Machine Learning

Allgemeines

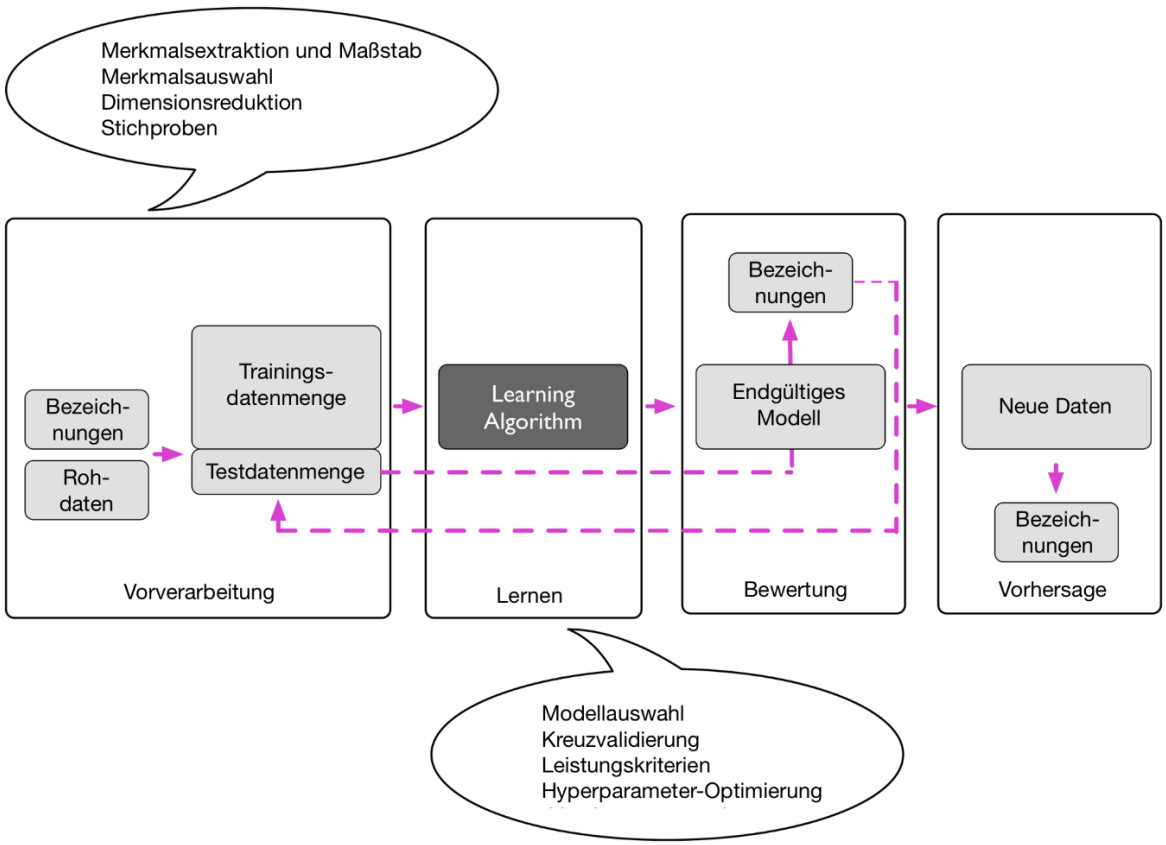
Was ist ML?

Wissenschaft von Algorithmen, die ohne menschliches Eingreifen (Ableitung von Regeln, Erstellung von Modellen) den Sinn von Daten erkennen.

Arten von ML

1) Überwachtes Lernen
   - gekennzeichnete Daten
   - direktes Feedback
   - Methoden
     - Klassifizierung
     - Regression (Voraussage stetiger Werte)
2) Verstärkendes Lernen
   - Agent, der seine Leistung durch Interaktion mit seiner Umgebung verbessert
   - Belohnungssystem
3) Unüberwachtes Lernen
   - keine Kennzeichnung
   - verborgene Strukturen finden
   - Methoden
     - Clustering - *unüberwachte Klassifizierung*
     - Datenkomprimierung

Ablauf



Datenvorbereitung

Entscheidende Faktoren für Datensätze sind Umfang, Auswahl und Qualität

1) Unvollständige Daten **entfernen** oder **ergänzen**
   - Daten mit fehlenden oder falschen Daten führen zu unvorhersehbaren Ergebnissen
   - stetige Werte könnten durch Interpolation ergänzt werden
2) **Kodierung** kategorialer Daten (Algorithmen benötigen numerische Werte)
   - Dictionary
     - Spaltenanzahl bleibt gleich; neue Zahl pro Wert
     - eignet sich für geordnete (*ordinale*) Werte, Mensch muss Ordnung festlegen
   - Label Encoder
     - Spaltenanzahl bleibt gleich; neue Zahl pro Wert
     - eignet sich für Klassenbezeichnungen
   - One-Hot-Codierung
     - neue Spalte pro Wert
     - eignet sich für bezeichnende (*nominale*) Werte
3) **Trennung** in Trainings- und Testdatensatz
4) Daten normieren oder **standardisieren**
   - Normierung: Abbildung auf das Intervall [1, 1]
   - Standardisierung: Zentrieren auf 0 mit Standardabweichung von 1
     - besser für Ausreißer und ungewöhnliche Werte
5) **Regularisierung**, um Ausreißer zu bestrafen
6) Geeignete **Merkmale auswählen** um die Komplexität zu reduzieren und Überanpassung zu verhindern
   - Dimensionsreduzierung
   - L1 Regularisierung
   - SBS Algorithmus
   - RandomForest

Datenkomprimierung

Mithilfe der Datenkomprimierung sollen Merkmale zusammengefasst werden, indem diese in einen neuen Merkmalsraum transformiert werden.

Principal Component Analysis

Mithilfe des PCA wird ein Untermerkmalsraum mit maximaler Varianz entlang der orthogonalen Merkmalsachse konstruiert. Es werden Kovarianzmatrizen konstruiert, und durch Zerlegung in Eigenwerte und Vektoren die Vektoren mit den größten Eigenwerten selektiert. Daraus wird eine Projektionsmatrix erzeugt, mit der ein neuer Merkmalsunterraum betrachtet werden kann (weniger Merkmale als vorher).

- unüberwachtes Verfahren
- erhöht Effizienz der Berechnung
- reduziert Rauschen

Linear Discriminant Analysis

LDA versucht einen Merkmalsunterraum zu finden, der die Trennbarkeit der Klassen optimiert (maximale Separierbarkeit). Dafür werden Mittelwertvektoren und zwei resultierende Streumatrizen berechnet (innerhalb S_W und zwischen S_B den Klassen). Anschließend erneute Eigenwertzerlegung mit $S_W^{-1}S_B$.

- überwachtes Verfahren (Klassenzugehörigkeit)
- erhöht Effizienz der Berechnung

Kernel PCA

Ziel des Kernel PCAs ist es, nicht-linear trennbare Daten in neue, linear trennbare Daten umzuwandeln. Berechnung einer zentrierten Kernel-Matrix für jedes Element der Datenmenge um anschließend die Eigenwerte und Eigenvektoren bestimmen zu können. Die Konstruktion eine Projektionsmatrix ist nicht notwendig.

1) Transformation der Aktivierungsfunktion ϕ in höherdimensionierten Raum
2) finden einer linearen Hyperebene
3) Rücktransformation

Klassifierzungs-Algorithmen

- Perzeptron
- Adaline
- Logistische Regression
- SVM
- Entscheidungsbäume
- Random Forest
- kNN
- Optimierungen
- Modellbewertung

Perzeptron

Mathematische Grundlagen

- binäre Klassifizierung (positiv (+1) und negativ (-1))
- Gewichtungsvektor **w** und Eingabevektor **x**
- Aktivierungsfunktion $\phi(z)$ mit $z = w_0x_0 + w_1x_1 + \cdots + w_ix_i$

$$w_0 = -\Theta$$
$$x_0 = 1$$

$$\phi(z) = \begin{cases} 1 & \text{wenn } z \geq 0 \\ -1 & \text{andernfalls} \end{cases}$$

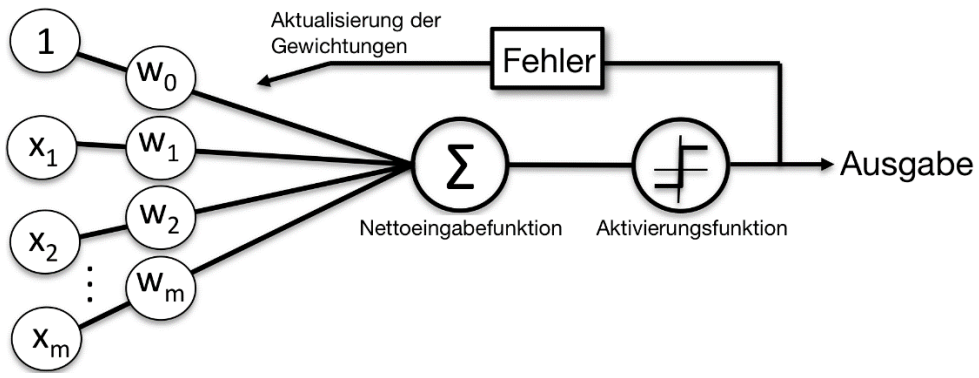
Ermittlung von **w**:

- Aktualisierung der Gewichte: $\mathbf{w_j} \coloneqq \mathbf{w_j} + \mathbf{\Delta w_j}$ (bei jedem Objekt)
  - mit $\mathbf{\Delta w_j} = \boldsymbol{\eta}(\boldsymbol{y^i} - \boldsymbol{\hat{y}^i})\boldsymbol{x_j^i}$
  - Die Aktualisierung der Gewichte erfolgt bei jedem Objekt
- $\boldsymbol{\eta}$: Lernrate (typischerweise zwischen 0 und 1)
- $\boldsymbol{y^i}$: tatsächliche Klassenbezeichnung
- $\boldsymbol{\hat{y}^i}$: vorhergesagte Klassenbezeichnung

Parameter

- **eta0**: Lernrate (Größe der Schritte)
- **max_iter**: Anzahl der Epochen

Ablauf / Prozess



Adaline (ADAptive LInear NEuron classifier)

Lineare Aktivierungsfunktion anstelle von einer Sprungfunktion. Gradientenabstiegsverfahren einer Straffunktion J

Mathematische Grundlagen
Straffunktion als Summe der quadratischen Abweichungen:

$$J(w) = \frac{1}{2} \sum_i \left(y^i - \phi(w^T x^i)\right)^2$$

Die Änderung $\Delta w = -\eta \nabla J(w)$ ergibt sich aus dem Gradienten, also der partiellen Ableitung von J(w):

$$\frac{\partial J}{\partial w_j} = \ldots = -\sum_i \left(y^i - \phi(z^i)\right) x_j^i$$

Aktualisierung der Gewichte w := w + Δw

- mit $\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = \eta \sum_i \left(y^i - \phi(z^i)\right) x_j^i$
- Die Aktualisierung erfolgt per Stapelverarbeitung nach Berechnung des Gradienten nur einmal

Parameter

- `eta0`: Lernrate (Größe der Schritte)
- `max_iter`: Anzahl der Epochen

Logistische Regression

Ähnlich zu Adaline, anstelle einer linearen Aktivierungsfunktion wird eine Sigmoid-Funktion verwendet. Es kann eine Aussage darüber getroffen werden, wie wahrscheinlich es ist, dass ein Merkmal x zu einer Klasse y gehört. Stochastisches Gradientenabstiegsverfahren ist möglich.
Mathematische Grundlagen
Sigmoid-Funktion

$$\phi(z) = \phi(w^T x) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-w^T x}}$$

als Aktivierungsfunktion ϕ(z).

