



# **VORLESUNG 11**

# **MASCHINELLES LERNEN**

# AGENDA

**1** Grundlagen des Maschinellen Lernens

**1.1** Supervised Learning

**1.2** Unsupervised Learning

**1.3** Weitere Lernkonzepte

**2** Tipps und Tricks

**3** Evaluierungskonzepte

**4** Zusammenfassung



KAPITEL 1

# GRUNDLAGEN DES MASCHINELLEN LERNENS

**A COMPUTER PROGRAM IS SAID  
TO LEARN FROM EXPERIENCE  $E$   
WITH RESPECT TO SOME CLASS  
OF TASKS  $T$  AND PERFORMANCE  
MEASURE  $P$  IF ITS PERFORMANCE  
AT TASKS IN  $T$ , AS MEASURED BY  
 $P$ , IMPROVES WITH EXPERIENCE  
 $E$ .**

Tom M. Mitchell  
Carnegie Mellon University

# WAS IST MASCHINELLES LERNEN?

- Formuliert wird eine Hypothese  $h(x, \omega)$
  - Der Lernschritt erfolgt über eine Anpassung der Gewichte  $\omega$  über Optimierung einer Funktion anhand gegebener Daten
  - Ziel ist es, die Hypothese so zu wählen, dass sie auf unbekannten Daten richtige Aussagen treffen kann
- Jedes maschinelle Lernen ist eine Funktionsoptimierung auf Daten!

# ÜBERSICHT DER TEILBEREICHE

- **Supervised Learning**
  - Es liegen Daten und entsprechende Labels vor
  - Direktes Feedback beim Lernen
  - Ziel: Vorhersage über zukünftige Ereignisse
- **Unsupervised Learning**
  - Es liegen Daten ohne Labels vor
  - Kein Feedback
  - Ziel: Auffinden von Strukturen in den Daten
- **Reinforcement Learning**
  - Algorithmus/Agent interagiert mit der Umgebung
  - Lernt Aktionen anhand eines Reward-Systems





KAPITEL 1.1

# SUPERVISED LEARNING

# ÜBERSICHT DER TEILBEREICHE

- Gegeben  $m$  Trainingsdaten  $\{(x^{(i)}, y^{(i)}); i = 1 \dots m\}$
- $x^{(i)}$ : Inputvektor
- $y^{(i)}$ : Target
  - Wenn  $y$  kontinuierlich: Regression
  - Wenn  $y$  diskrete: Klassifikation
- Ziel ist es, eine Hypothese / Funktion  $h$  so zu lernen, dass es für unbekannte Inputvektoren den richtigen Output vorhersagt.



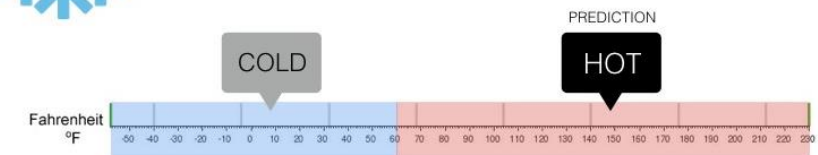
## Regression

What is the temperature going to be tomorrow?



## Classification

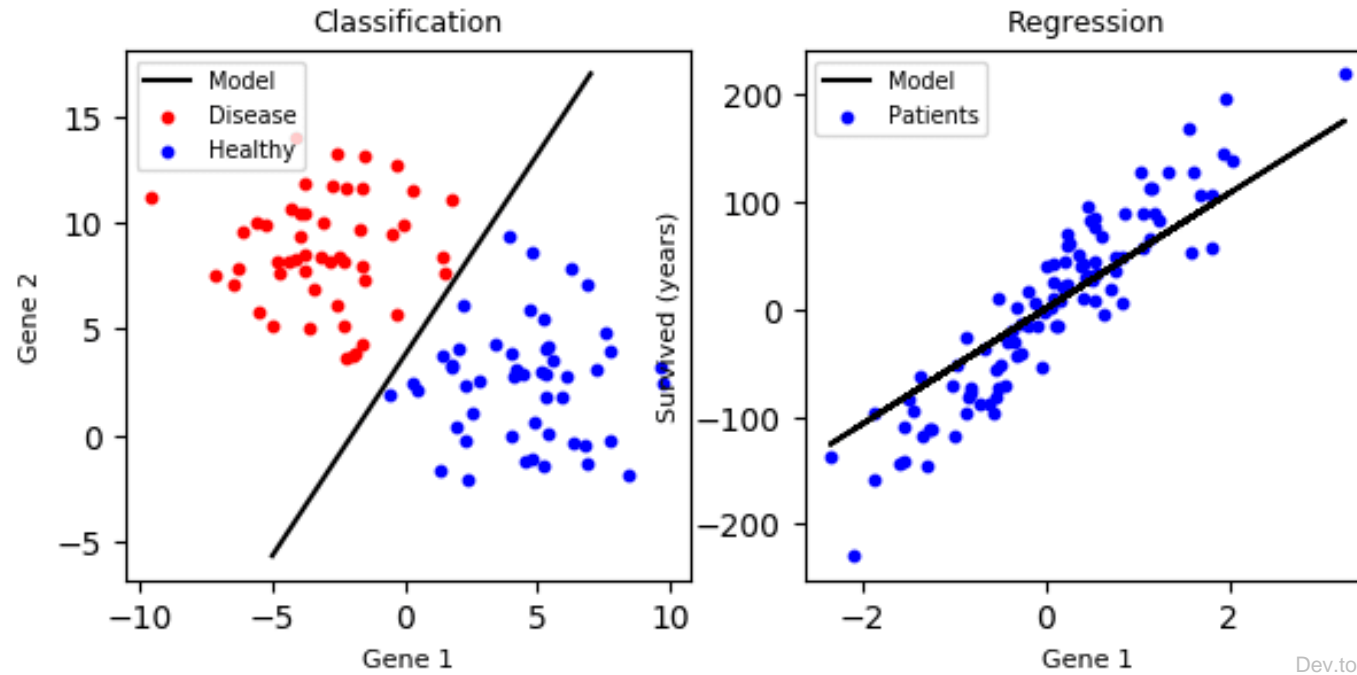
Will it be Cold or Hot tomorrow?



