die Klassenzugehörigkeit, sogenannte Labels y, zum Lernen zu Verfügung.

$\Rightarrow$ Kapitel 5

Unüberwachtes Lernen (engl. unsupervised learning)
Das ML-System erhält nur Datenbeispiele ohne Informationen über die Klassenzugehörigkeit, es lernt die Einteilung selbstständig.

$\Rightarrow$ Kapitel 6

Teilüberwachtes lernen (engl. semi-supvervised learning)
Dem ML-System steht neben einer großen Menge ungelabelter Daten auch eine kleine Menge an Daten mit Klassenlabels zur Verfügung.

3.3 - Merkmale

3.3 - Merkmale

3.3.1 - Merkmalsextraktion

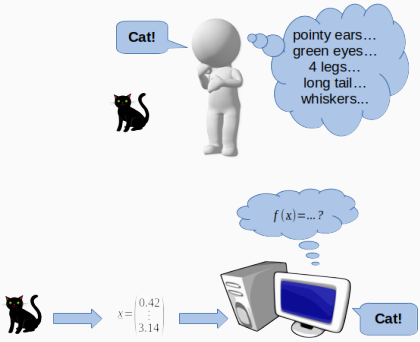
Was sind Merkmale?

Mensch:

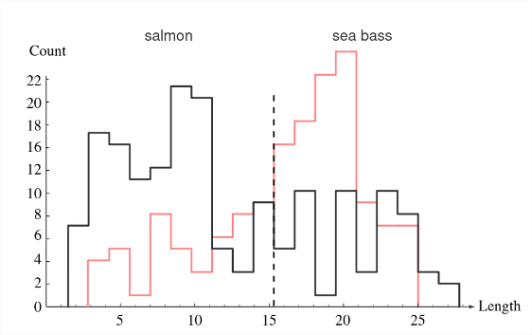
- sieht Objekt an
- erkennt signifikante Eigenschaften
- ruft Wissen über ähnliche Objekte ab
- Identifiziert Objekt basierend auf Erfahrung

Computer:

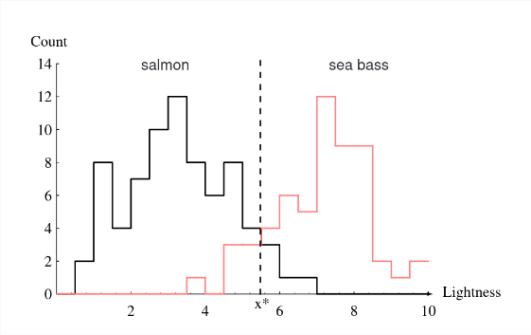
- sieht numerische Repräsentation des Objekts
- benötigt numerische Repräsentation der Eigenschaften $\Rightarrow$ **Merkmale**
- wendet Modell an, das mit Objekten gleicher Eigenschaften trainiert wurde
- identifiziert Objekt basierend auf Modell-Output



Beispiel 3.2: Fisch-Sortierung[1]



Länge des Fischs

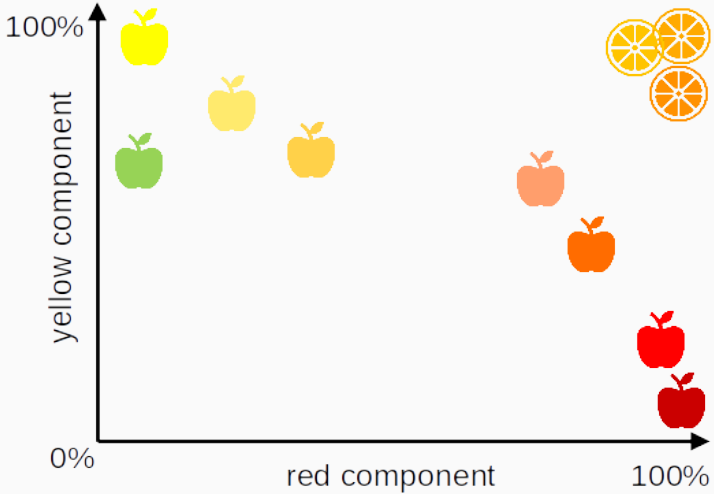


Helligkeit des Fischs

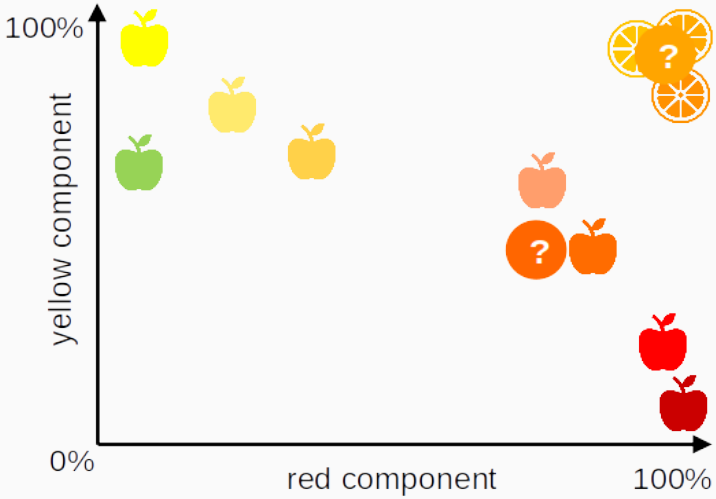
$\Rightarrow$ Keines der beiden Merkmale ist alleine gut genug geeignet, die Fisch-Spezies zu unterscheiden

[1] Richard O. Duda, Peter E. Hart, David G. Stork: "Pattern Classification", Wiley-Verlag

Beispiel 3.3: Äpfel vs. Orangen - Trainingsdaten



Beispiel 3.3: Äpfel vs. Orangen - neue Daten



Entscheidungsgrenzen, Underfitting und Overfitting

In Beispiel 3.3 lassen sich die beiden Klassen problemlos (linear) durch eine Entscheidungsgrenze trennen $\Rightarrow$ Perfekte Entscheidungsgrenze lernbar

Problem: Im Allgemeinen sind Daten in der Realität nicht vollständig trennbar, sondern überlappen sich in Randgebieten.