Neue Straffunktion als Likelihood L bzw. besser: Log-Likelihood.

$$J(w) = -l(w) = log(L(w))$$

$$log(L(w)) = \sum_{i=0}^{n} y^i \log(\phi(z^i) + (1 - y^i) \log(1 - \phi(z^i)$$

Die Änderung $\Delta w = -\eta \nabla J(w)$ erfolgt erneut mit der partiellen Ableitung der Straffunktion J.

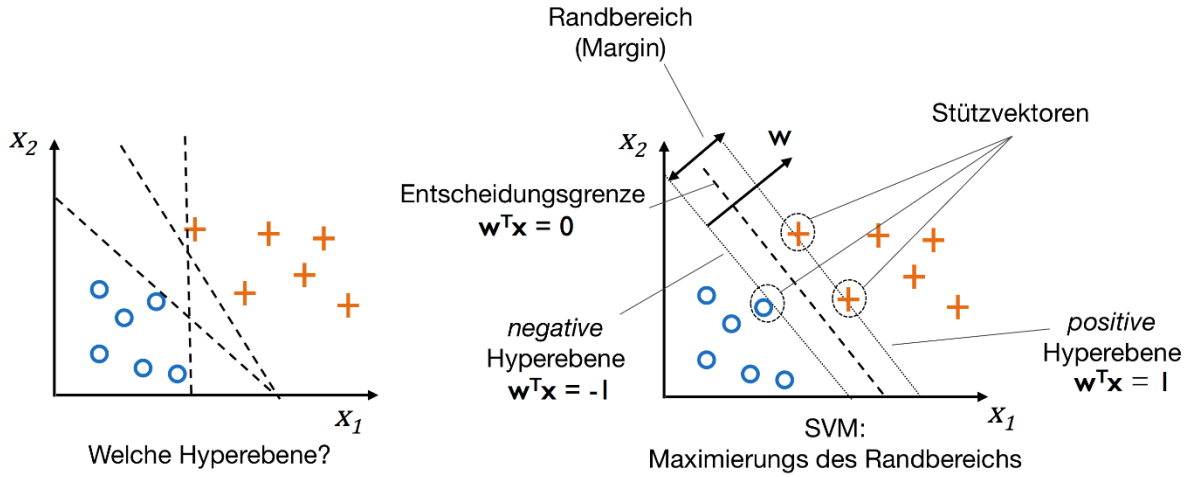
- Aktualisierung der Gewichte w := w + Δw
  - mit $\Delta w = \eta \sum_{i=1}^{n} \left(y - \phi(z^i)\right) x^i$
  - falsche Vorhersagen werden härter bestraft

Parameter

- `max_iter`: Anzahl der Epochen
- `solver`: z.B. `'lbfgs'`
- `penalty`: Regularisierung z.B. `l2`

Support Vector Machine

Methode, um den Abstand (*margin*) zwischen Klassengrenzen zu maximieren.



Auch nicht-lineare Entscheidungsgrenzen sind realisierbar.

Mathematische Grundlagen

Einteilung in

- Entscheidungsgrenze
- positive Hyperebene
- negative Hyperebene

Nicht-linear trennbare Entscheidungsgrenzen

1) Transformation der Aktivierungsfunktion ϕ in höherdimensionierten Raum
2) finden einer linearen Hyperebene
3) Rücktransformation

- Alternativ mithilfe eines Kernel-Tricks (**R**adial **B**ase **F**unction-Kernel)

Parameter

- `max_iter`: Anzahl der Epochen
- `C`: Regularisierung (niedrig = starke Regularisierung)
- `kernel`: z.B. `'rbf'` für nicht-lineare Entscheidungsgrenzen
- `gamma`: Als Parameter des veränderten Kernels

Entscheidunsbäume

Durch Treffen von Entscheidungen werden Grenzen gezogen (keine Funktion/Linie, sondern Stufen). Reelle Zahlen werden zu Entscheidungsfragen umgewandelt.

- Aufteilung der Daten, sodass der größte Informationsgewinn entsteht
- Wiederholen der Aufteilungsprozedur pro Knoten bis zu einem Blattknoten
- Tiefe des Baumes sollte begrenzt sein (Extremfall: jedes Trainingselement hat eigenen Blattknoten)

Mathematische Grundlagen

Ziel ist es, den Informationsgewinn IG zu berechnen.

- Setzt sich aus den Daten des Elternknotens und des linken und rechten Kindes zusammen
- *Impurity*-Funktionen zur Auswahl
    - Entropie I_H (mittlerer Informationsgehalt einer Nachricht)
    - Gini-Koeffizient I_G (Reinheit, Minimierung der Wahrscheinlichkeit einer Fehlklassifizierung)
    - Klassifizierungsfehler I_E (Stutzen eines Entscheidungsbaumes, Pruning)

Parameter

- `criterion`: *Impurity*-Funktion (Entropie, Gini, Klassifizierungsfehler)
- `max_depth`: Tiefe eines Entscheidungsbaums.

Random Forest

Ensemble von Entscheidungsbäumen. Nutzen mehrerer Entscheidungsbäume mit (jeweils) großer Varianz. Führt zu einem stabileren Modell, dass weniger anfällig für Überanpassung ist.

Ablauf

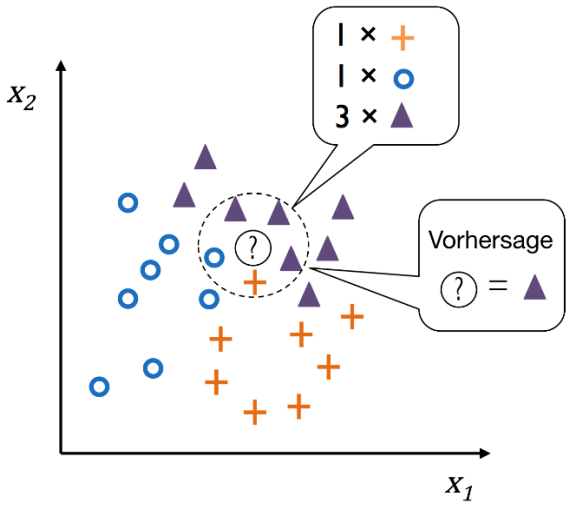
1) Wähle eine zufällige Stichprobe der Größe n aus den Trainingsdaten aus
2) Konstruiere einen Entscheidungsbaum
    a. Wähle zufällig d Merkmale aus
    b. Teilen sie den Knoten auf (anhand der Zielfunktion (*impurity*))
3) Wiederhole 1 und 2 k mal
4) Fasse die Vorhersagen der einzelnen Bäume durch Mehrheitsentscheidung zusammen

Parameter

- `criterion`: *Impurity*-Funktion
- `n_estimator`: kk: Anzahl der Entscheidungsbäume (z.B. Größe der Trainingsdaten)

kk-Nearest Neighbours

Zuweisung der Klassenbezeichnung durch Mehrheitsentscheidung anhand eines definierten Abstandsmaßes. Eine Standardisierung ist in jedem Fall notwendig



Parameter

- `metric`: Metrik z.B. `'minkowski'`
- `p`: `p=1` - Euklid-Abstand, `p=2` - Manhatten-Metrik

Optimierungen für Klassifizierung-Algorithmen

- Standardisierung
- Stoch. Gr. Desc.
- Überanpassung
- Regularisierung

Merkmal-Standardisierung

Verschieben der Merkmale auf den Mittelwert 0 durch eine Skalierung der Merkmale mittels μ und σ, sodass die Standardabweichung 1 ist.

$$x'_j = \frac{x_j - \mu_j}{\sigma_j}$$

Auch mithilfe eines `StandardScalers` möglich. Dabei gibt es `transform` und `inverse_transform` um die ursprünglichen Werte zu erhalten.

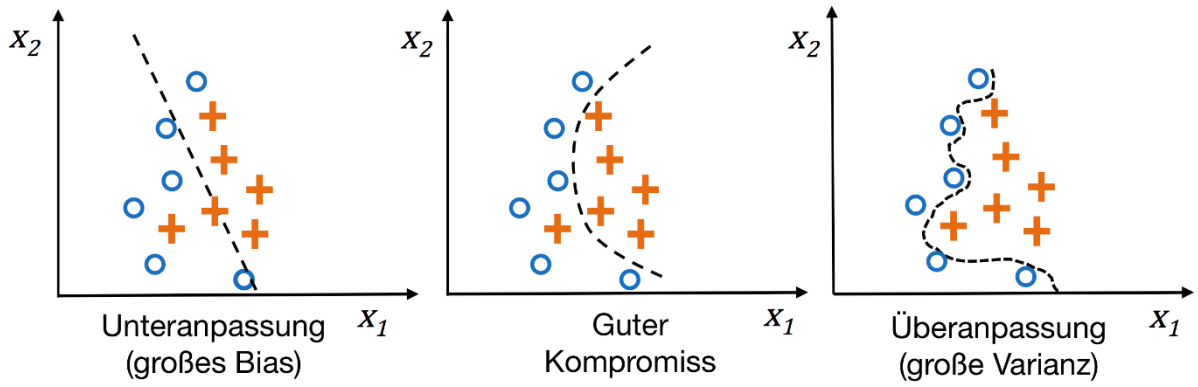
Stochastisches-Gradientenabstiegsverfahren

Gewichtung inkrementell für jedes Objekt aktualisieren.

- konvergiert schneller
- nur Näherung des normalen Gradientenabstiegsverfahrens
- höheres statistisches Rauschen
- Daten müssen nicht vorweg vorliegen, können inkrementell hinzugefügt werden

Überanpassung / Unteranpassung

- Überanpassung / Unteranpassung verfälscht die Vorhersage eines Modells und soll vermieden werden.