Towardsdatascience.com

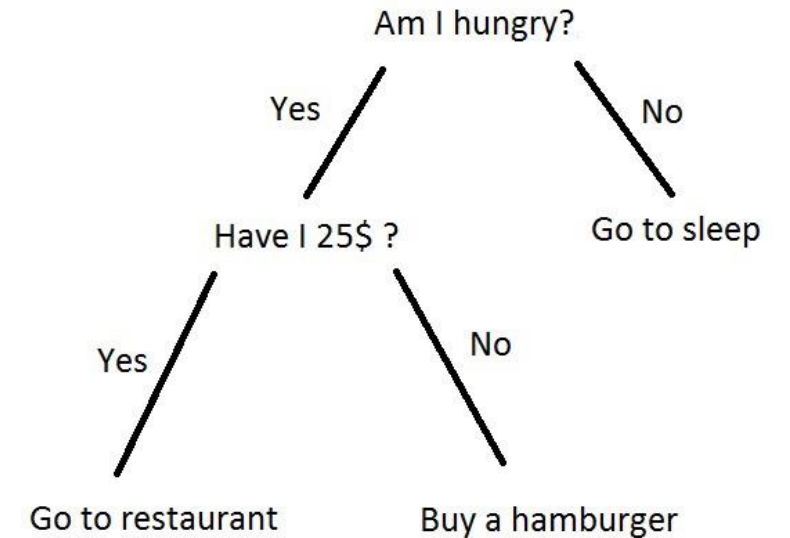


# KLASSIFIKATION VS REGRESSION



# DECISION TREES

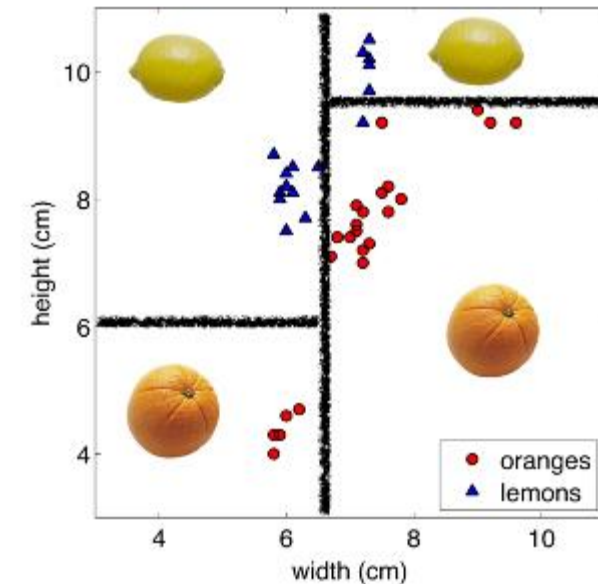
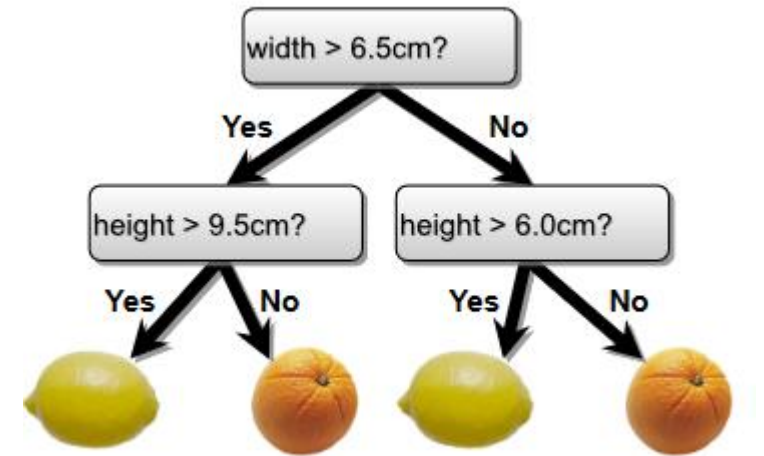
- Klassifikator in Baumstruktur
- Jeder Knoten ist ein Test für eine Variable
- Der Ausgang jeden Tests entscheidet, zu welchen Nachbarknoten man sich bewegt
- Endknoten stellen die Vorhersage / Prediction dar