Was bedeutet das für den Verlauf einer guten Entscheidungsgrenze?:

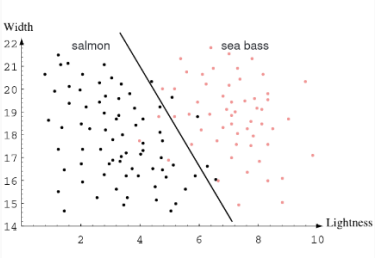
Zu trivial $\Rightarrow$ Daten werden zu schlecht getrennte $\Rightarrow$ "Underfitting"

Zu komplex $\Rightarrow$ Neue Daten nicht gut einzuordnen $\Rightarrow$ "Overfitting"

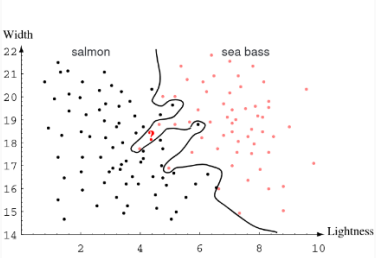
Ziel für die Entscheidungsgrenze:

Kompromiss: Weder zu einfach, noch zu komplex $\Rightarrow$ gute Generalisierung

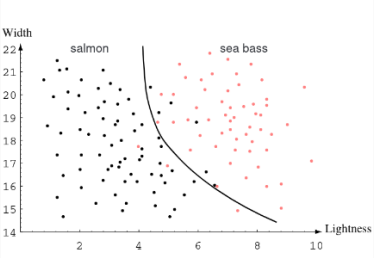
Beispiel 3.5: Fisch-Sortierung[1]



Zu einfach ⇒ Underfitting



Zu komplex ⇒ Overfitting

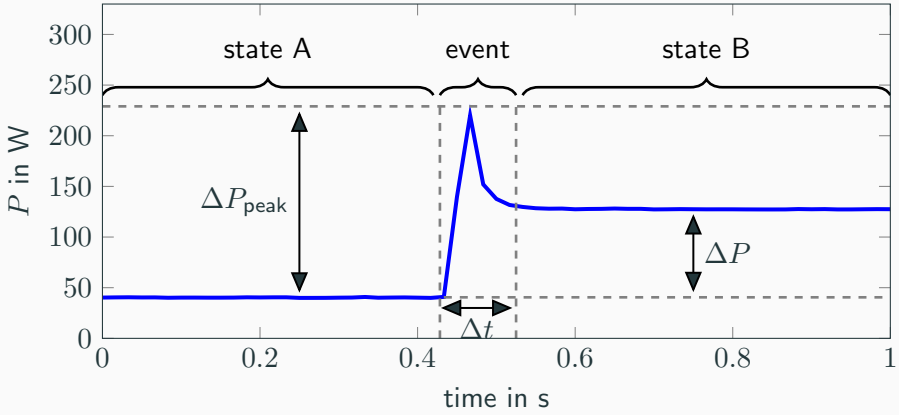


Kompromiss ⇒ gute Generalisierung

[1] Richard O. Duda, Peter E. Hart, David G. Stork: "Pattern Classification", Wiley-Verlag

Beispiel 3.6: Non-Intrusive Load Monitoring[2]

Zur Überwachung des Verbrauchs elektrischer Geräte eignen sich Merkmale, die aus der Lastkurve am Stromzähler extrahiert werden können.

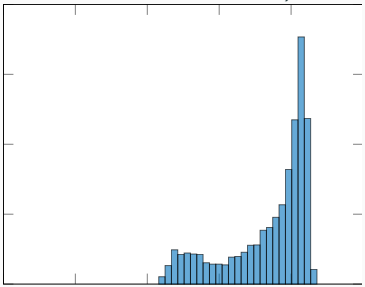


Beispiel: Unterschiede Wirkleistung, Unterschied Blindleistung, Änderungsdauer,...

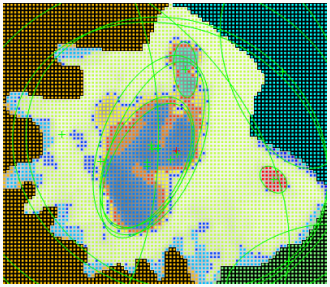
[2] F. Liebgott et al., EUSIPCO 2017

Beispiel 3.7: Charakterisierung von Lungentumoren in CT-Bildern [3]

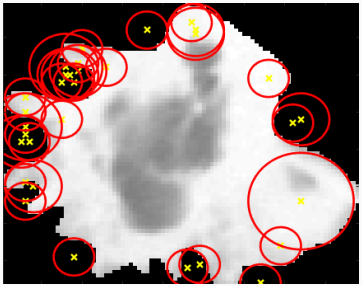
Bei Bildern bieten sich Merkmale an, die auf Variationen der Farbwerte (hier: Graustufenvarianzen) basieren.



Grauwert-Histogram



MSER



Hough-Transformation

Histogramm: z.B. Mittelwert, Kurtosis, Entropie,...

MSER: z.B. Anzahl gefundener Regionen, deren mittlere Größe, Schwerpunkt,...

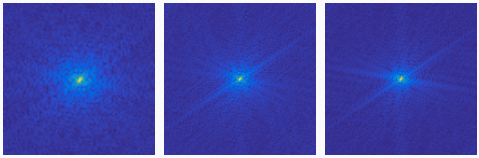
Hough-Transformation: z.B. Anzahl Kreise, maximale/minimale Durchmesser,...