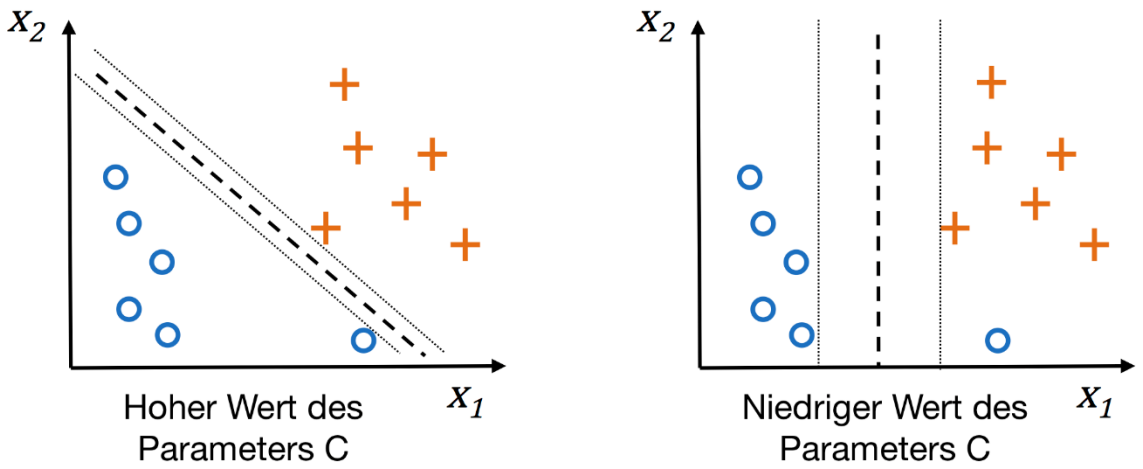
Regularisierung

Verhinderung von Überanpassung, indem extreme Parameter (Gewichte) durch einen zusätzlichen Bias bestraft werden. Besonders relevant für `LogisticRegression` und `Support-Vector-Machines`.
Einführung eines Regularisierungsparamters λ bzw.

$$C = \frac{1}{\lambda}.$$

- kleines C: starke Regularisierung
- großes C: starke Anpassung

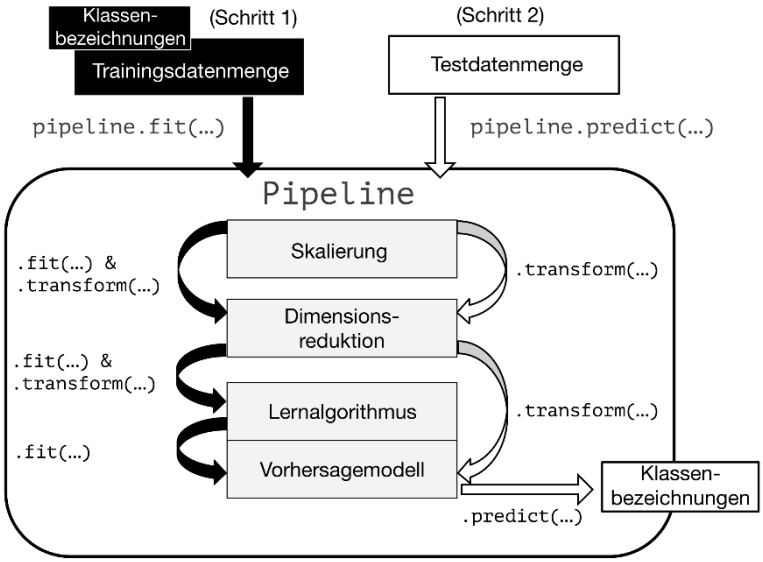
Beispiel für SVM:



Optimierung von Arbeitsabläufen

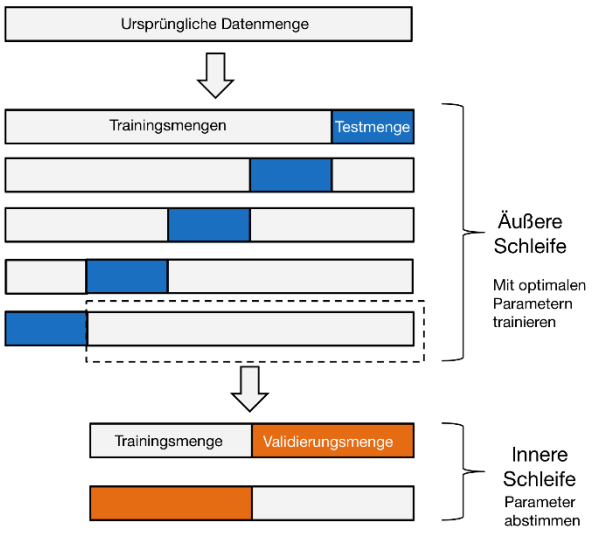
Durch Pipelines

- Aufruf von `fit`
  1) Scaler wird ausgeführt + Übergabe
  2) PCA wird ausgeführt + Übergabe
  3) Schätzer wird ausgeführt
- Aufruf von `predict`
  1) Scaler wird ausgeführt + Übergabe
  2) PCA wird ausgeführt (mit Daten aus `fit`) + Übergabe
  3) Schätzer wird ausgeführt + Vorsage berechnet



Kreuzvalidierung um Unter- und Überanpassung zu vermeiden

- k-fache Kreuzvalidierung in äußerer Schleife
  1) Aufteilung in Trainings, Testdaten zum Trainieren
  2) Wiederholen bei Neuaufteilung der Datensätze (jede Teilmenge einmal Trainings- bzw. Testdaten)
  3) Berechnung des Durchschnitts der Leistungseinschätzung
- j-fache Schleife zur Auswahl des Modells (Überprüfung mit Validierungsdaten)



Feinabstimmung von Hyperparametern durch Rastersuche (Grid Search)

- Bewertung für verschiedene Parameterkombinationen

Parameter:

- `estimator`: Pipeline
- `param_grid`: Dictionary für Hyperparameter-Kombinationen
- `cv`: Kreuzvalidierung

Bewertung des Modells anhand einer Wahrheitsmatrix (Konfusionsmatrix)

- Bestimmung von *true negatives* und *false positives*

Korrektklassifizierungsrate:

$$\frac{\text{Richtige Vorhersagen}}{\text{Alle}}$$

Fehlerquote:

$$\frac{\text{Falsche Vorhersagen}}{\text{Alle}}$$

Genauigkeit:

$$\frac{\text{TP}}{\text{TP} + \text{FP}}$$

Trefferqoute:

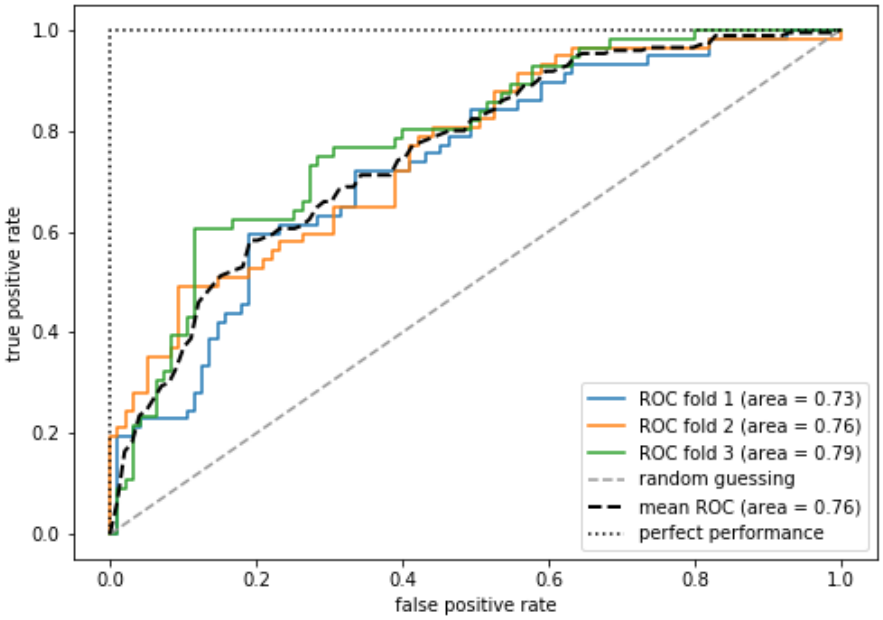
$$\frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1-Maß:

$$F1 = \frac{2 * GEN * TQ}{GEN + TQ}$$

ROC Diagramme (Receiver-Operating-Characteristic)

- Tool zur Auswahl von Modellen anhand der *true negatives* und *false positives*



Unausgewogene Klassenverteilung

- Härtere Bestrafung bei falschen Vorhersagen der selteneren Klassen
- Up- oder Down Sampling um den Anteil auszugleichen

Regressions-Algorithmen

Teil des überwachten Lernens zur Vorhersage eines numerischen, stetigen Wertes anstelle einer Klassenzugehörigkeit.

- Lineare Regression
- Polynomale Regression
- Enscheidungsbäume / Random-Forest
- Optimierung
- Bewertung

Lineare Regression

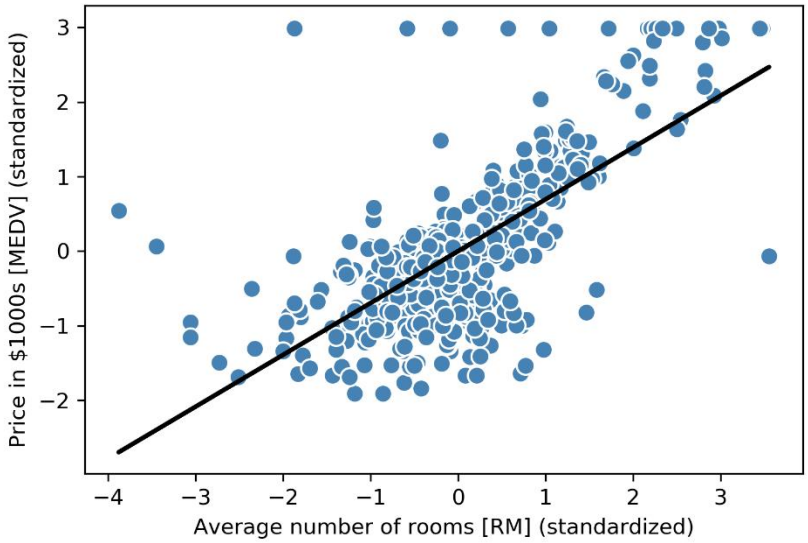
Suche nach einer Geraden (*Regressionsgrade*), die am Besten zu der Punktwolke passt.

Korrelationsmatrix

- neu skalierte Version der Kovarianz-Matrix (eng verbunden mit PCA, bei standardisierten Werten gleich)
- trifft eine Aussage darüber, ob eine Korrelation vorliegt (nicht nur linear)

Berechnung der Geraden mithilfe der Methode der kleinsten quadratischen Abstände

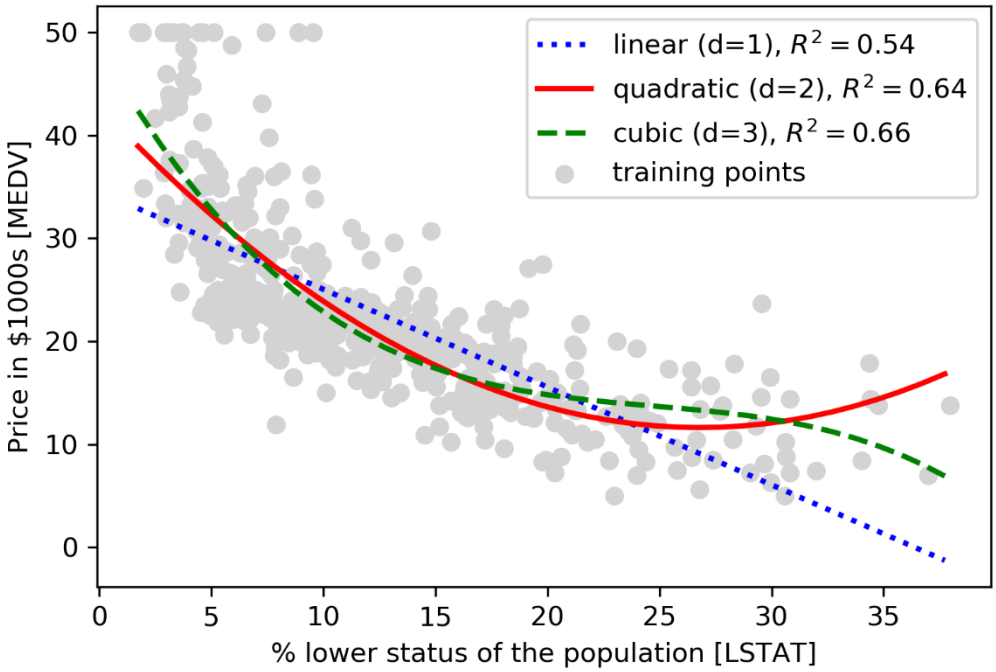
$$J(w) = \frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$