- Features
  - Hungrig (boolean)
  - Geld (kontinuierlich)

# DECISION TREES

- Auf jeder Ebene wird bestimmt
  - Welche Variable gesplittet werden soll.
  - Wo diese gesplittet werden soll.
- Die Bewertung der Splits kann über Entropie erfolgen.



# REGRESSION – LINEARE REGRESSION

- Gegeben  $m$  Trainingsdaten  $\{(x^{(i)}, y^{(i)}); i = 1 \dots m\}$

- $x^{(i)}$ : Inputvektor
- $y^{(i)}$ : Target (kontinuierlich)

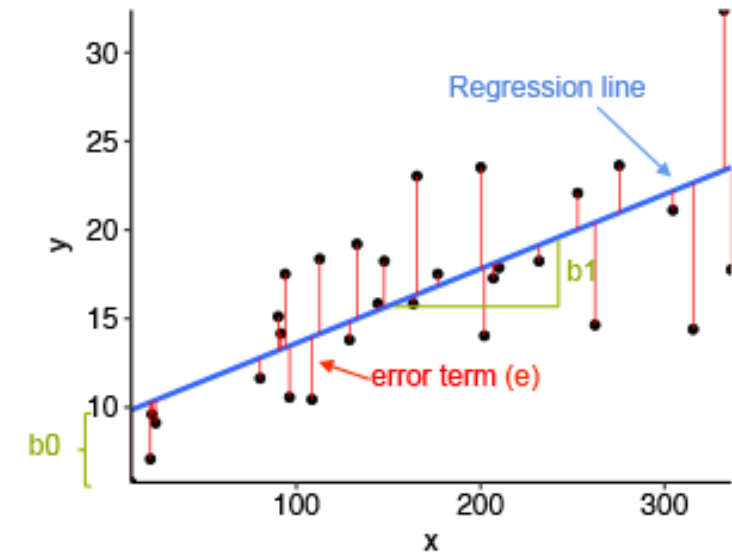
- Idee: gewichte jedes Feature linear

- Vorhersage / Hypothese:

$$h(x) = \sum_{i=0}^n \theta_i x_i = \theta^T x$$

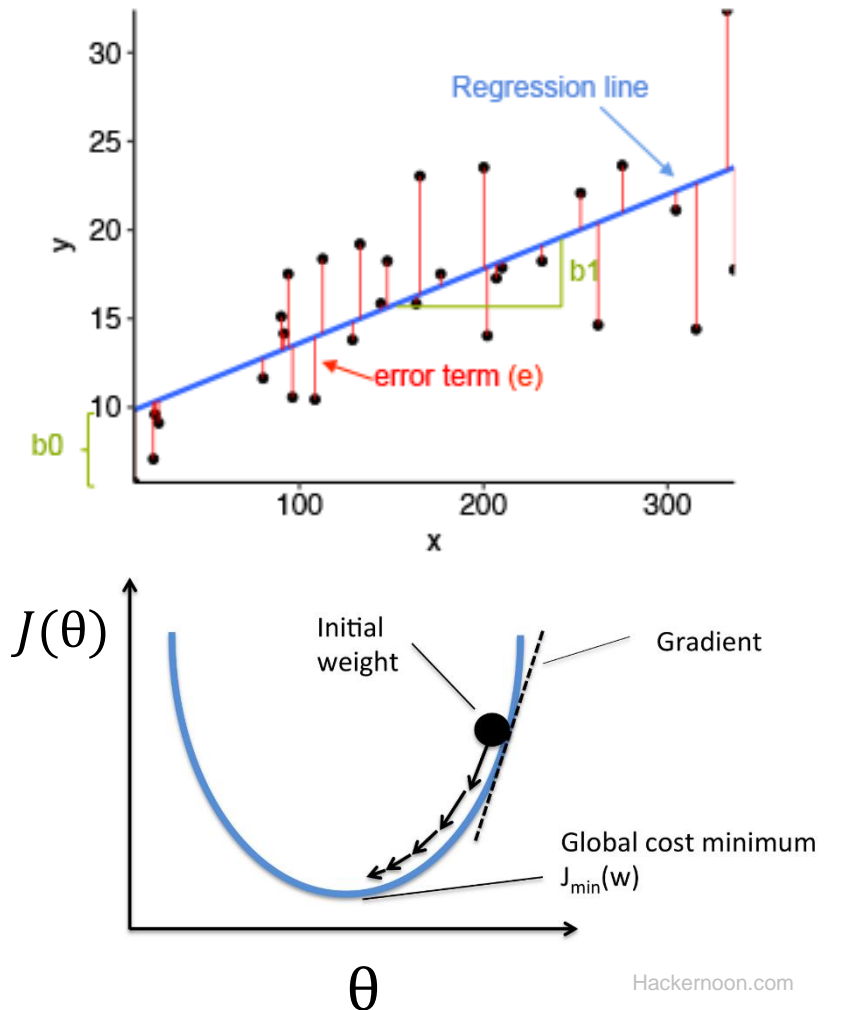
- Minimierung der folgenden Kostenfunktion (Least-Square):