[3] Liebgott et al.,IJCARS 2018

Beispiel 3.8: Unterscheidung medizinischer Bildgebungsverfahren [4]

Eine weitere Möglichkeit: Merkmale, die aus aus dem Ergebnis von Bild-Transformationen extrahiert werden.

(a) MRT **(b)** CT **(c)** PET



Fourier-Transformation

Gradientenbild

Gabor-Filter

[4] Liebgott et al.,IJCARS 2018

Skalierung von Merkmalen

Merkmale sind numerische Repräsentationen. Je nach dem, wie Merkmale berechnet werden, erstrecken sie sich über sehr unterschiedliche Zahlenbereiche.

Beispiel: Merkmal x_1 besetzt den Zahlenraum von 0 bis 1, Merkmal x_2 kann zwischen -10.000 und 10.000 liegen.

Problem: Viele Klassifikatoren skalieren die Merkmale nicht selbstständig
$\Rightarrow$ Änderungen von Merkmalen mit kleinem Zahlenraum fallen weniger ins Gewicht, als solche mit starken Unterschieden.

Lösung: Normierung aller Merkmale, beispielsweise auf bestimmten Zahlenraum (z.B. 0 bis 1) oder auf Standardnormalverteilung ($\mu = 0$, $\sigma = 1$). Letzteres wird auch z-Score genannt und häufig gewähl.

3.3 - Merkmale

3.3.2 - Merkmalsreduktion

Beispiele für Merkmalsreduktion durch Transformation

Merkmalstransformation

Merkmale werden reduziert, indem die der Merkmalsraum $\mathbb{R}^d$ in einen Raum $\mathbb{R}^{d'}$ mit $d' < d$ transformiert wird. Merkmalstransformationen sind sehr effektiv, die Bedeutung der einzelnen Merkmale geht dabei jedoch verloren.

Principal Component Analysis. Mithilfe einer Hauptachsentransformation wird ein Merkmalsraum so transformiert, dass sich korrelierte Merkmale durch ihre Linearkombinationen ausdrücken lassen.

Stochastic Proximity Embedding

Gaussian Process Latent Variable Model

Diffusion Mapping.

Isometric Feature Mapping

Beispiele für Merkmalsreduktion durch Selektion

Merkmalsselektion

Merkmale werden reduziert, indem die Optimale Untermenge der extrahierten Merkmale gesucht wird. Im Gegensatz zur Merkmalstransformation bleibt die Bedeutung der einzelnen Merkmale erhalten.

Brute-Force-Selektion. Systematisches Ausprobieren aller möglichen Merkmalskombinationen. Nur für sehr kleine d möglich, wird schnell sehr rechenintensiv und unpraktikabel.

Sequential Forward Selection und Sequential Forward Floating Selection.

Redundanzreduzierung, z.B. über Korrelationsanalysen

ReliefF

Fisher Score

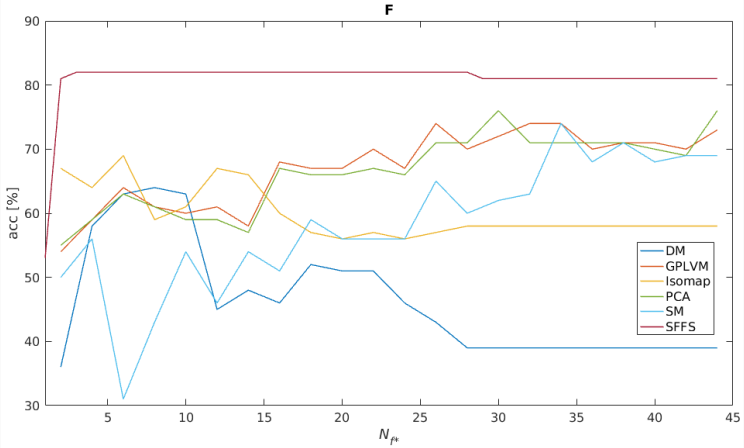
Merkmalsreduktion mithilfe von DL-Methoden

Neuronale Netze zur Feature-Selektion. Es gibt vielversprechende Ansätze, Neuronale Netze darauf zu trainieren, wichtige Merkmale aus Merkmalsraum zu selektieren.

Beispiele:

- Attention-based Feature Selection (AFS)
- Concrete Autoencoder (CAE)
- Teacher-Student Feature Selection (TSFS)
- Autoencoder-inspired feature selection (AEFS)

Beispiel 3.9: Merkmalsreduktion für Gelason-Score-Prediction[5] (1)

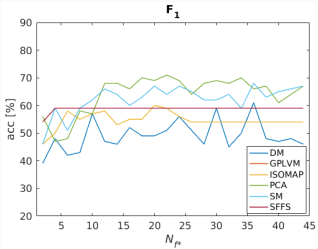


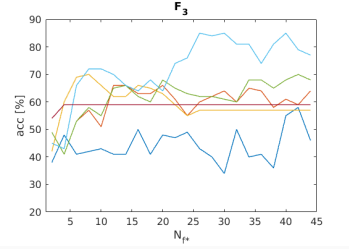
Gleason Score:
Parameter zur Risiko-Einstufung von Prostata-Tumoren

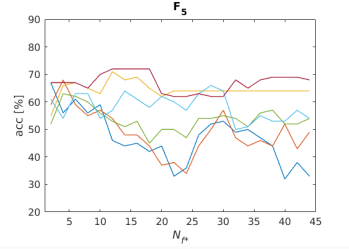
Klassifikationsaufgabe:
Anhand von Bildern des Tumors Hochrisikotumore von solchen mit niedrigem Risiko unterscheiden