Polynomale Regression

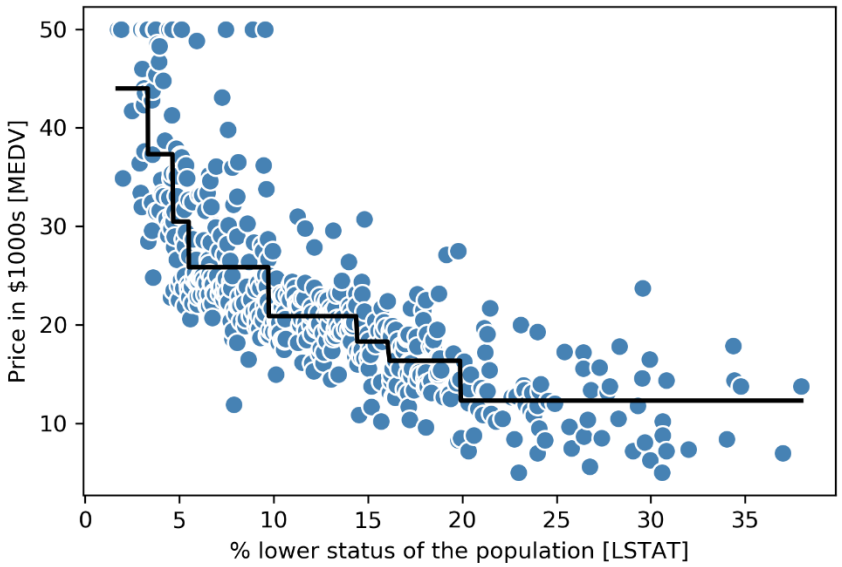
Bilden von Polynomen des Grades d.

$$y = w_0 + w_1 x_1 \rightarrow y = w_0 + w_1 x + w_2 x^2 + \cdots + w_d x^d$$



Entscheidungsbäume / Random-Forest

Mithilfe von Entscheidungsbäumen oder des Random-Forest über die Maximierung des Informationsgewinn IG oder die Minimierung der Unreinheit stetige Vorhersagen treffen.



Optimierung

Behandlung von Ausreißern ist schwer

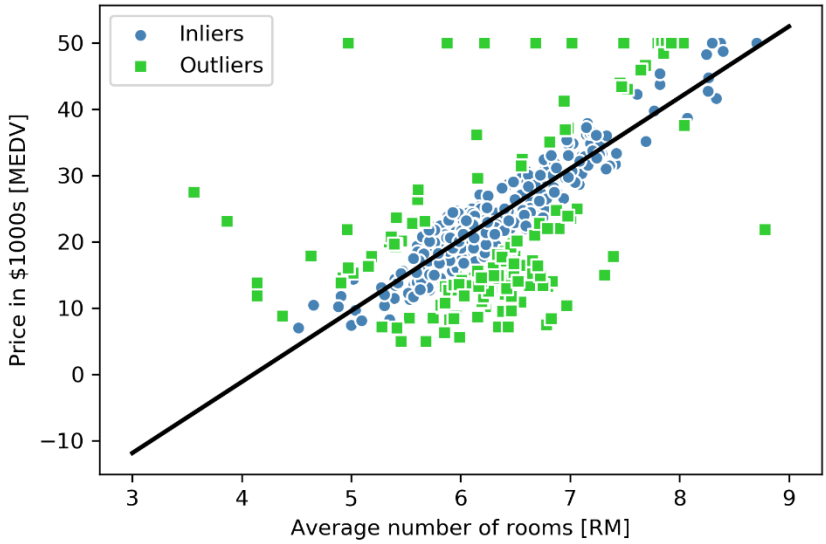
- RANSAC
- Regularisierung

***RAN**dom **SA**mple **C**onsensous*

1) Auswahl zufälliger Stichproben (*Inliners*) + Berechnung der Gewichte
2) Hinzufügen restlicher Datenpunkte und Berechnung von Abweichungen
   - Falls Abweichung klein ist, wird sie den Datenpunkten (*Inliners*) hinzugefügt
3) Neuberechnung der Gewichte
4) Abschätzen der Fehler und Überprüfung mit Schwellenwert

Parameter

- Anzahl der Stichprobe
- Anzahl der Versuche
- Schwellenwert
- Art der Abstandsbestimmung



Regularisierung

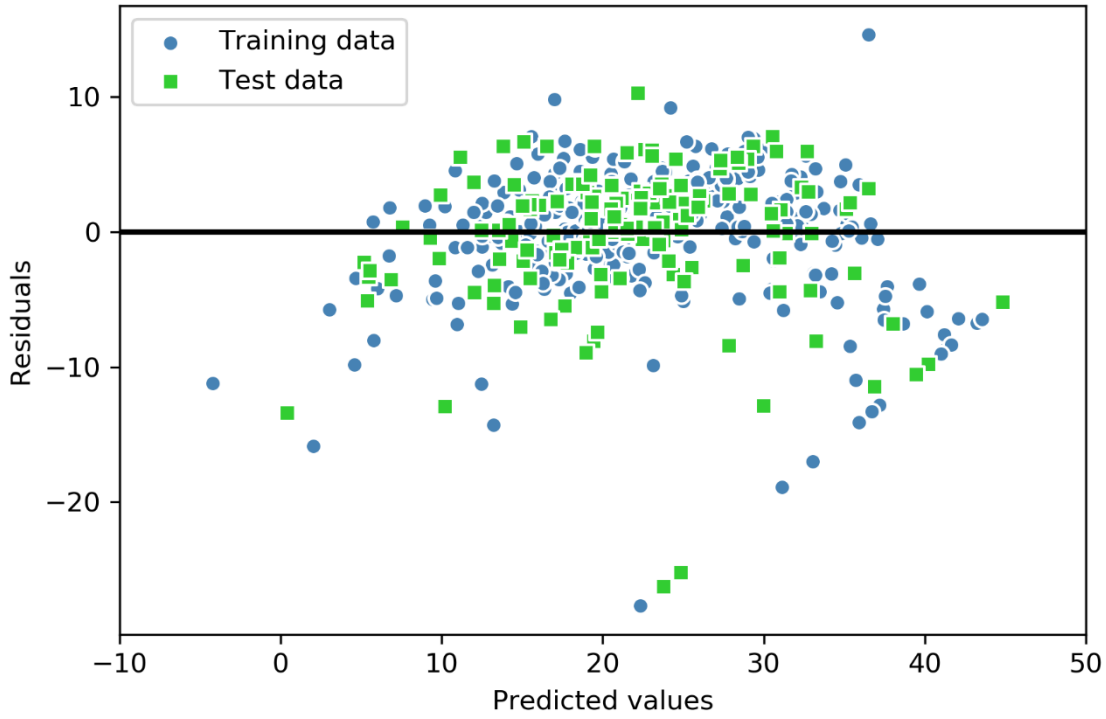
- Ähnlich zur Regularisierung bei Klassifizierern durch Hinzufügen von Informationen
- Ridge-Verfahren L2
- LASSO L1 (*Least Absolute Shrinkage and Selection Operator*)
- ElasticNet (Kompromiss zwischen Ridge und LASSO)

Bewertung von Regressionsmodellen

- Residuen-Diagramme
- MSE
- Bestimmtheitsmaß R^2

Residuen-Diagramme

Visualisierung der Abweichungen von Residuen bezüglich der tatsächlichen und der vorhergesagten Werte. Die Abweichungen sollten nahe 0 bzw. gleichverteilt um 0 sein. Falls Muster zu erkennen sind, ist das Modell vermutlich nicht in der Lage eine Vorhersage zu treffen.



Mean-Squared Error

Fehler der Durchschnittswerte für Trainings- und Testdaten mittels

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2.$$

Falls die Abweichung zwischen Trainings- und Testdaten zu groß ist, ist das Modell vermutlich überangepasst

Bestimmtheitsmaß R^2

Standardisierter MSE. Nutzt Sum of squared Error (SSE) und Sum of squared Total (SST).

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{MSE}{Var(y)}$$

Ensemble Learning

Kombination von Klassifizieren, um einen Meta-Klassifizierer zu erhalten. Kooperation statt Konkurrenz. Es sollte abgewogen werden, ob sich der Rechenaufwand lohnt.

- Mehrheitsenscheidung
- Bagging
- AdaBoost

Mehrheitsentscheidung

Auswahl der Klasse, die von den meisten Klassifizieren vorhergesagt wurde.

- Berechnung mithilfe eines Modalwerts, der das am häufigsten vorkommende Element auswählt.
- Eine Gewichtung der Algorithmen ist möglich

- Statt binärer Entscheidung der Klassifizieren sind auch Wahrscheinlichkeiten denkbar
- Eine Überprüfung mit ROC-Kurven ist sinnvoll

Bagging - Ensembles mit Bootstrap-Stichproben

Nicht mehr das Trainieren verschiedener Klassifizierer mit den gleichen Daten, sondern diese in Untermengen aufteilen und an die Klassifizierer verteilen. Anschließend erneute Kombination durch Mehrheitsentscheidung.

- Ein Objekt kann auch mehrfach oder gar nicht in Untermengen enthalten sein
- Verfahren um Varianz zu verringern (nur bei Modellen mit geringem Bias sinnvoll)

Adaptives Boosting

Verfahren, um mit schwachen Klassifizierern eine bessere Korrektklassifizierungsrate zu erhalten, indem iterativ aus Fehlklassifizierungen gelernt werden soll.

- Schrittweises Trainieren der Klassifizieren
    - in jedem Schritt wird ein Teil der zuvor fehlklassifizierten Objekte hinzugefügt (stärker gewichtet)
- zuletzt eine Mehrheitsentscheidung der trainierten Klassifizierern
- Bias und Varianz können kleiner werden, neigt jedoch zu Überanpassung mit hoher Varianz

Clusteranalyse

Verfahren des unüberwachten Lernens (nicht gekennzeichnete Daten) zum Finden von Kategorien / verborgenen Strukturen.

Anfällig für *Fluch der Dimensionalität* (Dimensionsreduktion ist wichtig)

- k-Means
- FCM
- Bewertung
- Hierarchisch
- DBSCAN

***k-Means** zur Gruppierung von ähnlichen Objekten*

- partionierendes prototypenbasiertes Verfahren
    - Zentroid (Durchschnitt ähnlicher Punkte mit stetigem Merkmal)
    - Medoid (repräsentativsten / am häufigsten vorkommender Punkt)
- gut für ringförmige Cluster und höherdimensionaler Daten
- Anzahl k der Cluster muss im Vorhinein festgelegt werden
- harte Zuordnung (jedes Objekt wird genau einem Cluster zugeordnet)
-

Mathematische Grundlagen

Ähnlichkeit von Objekten wird durch die Distanz d der quadrierten euklidischen Distanz zweier Punkte x und y festgelegt.