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$



# REGRESSION – LINEARE REGRESSION

- Kostenfunktion  $J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$
- Ableitung
 
$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_{\theta}(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_{\theta}(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_{\theta}(x) - y) \\ &= (h_{\theta}(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left( \sum_{i=0}^n \theta_i x_i - y \right) \\ &= (h_{\theta}(x) - y) x_j \end{aligned}$$
- Update-Regel:  $\theta_j := \theta_j + \alpha \sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)}$  (for every  $j$ ).



# KLASSIFIKATION – LOGISTIC REGRESSION

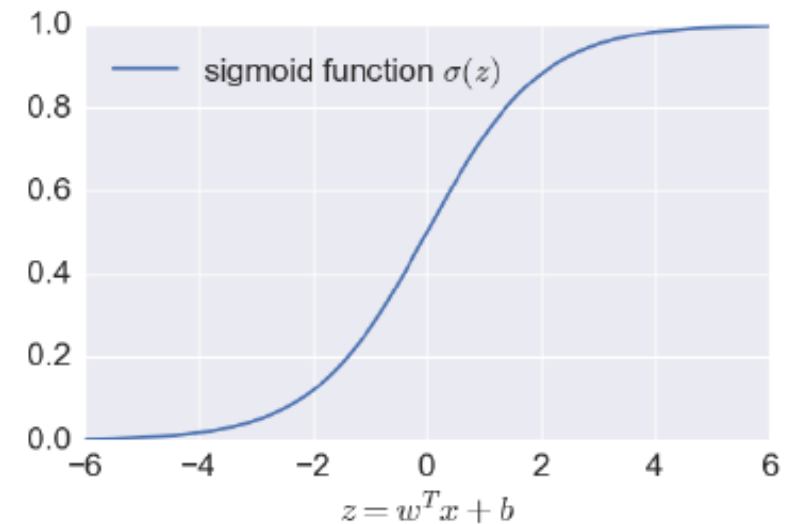
- Lineares Model für binäre Klassifikation (basierend auf lineare Regression)
- Es geht von der Annahme aus, dass sich die Wahrscheinlichkeiten der beiden eintretenden Klassen über Sigmoidfunktionen darstellen lassen:

$$\begin{aligned} p(y = 1|x, w, b) &= \sigma(w^T x + b) \\ p(y = 0|x, w, b) &= 1 - \sigma(w^T x + b) \end{aligned} \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

- Model kann als Bernoulliverteilung beschrieben werden:

$$p(y|x, w, b) = (\sigma(w^T x + b))^y \cdot (1 - \sigma(w^T x + b))^{1-y}$$

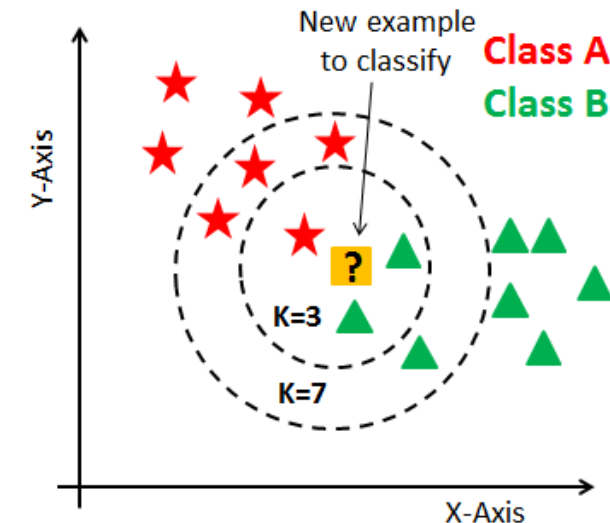
- Training findet über Maximum Likelihood Estimation (MLE) statt, d.h.  $p(y|x, w, b)$  wird über die Gewichte optimiert.