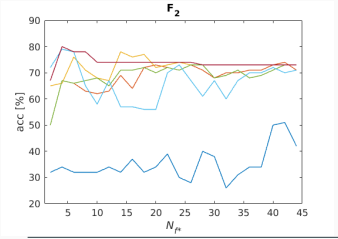
[5] Liebgot et al., ISMRM 2018

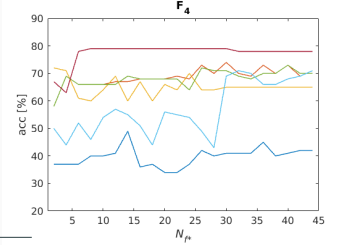
Beispiel 3.9: Merkmalsreduktion für Gelason-Score-Prediction[6] (2)

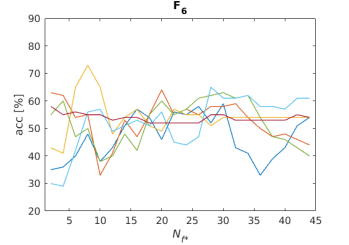












[6] Liebgot et al., ISMRM 2018

Einfluss von Merkmalen auf Kassifikator-Performance

Beispiel 3.9 zeigt deutlich:

Ohne die richtigen Merkmale funktioniert ein ML-System nicht. Dabei kommt es sowohl auf die ursprünglich extrahierten Merkmale als auch das richtige Reduktionsverfahren an.

Merke

Die angewendete Merkmalsreduktion hat großen Einfluss auf die Performance eines ML-Systems! Die Suche nach passenden Merkmalen kann sehr zeit- und kostenintensiv sein, ist aber ein notwendiger Schritt für solche Klassifikationsmodelle.

3.4 - Phasen des Trainings eines ML-Systems

Training vs. Einsatz eines Modells

Trainingsphase

Operativer Einsatz

Trainingsphasen

Die 3 Phasen eines Klassifikatortrainings:

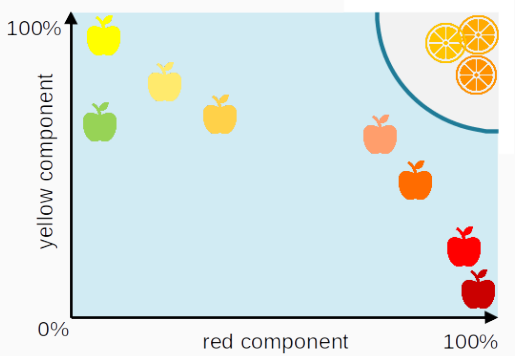
1. **Training.** Das Klassifikationsmodell wird darauf trainiert, die Daten in $\mathcal{D}_{\text{train}}$ unterscheiden zu können. Ziel ist dabei, den Trainingsfehler so gering wie möglich zu halten, ohne Overfitting zu verursachen.
2. **Test.** Das gelernte Klassifikationsmodell wird auf die bislang ungesehenen Daten in $\mathcal{D}_{\text{test}}$ angewendet. Ziel dabei ist, die Generalisierungsfähigkeit des Modells zu überprüfen.
3. **Deployment.** Das fertig trainierte Modell wird nach positiver Evaluation produktiv zur Klassifikation von neuen Daten verwendet.

Beispiel 3.10: Training und Klassifikation

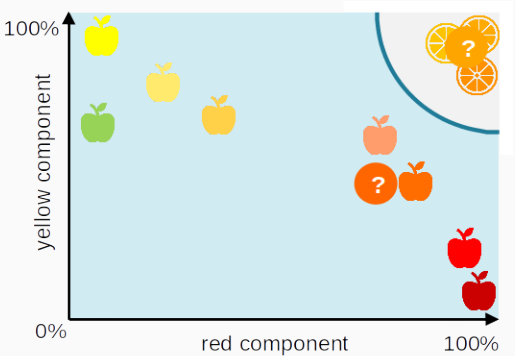
Training

Klassifikation

Beispiel 3.11: Äpfel vs. Orangen



Training



Klassifikation

Hyperparameter-Optimierung

Hyperparameter: Modell-abhängige ML-Parameter, die nicht während des Trainings optimiert werden.

Beispiele: Anzahl Entscheidungsbäume in Random-Forests, Kernelparameter einer Support Vector Machine, Anzahl gewünschter Cluster eines k-Means-Klassifikators, Anzahl Schichten und Neuronen pro Schicht in Neuronalen Netzen,...

Idealerweise: Optimierung der Hyperparameter, in der Regel mit systematischen Tests verschiedener Kombinationen (z.B. Grid Search) oder über Bayesian Optimierung, anhand eines separaten Validierungsdatensets $\mathcal{D}_{\text{val}}$ vor dem eigentlichen Training

Merke

Hyperparameter beeinflussen den Erfolg eines ML-Systems! Unterschiedliche Hyperparameterkombinationen zu testen lohnt sich meistens.

3.5 - Evaluation

Weitere Evaluationsmetriken (1)

Problem: ACC und ER gewichten FP und FN gleich stark. Es existieren aber Fälle, in denen FN schlimmer sind, als FP (und vice versa).

Beispiel 1: Krebsdiagnostik. FP führt zu weiteren Tests, Krebs kann dadurch ausgeschlossen werden (Korrektur des Fehlers). FN führt im Worst Case dazu, dass der Patient stirbt, weil die Krankheit zu lange unerkannt bleibt.

Beispiel 2: KI-basierte Verurteilung von Kriminellen. FN führt dazu, dass Kriminelle nicht verhaftet werden. FP führt dazu, dass Menschen unschuldig hinter Gittern landen. Basierend auf dem juristischen Prinzip der Unschuldsvermutung wäre letzteres schlimmer.