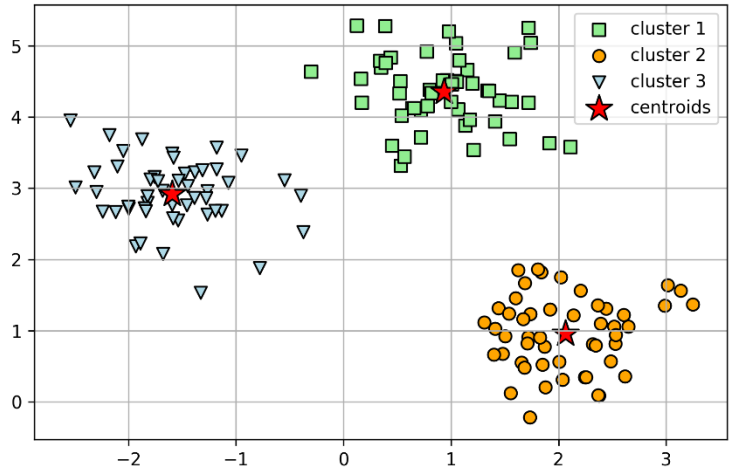
$$d(\mathbf{x}, \mathbf{y})^2 = \sum_{j=1}^{m} (x_j - y_j)^2 = ||\mathbf{x} - \mathbf{x}||_2^2$$

- mit j = Dimension der Merkmale

Sonstiges

- Minimierung von d mittels des SSE (auch Trägheit)
- Standardisierung sollte vorgenommen werden

Eine Verbesserung ist durch den **k-Means-++** Algorithmus gegeben, indem die anfänglichen Zentroiden nicht zufällig, sondern weit voneinander entfernt gesetzt werden.



*Fuzzy-C-Mean-**Algorithmus***

- weiche Zuordnung (ersetzten der harten Clusterzugehörigkeit durch eine Wahrscheinlichkeit)
- Anzahl k der Cluster muss auch im Vorhinein festgelegt werden

Mathematische Grundlagen

Minimierung der Zielfunktion J_m

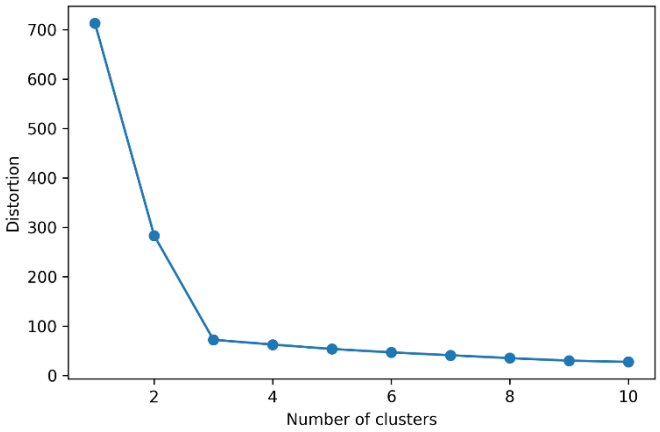
$$J_m = \sum_{i=1}^{n}\sum_{j=1}^{k}\left(w^{(i,j)}\right)^m ||\mathbf{x}^{(i)} - \mu^{(j)}||_2^2, \quad m \in [1,\infty]$$

- mit w(i,j) als Wahrscheinlichkeit zwischen 0 und 1 - Exponent m als *Fuzziness*-Koeffizient

Modellbewertung

- Ellbogenkriterium
- Silhouetten-Diagramm

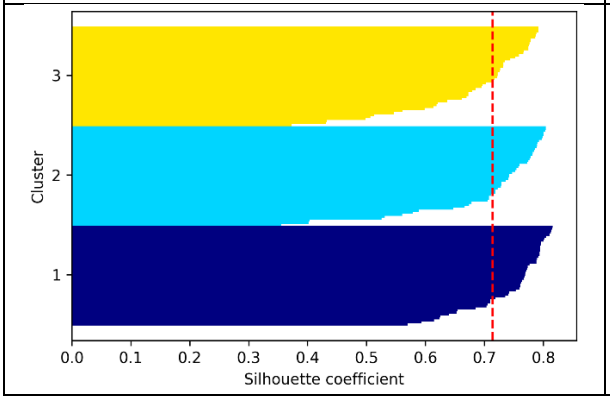
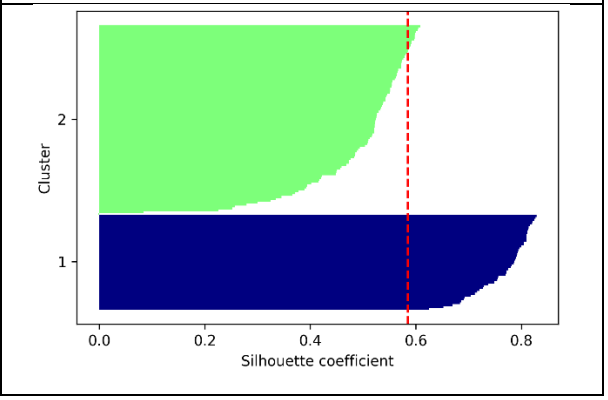
Ellbogenkriterium zum Finden des kleinsten k, bei dem die Verzerrung am heftigsten ansteigt. Je höher die Anzahl der Cluster sind, desto niedriger sind die Verzerrungen. Hier wäre k=3 am ehesten geeignet.



Silhouetten-Diagramm als graphische Methode zur Darstellung der Qualität eines Clusters.

- Berechnung der Geschlossenheit a (mittlere Distanz innerhalb eines Clusters)
- Berechnung der Distanz b zum nächsten Cluster (mittlere Distanz zwischen Clustern)
- Berechnung der Silhouette

$$s^{(i)} = \frac{b^{(i)} - a^{(i)}}{\max(a^{(i)}, b^{(i)})}$$

gut	*schlecht*
	

Hierarchisches Clustering

- Anzahl k der Cluster muss *nicht* bekannt sein

Divisiv

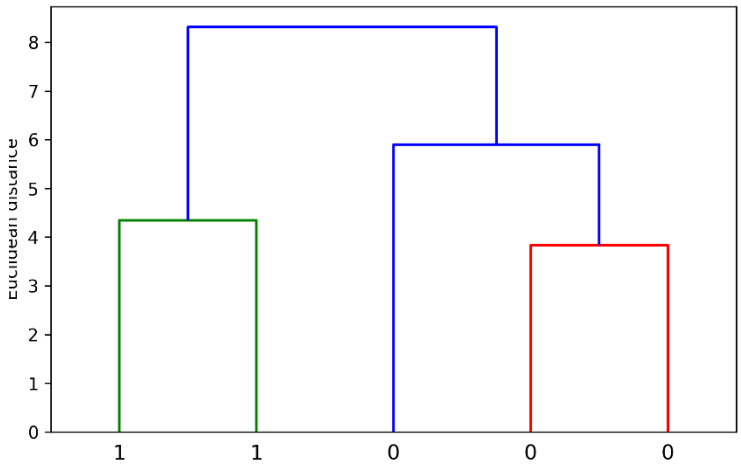
- Man geht von einem Cluster aus, das alle Objekte umfasst
- schrittweises Aufteilen in kleinere Cluster
- bis jedes Objekt ein Cluster bildet

Agglomerativ

- man geht davon aus, dass jedes Objekt ein Cluster bildet
- Zusammenführen der nächst-gelegenen Clusterpaare zu einem neuen Cluster
- bis nur noch ein Cluster verbleibt

Das Complete-Linkage Verfahren berechnet die Distanzmatrix aller Objekte, bevor die beiden nächstgelegenen Cluster zusammengeführt werden.

Dendogramm



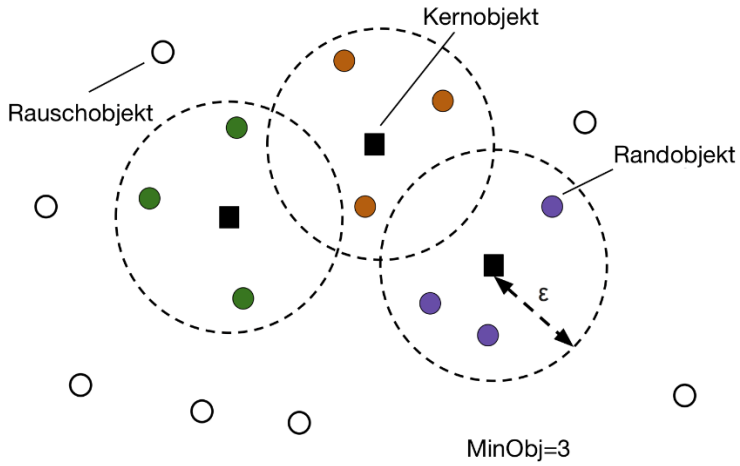
Density Based Spatial Clustering of Applications with Noise

- keine Annahmen über ringförmige / sphärische Cluster notwendig
- nicht alle Objekte müssen Cluster zugehören (Rauschen)

Verfahren

- Kernobjekt
  - innerhalb eines Radius $$
  - weitere Objekte in der Nähe
- Randobjekt
  - innerhalb eines Radius $$
  - wenige Objekte in der Nähe

- Rauschobjekt



Deep-Learning

Bezeichnet mehrschichtige Neuronaler Netze. Adaline ist prinzipiell ein einschichtiges neuronales Netzwerk (nur eine Verbindung).

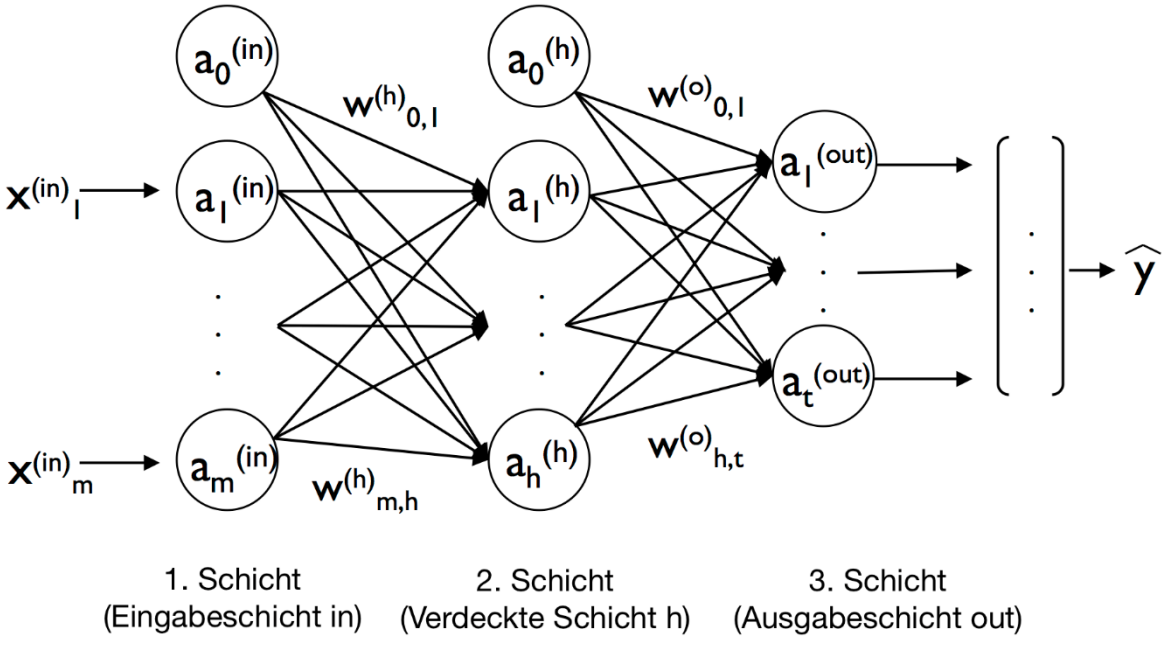
- MLP Übersicht
- MLP Aktivierung
- MLP Straffunktion
- MLP Backpropagation
- CNN
- Dropout
- RNN

MLP Übersicht

Mehrschichtiges Feedforward Netz (alternativ 'multi layer perceptron' kurz MLP). Unterscheidung in unterschiedliche Schichtarten:

- Eingabeschicht
- Verdeckte Schicht (Hidden Layer, mehrere möglich)
- Ausgabeschicht

Die einzelnen Schichten sind vollständig mit den Angrenzenden Schichten verknüpft.