# K-NEAREST NEIGHBOR

- Lazy learner: es werden alle Trainingsdaten einfach abgespeichert
- Während der Vorhersage:
- Finde die  $k$  nächsten Nachbarn
- Für **Klassifikation**:
  - Abstimmung der Nachbarn für welche Klasse diese sind (Label auslesen)  
→ Mehrheit entscheidet
  - Evtl. gewichtet mit der Distanz zum Query



- Für **Regression**:
  - Mittelwert der Nachbarlabels  
→ Mittelwert der Labels werden zurückgegeben
  - Evtl. gewichteter Mittelwert

# SUPPORT VECTOR MACHINE KLASSIFIKATION

- Support Vector Machines (SVM) sind aus der statistischen Lerntheorie entstanden.
- Diese wählt ein „optimales“ Modell aus einer Menge von Modellen und nimmt nicht vorher an, das korrekte Modell zu kennen.
- „Optimal“ bezieht sich auf die Generalisierungsfähigkeit des Modells und daher auf die Fähigkeit, die Fehlerwahrscheinlichkeit auf allen Daten zu minimieren.

$$R(\mathbf{w}) \leq R_{emp}(\mathbf{w}) + \epsilon(n, p^*, h)$$

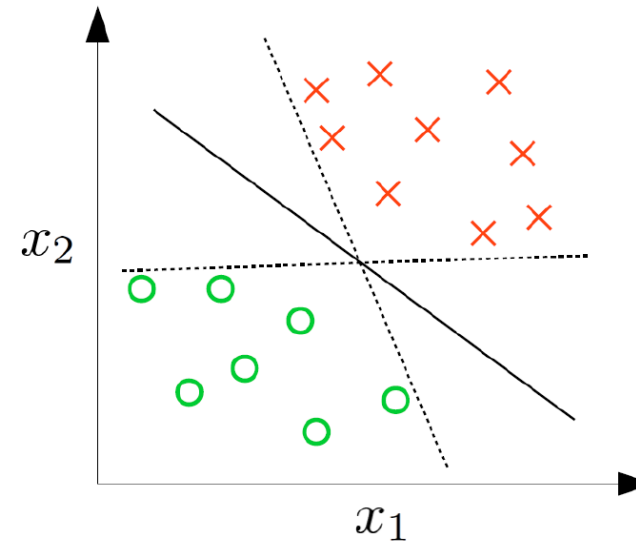
True risk

Empirical risk  
(training error)

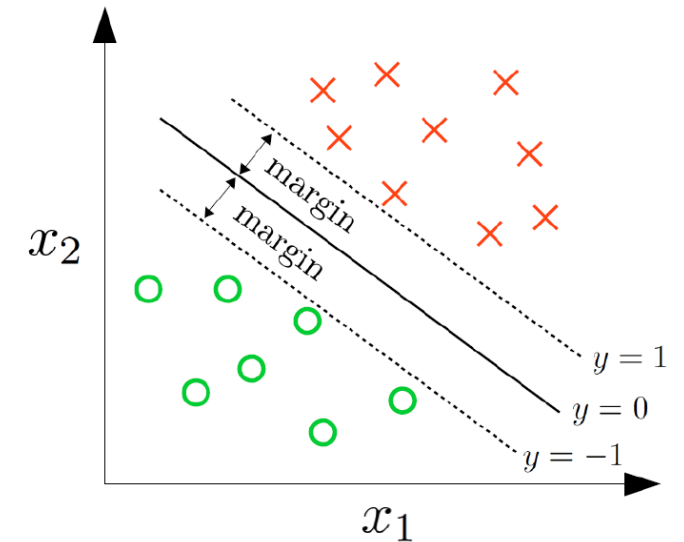
Complexity

# SUPPORT VECTOR MACHINE KLASSIFIKATION

- Idee von (linearen) SVM:
  - Finde eine Decision Boundary, die die Margin zwischen Decision Boundary und den nächstliegenden Punkten maximiert.