Lösung: Passendere Metriken zur Evaluation nutzen.

Weitere Evaluationsmetriken (1)

Precision oder Positive Predictive Value (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Sensitivity, Recall oder True Positive Rate (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Specificy oder True Negative Rate (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

F1-Score (F_1)

$$\mathrm{F}_1 = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

Beispiel 3.12: Äpfel vs. Orangen in Metriken (1)

Task: Es werden 100 Stücke Obst von einem trainierten binären Model klassifiziert.

Annahme: $N_{\text{Äpfel}} = 50$, $N_{\text{Orangen}} = 50$, $\omega_1 \widehat{=}$ Apfel, $\omega_2 \widehat{=}$ Orange

$$\mathbf{K} = \begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} = \begin{bmatrix} 35 & 10 \\ 15 & 40 \end{bmatrix} \quad \Rightarrow \quad [P_{ij}] = \begin{bmatrix} 35\% & 10\% \\ 15\% & 40\% \end{bmatrix}$$

$$\begin{aligned} \text{TNR} &= \tfrac{n_{11}}{n_{11}+n_{21}} = \tfrac{35}{50} = 0.7 \\ \text{FPR} &= \tfrac{n_{21}}{n_{11}+n_{21}} = \tfrac{15}{50} = 0.3 \\ \text{FNR} &= \tfrac{n_{12}}{n_{12}+n_{22}} = \tfrac{10}{50} = 0.2 \\ \text{TPR} &= \tfrac{n_{22}}{n_{12}+n_{22}} = \tfrac{40}{50} = 0.8 \end{aligned}$$

$$\begin{aligned} \text{ACC} &= \tfrac{\text{TP}+\text{TN}}{N} = \tfrac{75}{100} = 0.75 \\ \text{ER} &= 1 - \text{ACC} = 1 - 0.75 = 0.25 \end{aligned}$$

Beispiel 3.12: Äpfel vs. Orangen in Metriken (2)

Problem: Unausgeglichenes Datenset, $N_{\text{Äpfel}} = 90$, $N_{\text{Orangen}} = 10$

$$\mathbf{K} = \begin{bmatrix} 87 & 9 \\ 3 & 1 \end{bmatrix}$$

9 von 10 Orangen falsch klassifiziert $\Rightarrow$ schlechte Performance

Aber: $\text{ACC} = 0.88$, $\text{ER} = 0.12$ sieht gut aus

Lösung: gewichtete Metriken für realistischere Einschätzung nutzen

Balanced Accuracy (BACC) und Balanced Error Rate (BER)

$$\text{BACC} = \frac{1}{2}\left(\frac{n_{11}}{n_{11}+n_{21}} + \frac{n_{22}}{n_{12}+n_{22}}\right) = \frac{1}{2}\left(\frac{87}{87+3} + \frac{1}{9+1}\right) = 0.53$$

$$\text{BER} = \frac{1}{2}\left(\frac{n_{21}}{n_{11}+n_{21}} + \frac{n_{12}}{n_{12}+n_{22}}\right) = \frac{1}{2}\left(\frac{3}{87+3} + \frac{9}{9+1}\right) = 0.47$$

Kreuzvalidierung zur Evaluation

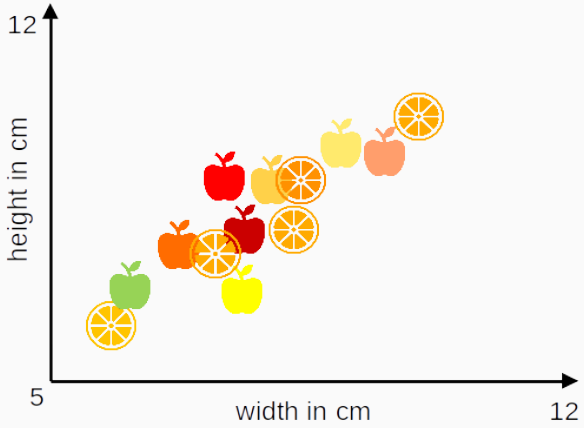
Problem: Führt man ein Klassifikator-Training einmal mit einem Datensplit in Trainings-und Testdatenset durch, besteht die Möglichkeit, dass die Performance auf Zufallseffekten basiert.

Lösung: Kreuzvalidierung, d.h. mehrfaches Trainings des Klassifikators mit unterschiedlichen Datensplits und Performance über Mittelwert und idealerweise Standardabweichung angeben.

Empfehlung:

- Bei sehr kleinen Datensets, sehr unausgeglichenen Datensets oder speziellen Anforderungen an die Klassenverteilung mehrere feste Datensplits auswählen.
- Allgemein: zufällige Datensplits wählen, idealerweise initialisiert durch Setzen von Random Seeds, um Reproduzierbarkeit zu gewährleisten.
- Je mehr Trainingsdurchläufe gemittelt werden, desto verlässlicher das Ergebnis

Typische Fehlerquelle 1: Falsche Merkmale gewählt

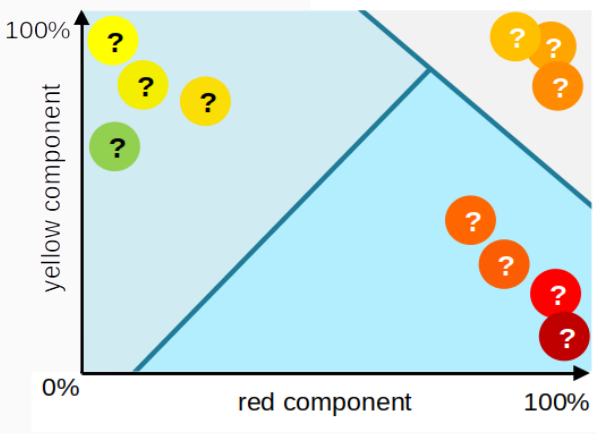


Problem:
Die Trainingsdaten befinden sich im Merkmalsraum "auf einem Haufen"

$\Rightarrow$ Klassen nicht unterscheidbar

Merke:
Ohne gute Merkmale kann ein Klassifikator keine Unterscheidung lernen!