Jede Schicht l hat n_l Aktivierungseinheiten $(a_i^{(l)})$:

- Die Größen der Schichten müssen nicht identisch sein.
- Jede Schicht hat eine eigene Bias-Einheit $(a_0^{(l)})$
- Die Aktivierungsfunktion ist die Sigmoid Funktion (siehe Logistische Regression)

Die Verbindungen zwischen den Schichten werden über Gewichte realisiert. Jede Schicht hat Gewichte für die Vorherige Schicht. Da die Elemente vollständig verknüpft sind kann man die Gewichte als Matrix $(w^{(l)})$darstellen.
Für die Ausgabeschicht wird meistens eine One-Hot-Kodierung verwendet (ein Eintrag für jede Klasse).
Hyperparameter für das MLP sind:

- Anzahl der Hidden Layers
- Anzahl der Aktivierungseinheiten pro Hidden Layer

Mehr Ebenen sorgen für einen kleineren Fehlergradienten und damit für ein langsameres lernen.

MLP Aktivierung

- Die Werte der Eingabeschicht entsprechen denen des Eingabevektors (X)
- Für jede versteckte Schicht kann die Nettoeingabefunktion einer Aktivierungseinheit wie folgt berechnet werden:

$$z_i^{(l)} = \sum_m a_m^{(l-1)} \cdot w_{m,i}^{(l)}$$

Oder alternativ in Vektorschreibweise:

$$z^{(l)} = a^{(l-1)} \cdot w^{(l)}$$

Anschließend, analog zu bisherigen Netzen, Bestimmung der Aktivierungsfunktion:

$$a_i^{(l)} = \phi(z_i^{(l)})$$

MLP Straffunktion

Analog zur logistischen Regression kann mit dem L2 Regulierungsterms die folgende Straffunktion aufgestellt werden:

$$J(\mathbf{w}) = -\left[\sum_{i=1}^{n} y^{[i]} \log(a^{[i]}) + (1 - y^{[i]}) \log(1 - a^{[i]})\right] + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

Da wir aber nun nicht mehr nur einen Ausgabewert haben (One-Hot), muss die Straffunktion dafür veralgemeinert werden:

$$J(\mathbf{w}) = -\sum_{i=1}^{n}\sum_{j=1}^{t} y_j^{[i]} \log(a_j^{[i]}) + (1 - y_j^{[i]}) \log(1 - a_j^{[i]}) + \frac{\lambda}{2}\sum_{l=1}^{L-1}\sum_{i=1}^{u_l}\sum_{j=1}^{u_{l+1}} (w_j, i^{(l)})^2$$

- t - Anzahl der Ausgabewerte.
- L - Anzahl der Schichten (Regularisierung muss alle Matrizen berücksichtigen)

Die hohe Anzahl der Dimensionen sorgt dafür, dass die Straffunktion sehr uneben ist und daher ein Verfahren gefunden werden muss, welches nicht einfach in lokalen Minima stecken bleibt.
Die Minimierung dieser Straffunktion (Gradientenabstiegsverfahren) ist sehr aufwendig und Rechenintensiv.
⇒ Backpropagation

MLP Backpropagation

Grundlagen

Kettenregel

Verschachtelte komplexe Funktion:

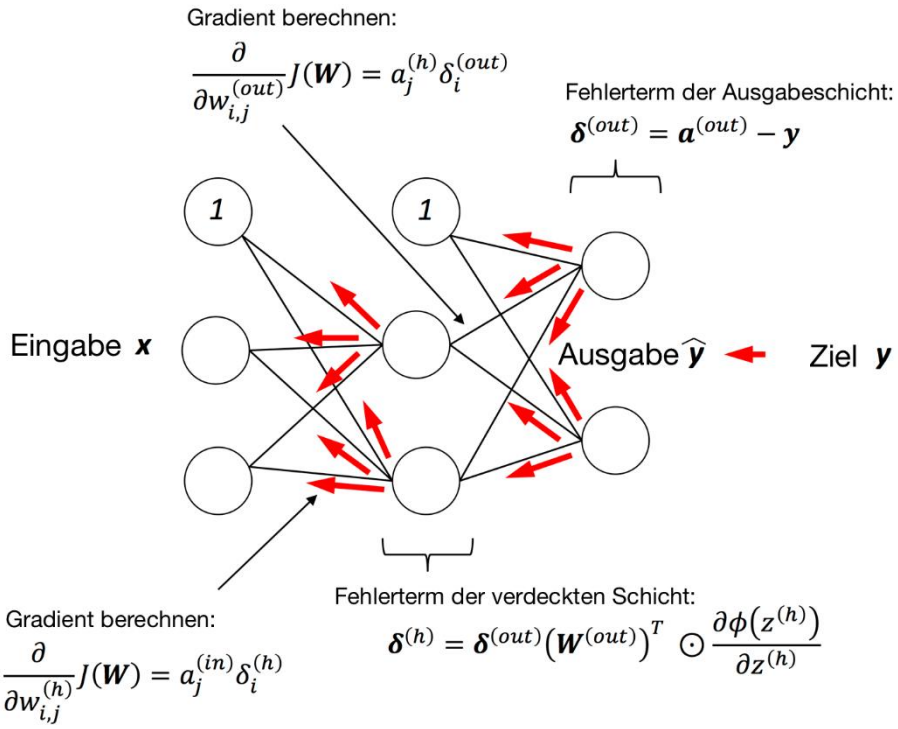
$$f(g(h((u(vx)))))$$

Ableitung der Funktion

$$\frac{d}{dx}[f(g(h((u(vx)))))] = \frac{df}{dg}\frac{dg}{dh}\frac{dh}{du}\frac{du}{dv}\frac{dv}{dx}$$

Das lässt sich mit automatischer Differenzierung im Rückwärtsmodus effizient berechnen.

Ablauf



1) Standard Foward-Propagation (Aktivierung des Netzes)
2) Bestimmung des Fehlers der Ausgabeschicht:

   $$\delta^{out} = a^{out} - y$$

3) Fehlerterm der Hidden-Layer bestimmen:

   $$\delta^h = \delta^{out} (W^{(out)})^T \odot \frac{\partial \phi(z^{(h)})}{\partial z^{(h)}}$$

   $$= \delta^{out} (\mathbf{W}^{(out)})^T \odot \left(a^{(h)} \odot (1 - a^{(h)}) \right)$$

4) Ableiten der Straffunktion nach verschiedenen Schichten:

Ableitungen nach Ausgabe-Schicht

$$\frac{\partial}{\partial w_{i,j}^{(out)}} J(\mathbf{W}) = a_j^{(h)} \delta_i^{(out)}$$

Ableitungen nach hidden-Schicht

$$\frac{\partial}{\partial w_{i,j}^{(h)}} J(\mathbf{W}) = a_j^{(in)} \delta_i^{(h)}$$

5) Zusammenführen der partiellen Ableitungen

$$\Delta^{(h)} := \Delta^{(h)} + (\mathbf{A}^{(in)})^T \delta^{(h)}$$

$$\Delta^{(out)} := \Delta^{(out)} + (\mathbf{A}^{(h)})^T \delta^{(out)}$$

6) Hinzufügen eines Regulierungsterms λ zu den Zusammengefassten Ableitungen:

$$\Delta^{(l)} := \Delta^{(l)} + \lambda^{(l)}$$

7) Aktualisierung der Gewichte mit der Lernrate (η):

$$\mathbf{W}^{(l)} := \mathbf{W}^{(l)} - \eta \Delta^{(l)}$$

CNN

Konvolutionale Neurale Netzwerke werden vor allem in der Bildverarbeitung eingesetzt.

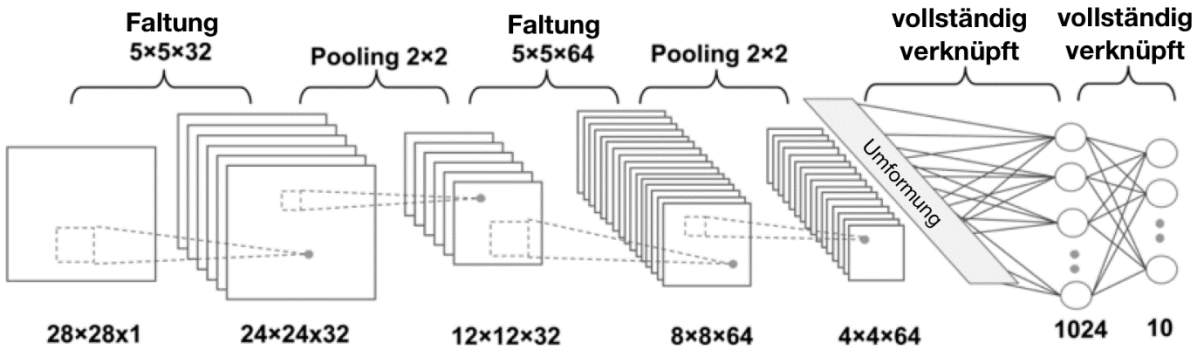
Extrahieren Features aus den Eingaben (meistens Bilder) ⇒⇒ Es müssen keine Merkmale von Hand ausgewählt werden.

Man unterscheidet zwischen zwei Verschiedenen Schichten:

1) Konvolutionsschichten (Faltungen)
   - Die Filtermatrizen müssen trainiert werden
   - Die generierten Matrizen werden als Merkmalskarten bezeichnet
   - Es können parallel mehrere Filter eingesetzt werden, siehe Beispiel
     - Mehrere Merkmalskarten
2) Pooling-Schichten (Subsampling, Unterstichproben)

- Besitzen keine lernbaren Parameter

Auf diese Schichten folgen dann noch einige weitere Vollverknüpfte schichten, analog zum MLP

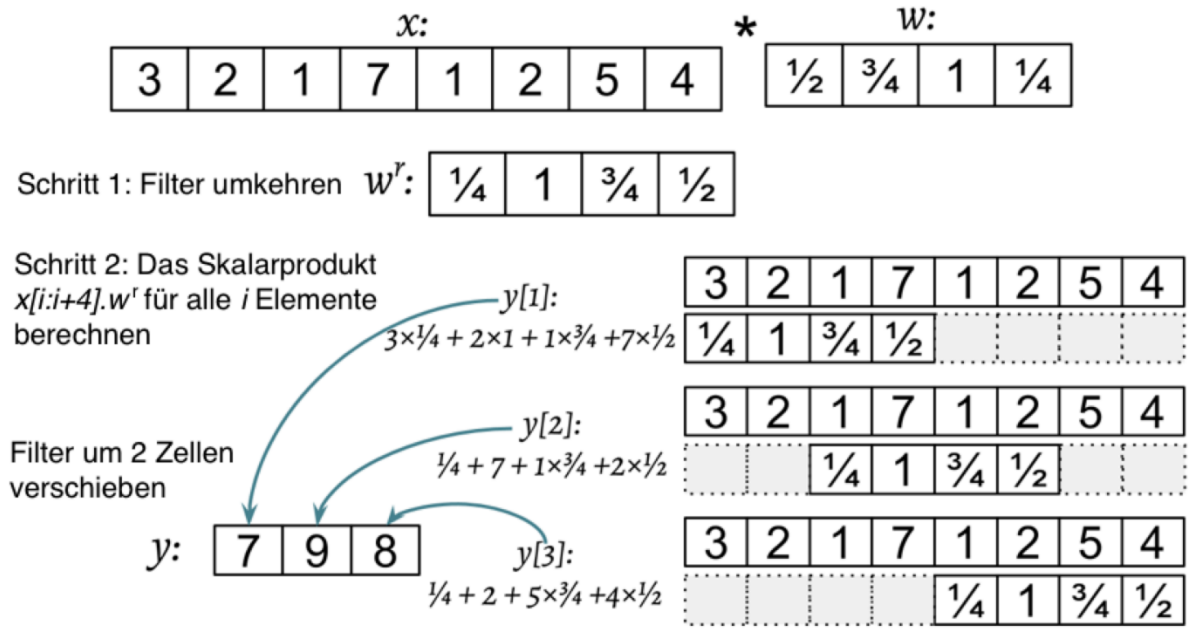


Faltung / Convolution

Wird eine Funktion f mit einer Funktion g gefaltet, so schreibt man: $(f*g)(x)$

Unterscheidung zwischen ein und zwei dimensionaler Faltung:

Eindimensional



In CNNs wird stets die Diskrete Faltung verwendet:

$$\mathbf{y} = \mathbf{x} * \mathbf{w}$$

$$\rightarrow \mathbf{y}[i] = \sum_{k=-\infty}^{\infty} \mathbf{x}[i-k]\mathbf{w}[k]$$

In der Mathematik gibt es aber auch einige stetige Faltung:

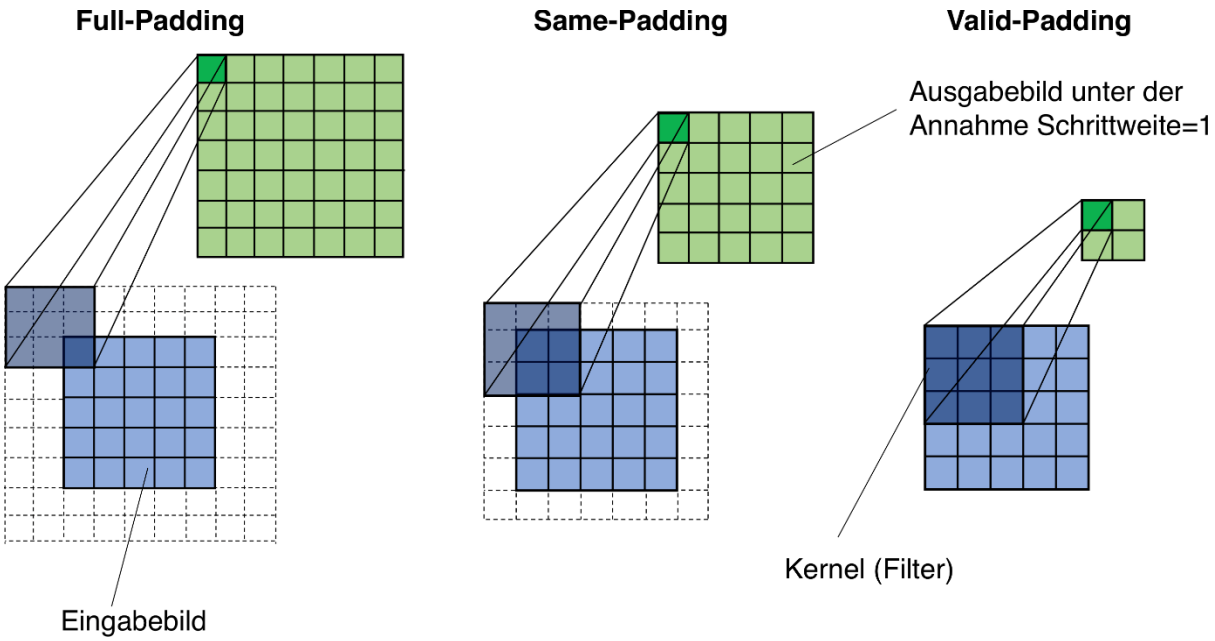
$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-\tau)\ g(\tau)\ d\tau$$

Damit die Werte in der Mitte keine höhere Bedeutung erhalten, kann man den Vektor xx mit Nullen auffüllen, man spricht hier von Padding (siehe Zweidimensionale Faltung).

- Mit höherem Padding wächst auch der Ausgabevektor y.
- Unterscheidung zwischen Full-Padding (p = m - 1), Same-Padding (len(y) == len(x)) und Valid-Padding (p=0, effektiv kein Padding)

Zweidimensional

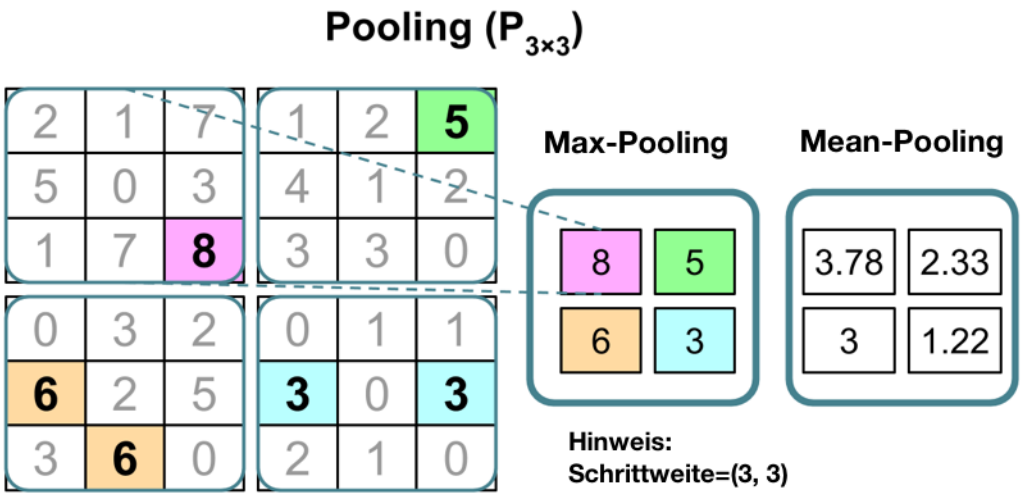
Die Zweidimensionale Faltung funktioniert quasi genauso wie die Eindimensionale. Statt zwei Vektoren werden hier aber nun zwei Matrizen verwendet.



Subsampling / Pooling

Eine Poolingschicht wird mit P_{n1xn2} bezeichnet

$n_1 n_1$ und $n_2 n_2$ geben die Größe der Poolingschicht an



Pooling verringert die Anzahl der Merkmale

- Effizienter
- Überanpassung kann reduziert werden.

Man unterscheidet zwischen

- Max Pooling (Maximalwert aus Nachbarschaft, führt zu einer leichten Invarianz)
- Mean Pooling (Mittelwert aus Nachbarschaft)

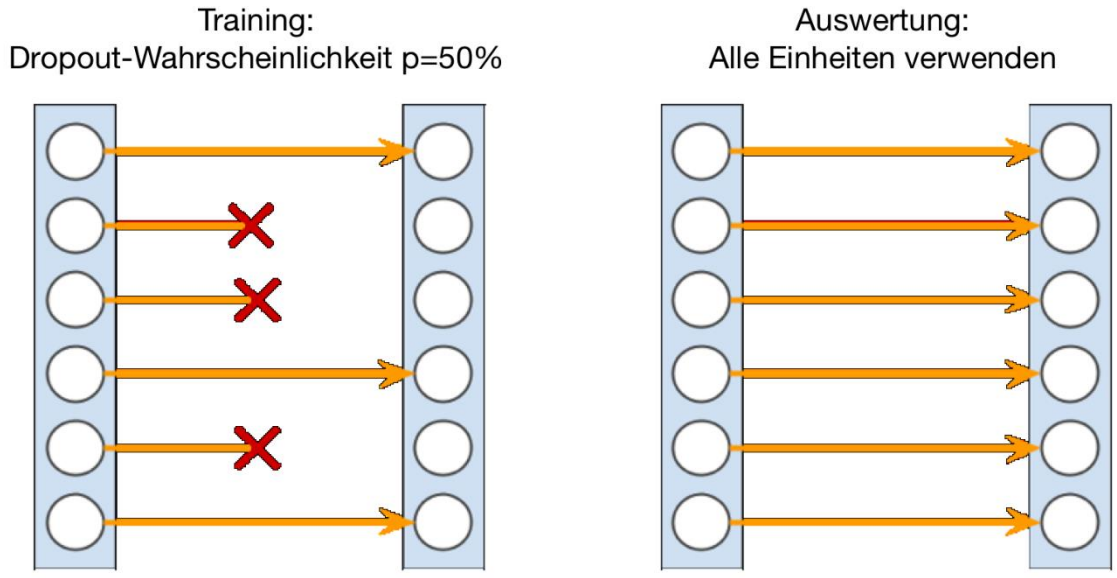
Parameter fürs Pooling sind:

- Pooling Größe (Spezifiziert als n_1 und n_2)
- Schrittweite (Verschiebung der Pooling Region, meistens so groß wie Pooling-Größe ⇒⇒ keine Überschneidungen)

Wichtig zu beachten ist, eine Pooling-Schicht besitzt keine erlenbaren Parameter, sie bleibt stets gleich.

Dropout

Neues Regularisierungsverfahren (Vermeidung von Überanpassung)



Während des Trainings wird der Einfluss von einzelnen Neuronen in jeder Iteration zufällig weggelassen