(a) Possible decision boundaries

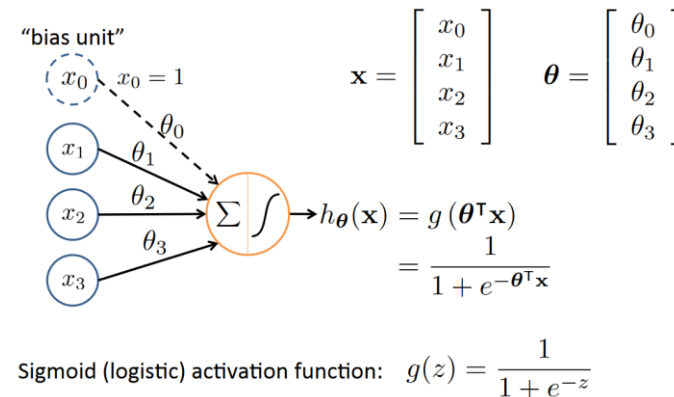


(b) Maximum margin principle

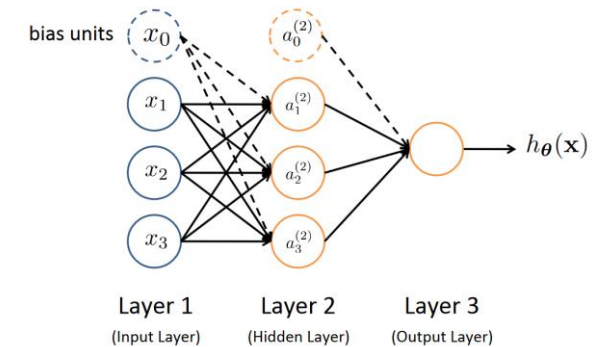
# NEURONALE NETZE

- Lernen einer Funktionen, deren Komplexität sehr variabel modelliert werden kann
- NN bestehen aus Knoten und Verbindungen
- Jede Verbindung ist mit einem Gewicht assoziiert
- Jeder Knoten hat eine Aktivierungsfunktion und einen Output

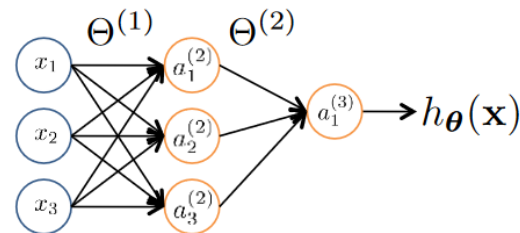
Neuron Model: Logistic Unit



Neural Network



# NEURONALE NETZE



$a_i^{(j)}$  = “activation” of unit  $i$  in layer  $j$

$\Theta^{(j)}$  = weight matrix controlling function mapping from layer  $j$  to layer  $j + 1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has  $s_j$  units in layer  $j$  *and*  $s_{j+1}$  units in layer  $j+1$ ,  
then  $\Theta^{(j)}$  has dimension  $s_{j+1} \times (s_j + 1)$  .

$$\Theta^{(1)} \in \mathbb{R}^{3 \times 4} \quad \Theta^{(2)} \in \mathbb{R}^{1 \times 4}$$

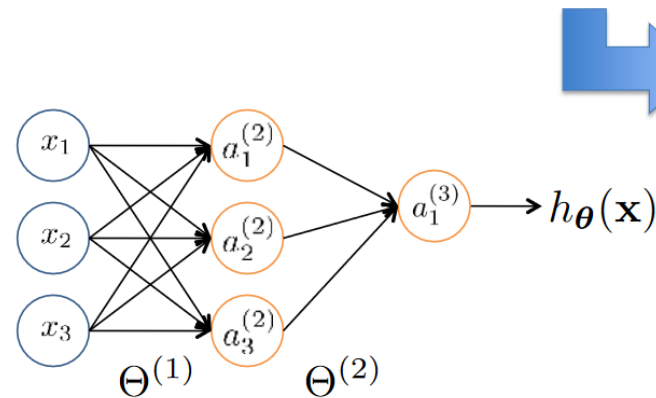
# NEURONALE NETZE

$$a_1^{(2)} = g \left( \Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3 \right) = g \left( z_1^{(2)} \right)$$

$$a_2^{(2)} = g \left( \Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3 \right) = g \left( z_2^{(2)} \right)$$

$$a_3^{(2)} = g \left( \Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3 \right) = g \left( z_3^{(2)} \right)$$

$$h_{\Theta}(\mathbf{x}) = g \left( \Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)} \right) = g \left( z_1^{(3)} \right)$$



## Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

Add  $a_0^{(2)} = 1$

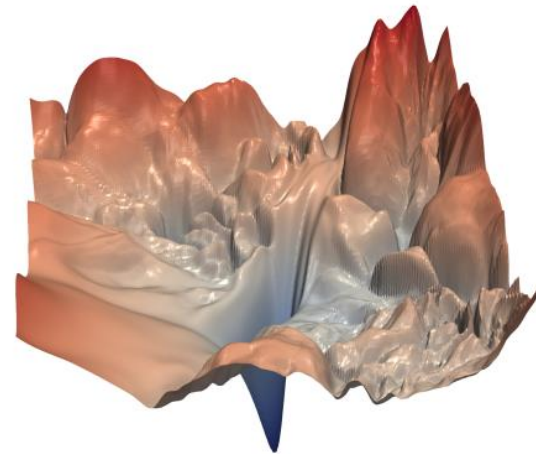
$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