Typische Fehlerquelle 2: Falscher Klassifikator gewählt



Problem:

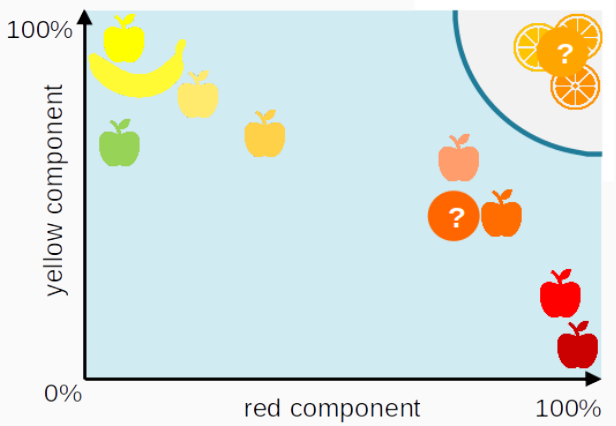
Hier: Clustering-Algorithmus (unsupervised) interpretiert Trainingsdaten falsch

$\Rightarrow$ 3 Klassen statt 2

Merke:

Der verwendete Klassifikator muss zur Datenlage passen!

Typische Fehlerquelle 3: Trainingsdaten nicht zum Problem passend gewählt



Problem:

Was passiert mit der Banane?

$\Rightarrow$ wird als Apfel klassifiziert

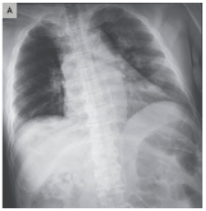
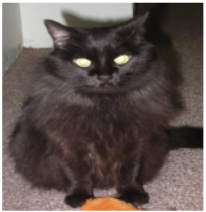
Merke:

Ein Klassifikator kann nur das, was ihm beigebracht wurde!

Beispiel 3.13: COVID-19 or Cat?[7]

Inhalt des Papers:

Klassifikator, der Lungen-CTs von Covidpatienten und Covid-freien Patienten unterscheiden sollte, bekam Katzenbilder vorgelegt. Autoren bemängeln, dass viele Modelle mit hoher Konfidenz eine beider Klassen auswählten, während ihr Modell das nicht tut

True Label	COVID-19 (Training Data)		COVID-19 (Unseen Data)		Cat (Unrelated Data)	
						
Model	**Prediction**	**Confidence**	**Prediction**	**Confidence**	**Prediction**	**Confidence**
DNN	COVID-19	99.7%	Non-COVID	75.1%	COVID-19	100%
BNN	COVID-19	95.5%	COVID-19	67.1%	COVID-19	99.8%
Ours	COVID-19	99.9%	COVID-19	69.0%	COVID-19	50.1%

[7] Paper von Mallick et al., das im Juni 2021 veröffentlich wurde und eine Kontroverse Diskussion ausgeöst hat. Schlussendlich wurde das Paper von den Autoren zurückgezogen und überarbeitet.

3.6 - Weiterführende Konzepte

3.6 - Weiterführende Konzepte

3.6.1 - Active Learning

Active Learning - Grundidee

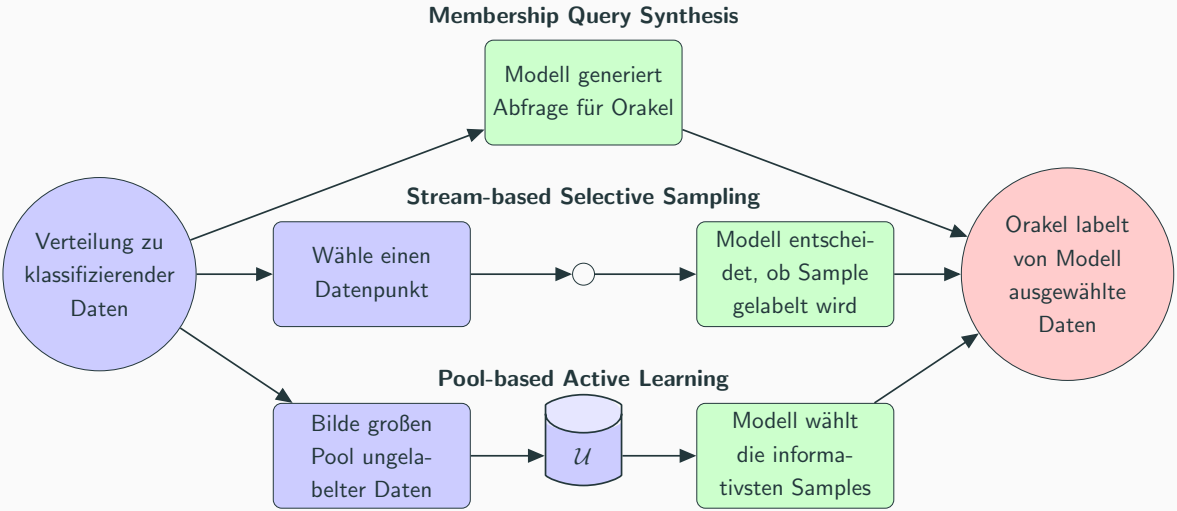
Was ist Active Learning?