- Wahrscheinlichkeit p, im Regelfall p=0.5

Das Netz wird daher gezwungen die Informationen etwas redundant zu verarbeiten

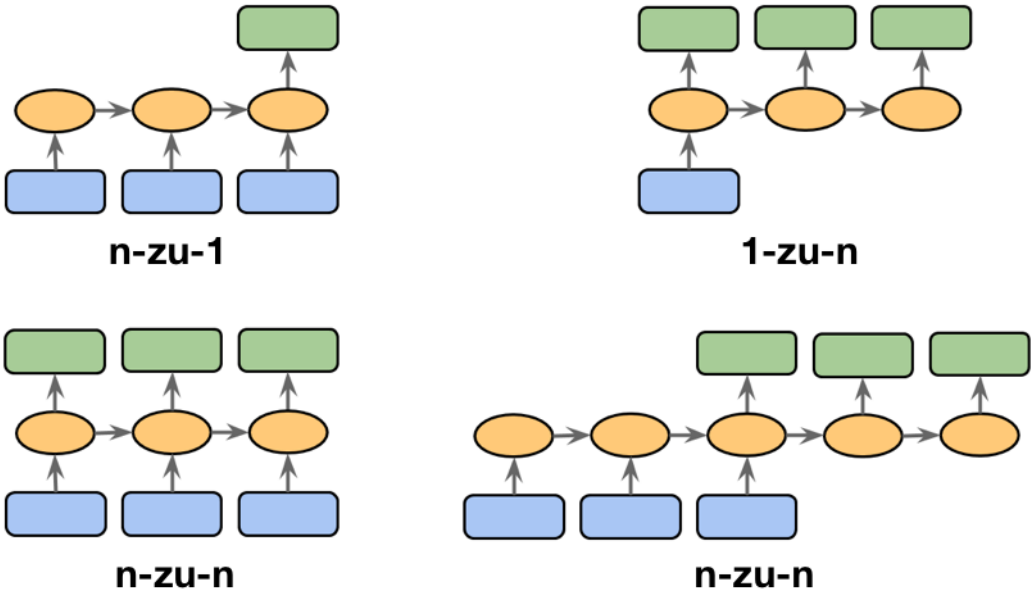
- Überanpassung wird effektiv reduziert

Rekurrente neuronale Netze

RNN berücksichtigen die Reihenfolge der verarbeiteten Daten

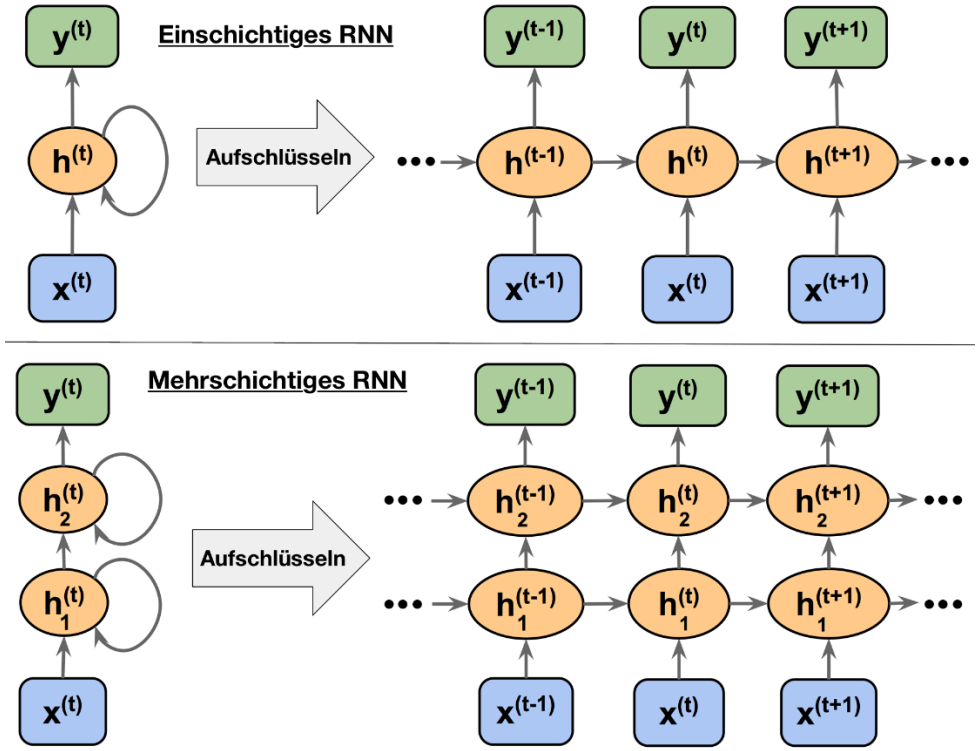
- Daten mit Reihenfolge ≙ Sequenz
- Es wird eine “Erinnerung” an bereits verarbeitete Objekte geschaffen.

Man unterscheidet vier Kategorien von RNNs:



- n-zu-1: Eingabe ist eine Sequenz, Ausgabe ist ein Vektor (z.B. Stimmungsanalyse von Text)
- 1-zu-n: Eingabe ist ein Vektor, Ausgabe ist eine Sequenz (z.B. Betitelung von Bildern)
- n-zu-n (synchron): Beides sind Sequenzen, Ausgabe erfolgt gleichzeitig mit der Eingabe (z.B. Videoklassifizierung wo jedes Einzelbild einzeln Klassifiziert wird)
- n-zu-n (asynchron): Wieder zwei Sequenzen, Ausgabe aber nicht Zeitgleich mit der Eingabe (z.B. Übersetzung: Einlesen von Satz, Übersetzen, Ausgeben)

Die einzelnen Schichten enthalten nun nicht mehr nur den einen Input von der vorherigen Schicht sondern ebenfalls den der aktuellen Schicht aus den vorherigen Datum.



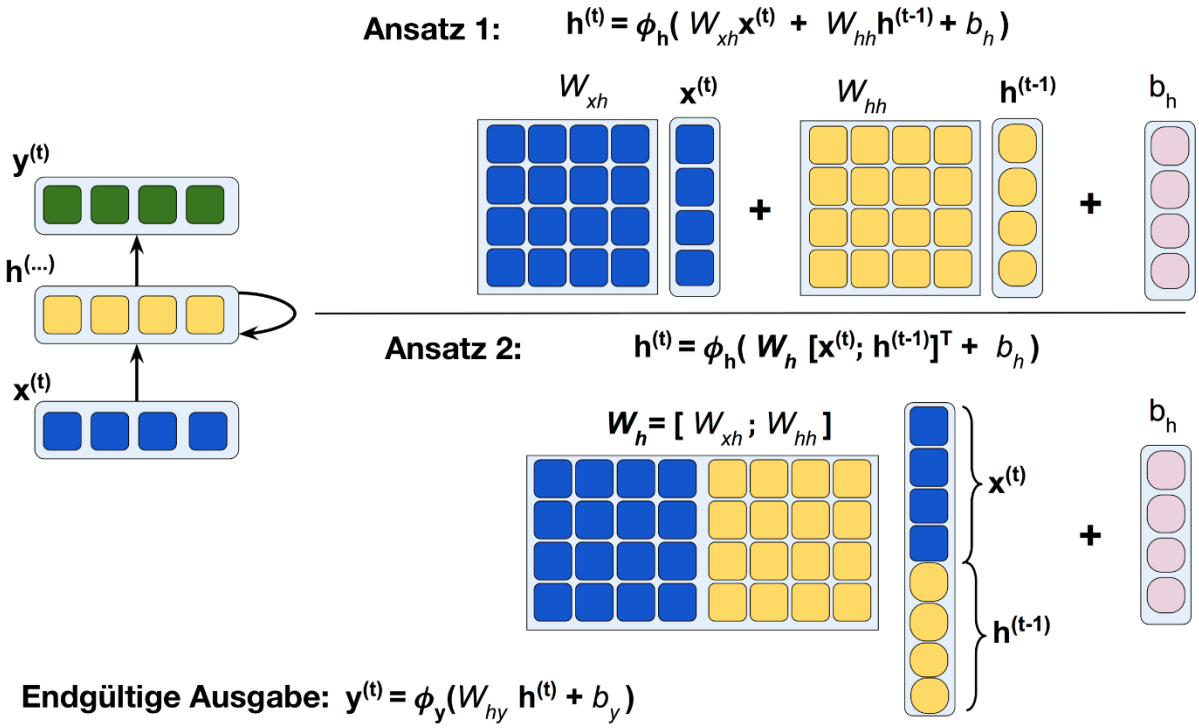
- Die zyklische Kante wird als rekurrente Kante bezeichnet (daher der Name)
- Für die Aktivierung werden beide Eingabevektoren mit den Jeweiligen Gewichten berechnet und anschließend addiert:

$$z_h^{(t)} = \mathbf{W_{xh}} x^{(t)} + \mathbf{W_{hh}} h^{(t-1)} + b_h$$

Zur Optimierung kann man die Beiden Gewichtungsmatrizen auch nebeneinander Schreiben und die beiden Aktivierungsvektoren übereinander anordnen und kann so die Aktivierung in einer einzelnen Vektor-Matrix Multiplikation ermitteln:

$$h^{(t)} = \phi_h \left([\mathbf{W_{xh}}. \mathbf{W_{xh}}] \begin{bmatrix} x^{(t)} \\ h^{(t-1)} \end{bmatrix} + b_h \right)$$

Grafisch dargestellt sieht das dann wie folgt aus:



Für das Training unterscheidet man vor allem zwischen drei Ansätzen:

BackPropagation-Through-Time

- Der Gesamtverlust (L) wird als Summe der einzenlen Verluste l) über die Zeit zwischen [1,T] berechnet:

$$L = \sum_{t=1}^{T} l^{(t)}$$

Der Gradient enthält einen Faktor, welcher dafür sorgt, dass er je nach Gewichtung der rekurenten Verknüpfung verschwindet ($|w_{hh}|<1$) oder explosionsartig wächst $|w_{hh}>1$).

- Gewünscht ist: $w_{hh}=1$

Die Naive Lösung ist, die Gewichte stets entsprechend zu skalieren.

Alternativen sind die anderen beiden Ansätze

Truncated BackPropagation-Through-Time

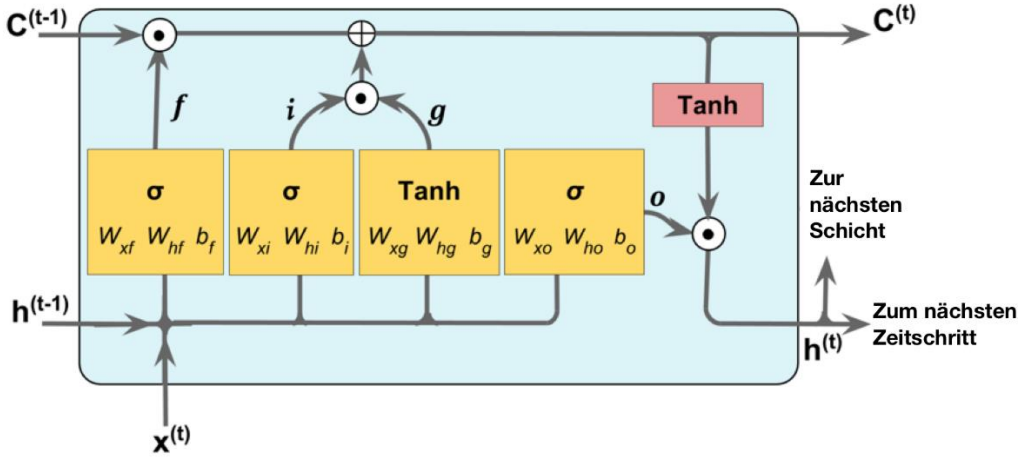
Funktioniert nahezu genauso wie **BPTT**, jedoch wird der Gradient ab einem bestimmten Grenzwert einfach abgeschnitten.

- Explosionsartiges Wachstum wird verhindert

- Gradient wird irgendwann nicht mehr sinnvoll fürs Training

Long Short-Term Memory

- Erfolgreich bei der Modellierung umfangreicher Sequenzen
- Einführung von Speicherzellen
    - Ersetzen die Bisherige Aktivierungsfunktion (?)



Beispiel - Natural Language Processing

<u>Klassifizierung</u>

1) Durch reguläre Ausdrücke Sonderzeichen entfernen oder formatieren
2) Text in Token zerlegen (einzelne Wörter)
3) Tokens filtern
    - Auf Wortstamm bringen mittels *stemming*
    - Stoppwörter entfernen (durch `nltk` bereitgestellt)
4) Worte in Merkmale umwandeln (Merkmalsvektor, **Bag-of-Word**)
5) mittles tf-idf-Maß (**Raw Term Frequency**) durchweg häufig vokommende Wörter bestrafen
6) Rastersuche (oder stochastisches Verfahren), um das Modell zu trainieren
7) stochastisches Verfahren: Klassifizierer serialisieren

<u>Topic Modeling (Clustering, unüberwacht)</u>

Mittels Latent Dirichlet Allokation (LDA) Wortgruppen verschiedener Dokumente finden, die Themen widerspiegeln. Die Anzahl der Themen muss im Vorhinein festgelegt werden.