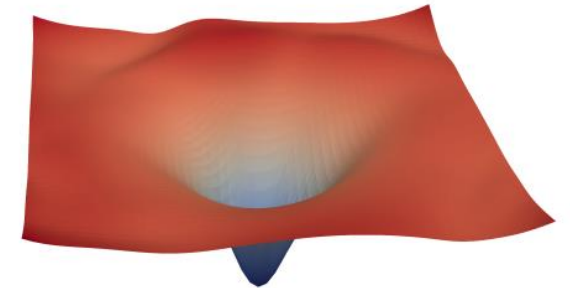


# NEURONALE NETZE

- Für das Training muss eine nicht-konvexe Kostenfunktion bestimmt werden
- Diese wird meist über ein Gradientenabstiegsverfahren minimiert



(a) without skip connections



(b) with skip connections

The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

Li, Hao, et al. "Visualizing the loss landscape of neural nets." Advances in neural information processing systems 31 (2018).



KAPITEL 1.2

# UNSUPERVISED LEARNING

# ÜBERSICHT DER TEILBEREICHE

- Supervised Learning
  - Es liegen Daten und entsprechende Labels vor
  - Direktes Feedback beim Lernen
  - Ziel: Vorhersage über zukünftige Ereignisse
- **Unsupervised Learning**
  - **Es liegen Daten ohne Labels vor**
  - **Kein Feedback**
  - **Ziel: Auffinden von Strukturen in den Daten**
- Reinforcement Learning
  - Algorithmus/Agent interagiert mit der Umgebung
  - Lernt Aktionen anhand eines Reward-Systems

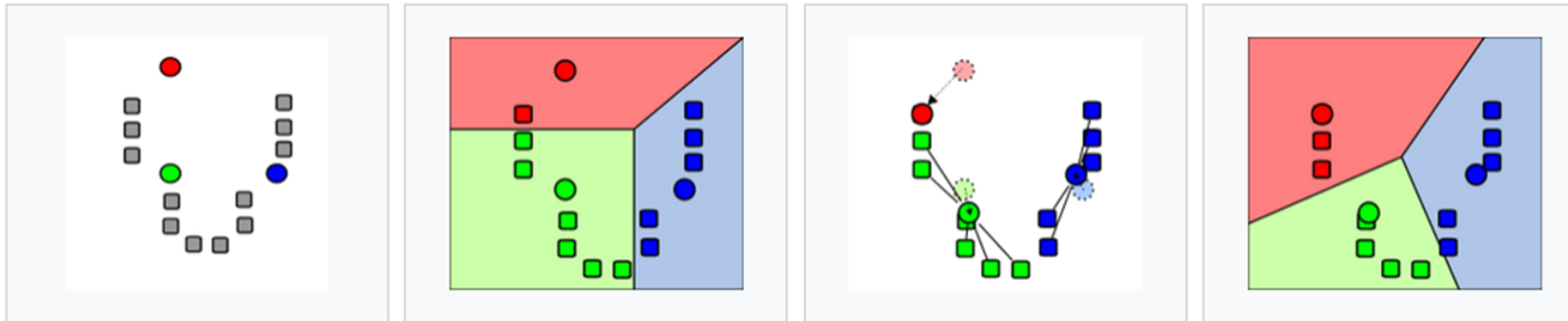
# K-MEANS CLUSTERING

- Ziel: Finde einzig anhand der Daten Gruppen (Cluster) von Datenpunkte, die ähnliche Eigenschaften besitzen.
- Initialisiere zufällig Centroids  $\mu_i$  für jeden Cluster und löse folgendes Optimierungsproblem:

$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \mu_i\|^2$$

- Indem folgende zwei Schritte abwechselnd wiederholt werden:
  - Weise jeden Datenpunkt einem Cluster zu, indem die kürzeste Distanz als dessen Zugehörigkeit interpretiert wird. (Assignment)
  - Berechne die neuen Centroids, indem die Schwerpunkte pro Cluster anhand deren Datenpunkte neu berechnet wird (Update)