Ein Konzept, um einen mit nur wenig Daten trainierten Klassifikator zu erweitern, um seine Performance zu verbessern. Zentraler Punkt: die Auswahl von geeigneten Samples, die für das weitere Klassifikatortraining verwendet werden sollen und einem sog. "Orakel" zum Labeln vorgelegt werden.

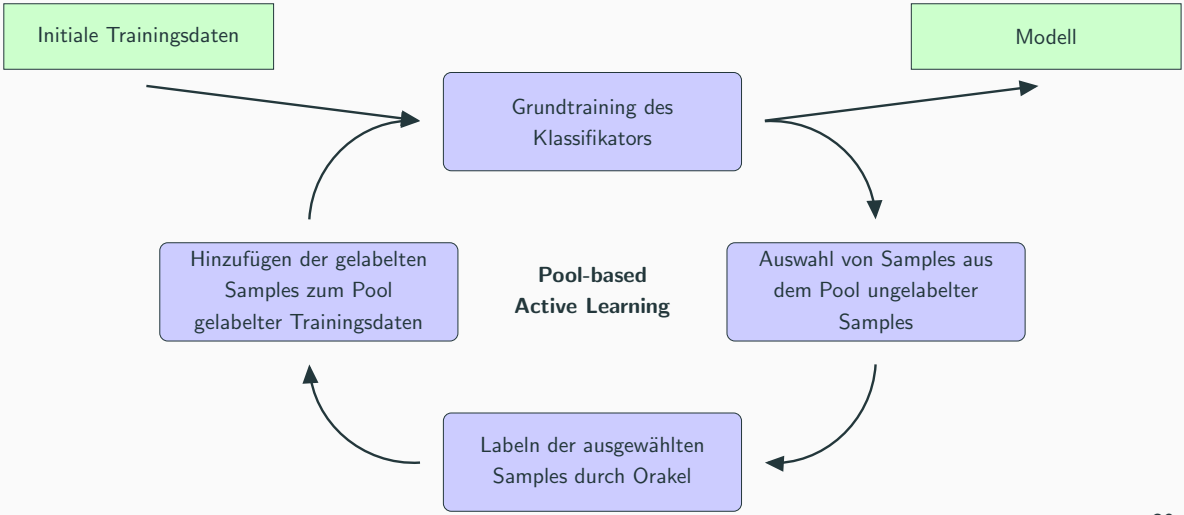
Wofür kann Active Learning eingesetzt werden?

- Die Menge benötigter Trainingsdaten reduzieren
- Den Labeling-Aufwand bei überwachtem
- Vermeiden von Redundanzen im Trainingsdatenset
- Anpassen von Klassifikatormodellen an sich ändernde Daten (z.b. Hinzufügen zusätzlichen Klassen in Online-Systemen)

3 Formen von Active Learning

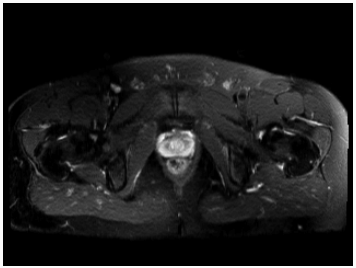
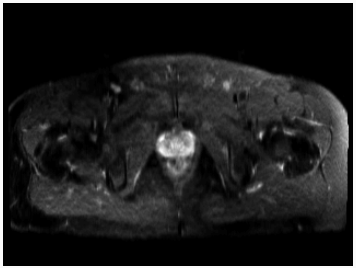
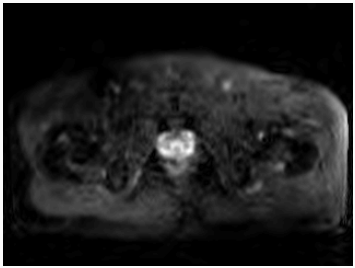


Pool-based Active Learning (Grundprinzip)



Beispiel 3.14 Automatische Qualitätsbewertung von MRT-Bildern (1)

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

Ziel des Active Learning: Reduktion der benötigten Trainingsdaten

[8] Liebgott et al., ICASSP 2016

Beispiel 3.14 Automatische Qualitätsbewertung von MRT-Bildern (2)

Klassifikations-Setup:

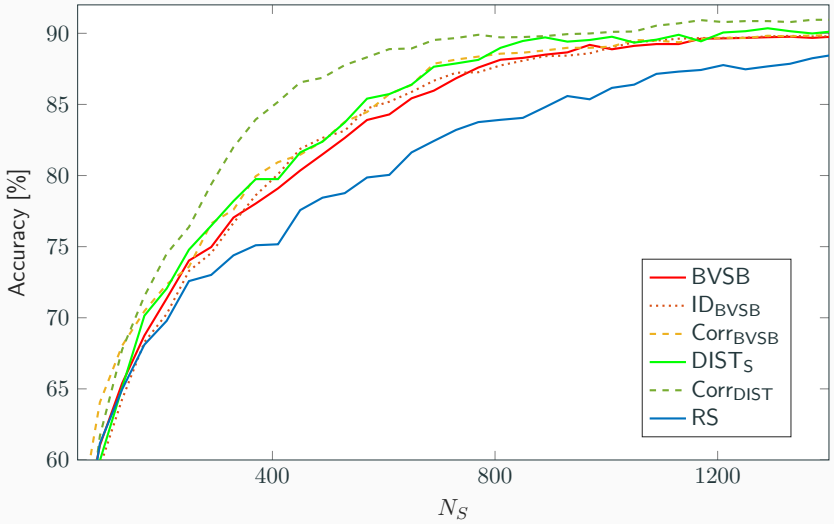
- Input-Daten: 2911 2D-Bilder von 100 Patienten, gelabelt durch Radiologen
- 2871 Merkmale basierend auf Graustufenvarianzen
- Merkmalsreduktion: PCA
- Klassifikator: Soft-Margin Multi-Class Support Vector Machine
- 70 % Trainings-, 30 % Testdaten

Baseline-Accuracy bei Training mit dem gesamten Datenset: $\mathrm{ACC} = 90\,\%$

Auswahl der neuen Trainingsdaten aus dem Pool ungelabelter Samples:

- Klassen-Wahrscheinlichkeiten
- Distanz zur Entscheidungsgrenze

Beispiel 3.14 Automatische Qualitätsbewertung von MRT-Bildern[9] (3)



Ergebnisse nach einigen Iterationen mit 40 Samples, die pro Iteration zum Trainingsdatenset hinzugefügt wurden

N_S: Gesamtzahl Samples im Trainingsdatenset

[9] Liebgott et al., Annual Meeting of the ISMRM, 2018

3.6 - Weiterführende Konzepte

3.6.2 - One-Class-Klassifikation

One-Class-Klassifikation - Grundidee

Unterschied zu traditioneller Klassifikation:

Ziel des Trainings ist nicht, zwischen Samples verschiedener Klassen zu Unterscheiden, sondern Samples einer Klasse zuverlässig zu erkennen.

Herangehensweise:

Klassifikator wird primär mit Samples der gewünschten Klasse trainiert. Zur Verfeinerung der Entscheidungsgrenzen gibt es auch Ansätze, bei denen eine geringe Menge relevanter Out-of-Class-Samples im Training verwendet werden.

One-Class-Klassifikation ist prinzipiell mit verschiedenen Klassifikatoren möglich

Zweck von One-Class-Klassifikation

Wann ist One-Class-Klassifikation sinnvoll?

- Wenn nur wichtig ist, eine Klasse zu identifizieren und die Klassen aller anderen Samples egal sind
- Wenn es während des Trainings keine oder wenige (gelabelte) Samples anderer Klassen gibt
- Wenn das Labeln von anderen Klassen hohe Kosten verursacht
- Wenn die anderen Daten variabel sind, beispielsweise mit der Zeit neue Klassen hinzukommen

Beispiel: Outlier-Detektion, Anomalie-Detektion, Novelty-Detektion

Anwendungsbeispiele für One-Class-Klassifikation

Umgebungsunabhängige Objektdetektion: Beispielsweise Gesichtsdetektion, die unabhängig vom Ort der Aufnahme erfolgen soll.

Zustandsüberwachung: Beispielsweise Erkennung von bestimmten Fehlfunktionen in der Produktion, wenn Maschinenparameter sich außerhalb ihres Normbereichs bewegen.

Zustandsdetektion: Beispielsweise erkennen von Werkstücken mit Produktionsfehlern oder von Auffälligkeiten in Gesundheitsdaten (z.B. radiologische Bilder, Blutwerte)

3.6 - Weiterführende Konzepte

3.6.3 - Reinforcement Learning

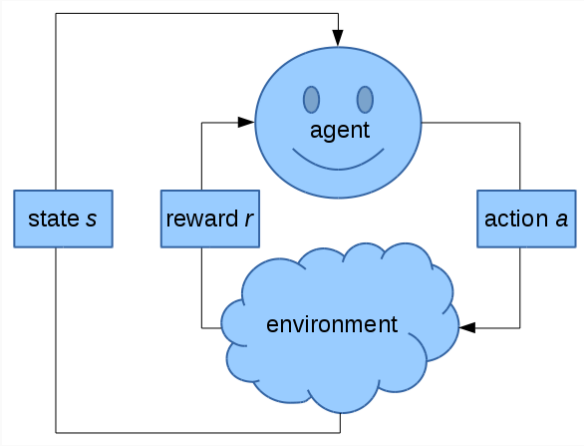
Reinforcement Learning - Grundidee

Hintergrund:

Ziel der Methode ist es, ein Modell iterativ angelehnt an das verhaltens-psychologischen Konzepts der positiven Verstärkung zu trainieren.

Vorgehensweise:

- Ein sog. Agent wählt eine von N möglichen Aktionen aus
- Durch die Aktion interagiert er mit seiner Umgebung und erhält bei guter Auswahl eine Belohnung
- Agent strebt an, die Belohnung zu maximieren



Anwendungsbeispiele für Reinforcement Learning

Einige mögliche Anwendungen:

- Optimierung von Ressourcen-Zuweisung in Computer-Clustern [10]
- Optimierung von Ampelanlagen [11]
- Verschiedene Tasks im Bereich Robotics[12]
- KI in PC-Games[13]
- ...

[10] z.B. Mao et al.: "Resource Management with Deep Reinforcement Learning"
[11] z.B. Arel et al.: "Reinforcement learning-based multi-agent system for network traffic signal control"
[12] Eine Übersicht: Kober et al: "Reinforcement Learning in Robotics:A Survey"
[13] z.B. Silver et al.: "Mastering the game of Go without human knowledge"

Aufgabe: Anforderungen an medizinisches ML-System

In einem Krankenhaus soll ein neues System eingeführt werden, das mithilfe von KI-basierter Computer-Aided Diagnostic (CAD) Radiologen bei ihren diagnostischen Aufgaben unterstützt. Während die Radiologen sich beim Betrachten von MRT-Aufnahmen auf eine spezielle medizinische Fragestellung konzentrieren, soll das CAD-System die automatisiert die Aufnahmen auf eventuelle Anomalien untersuchen.

Aufgabe: Was muss bei der Entwicklung und dem Training eines solchen Systems bedacht werden im Hinblick auf

a) die zu verwendenden Trainingsdaten?

b) die Definition der Aufgabenstellung für das CAD-System?

c) die Evaluation des ML-Modells?

d) den praktischen Einsatz des CAD-Systems?