# K-MEANS CLUSTERING



Zufällige  
Initialisierung

Assignment  
step

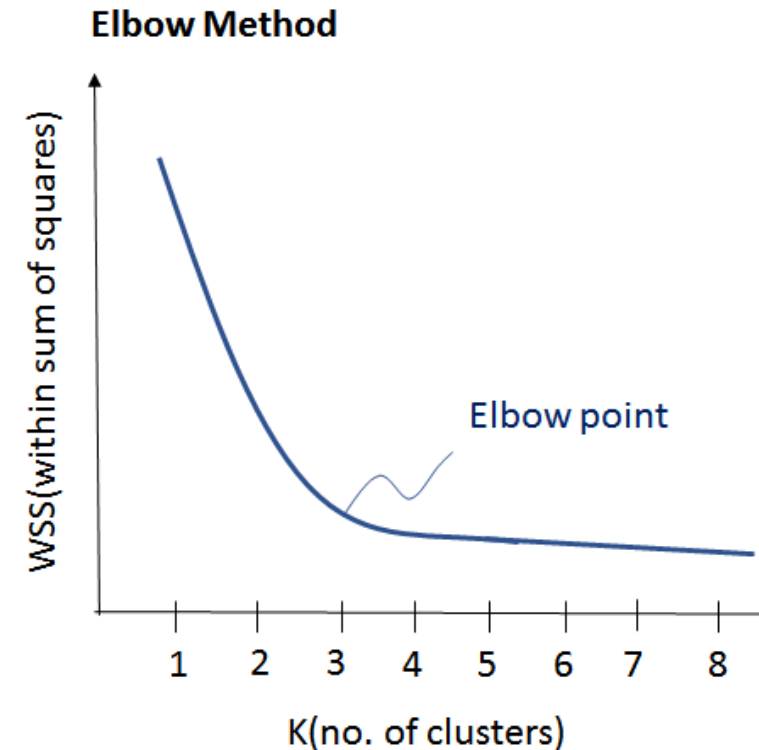
Update step



Wiederholen, bis es konvergiert

# K-MEANS: WIE VIELE CLUSTER

- K-Fold Cross Validation (wird später behandelt)
- Ellenbogen-Methoden
  - Wähle  $k$  im „Knick“





# K-MEANS CLUSTERING

- **Vorteile:**
  - Simple und effektive  
(auch heute noch das meist genutzt Clustering-Verfahren)
  - Gute Skalierbarkeit
- **Nachteile:**
  - Anzahl der Cluster muss manuell gewählt werden
  - Probleme, wenn die Cluster in Größe und Dichte variieren
  - Sensitiv gegenüber Outlier

# DBSCAN

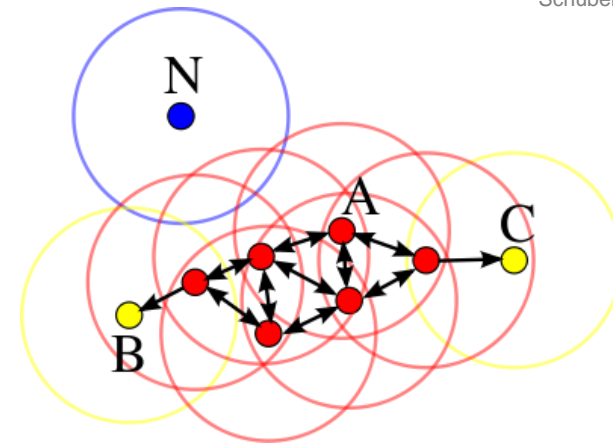
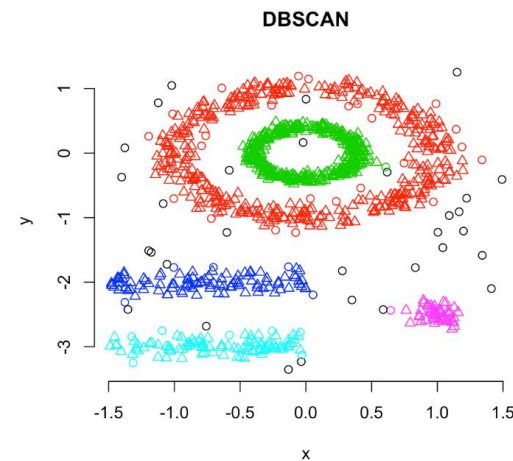
- Density-based spatial clustering of applications with noise (DBSCAN)
- Dichte-basierendes Clustering-Verfahren
- Kann Outlier detektieren
- Zwei Parameter:
  - epsilon: maximale Distanz, die zwei benachbarte Punkte in einem Cluster besitzen dürfen
  - minPoints: die minimale Anzahl an Datenpunkten, die zusammen ein Cluster bilden dürfen

# DBSCAN

## ALGORITHM 2: Abstract DBSCAN Algorithm

- |   |  |                         |
|---|--|-------------------------|
| 1 | Compute neighbors of each point and identify core points | // Identify core points |
| 2 | Join neighboring core points into clusters               | // Assign core points   |
| 3 | <b>foreach</b> non-core point <b>do</b>                  |                         |
| 4 | Add to a neighboring core point if possible              | // Assign border points |
| 5 | Otherwise, add to noise                                  | // Assign noise points  |

Schubert et al. 2017



# DBSCAN

- **Vorteile:**
  - Kann sehr gut mit Outlier umgehen
  - Anzahl von Clustern muss nicht vorgegeben werden
- **Nachteile:**
  - Kann nicht gut mit Clustern unterschiedlicher Densities umgehen
  - Probleme bei hochdimensionalen Daten

# REPRESENTATION LEARNING

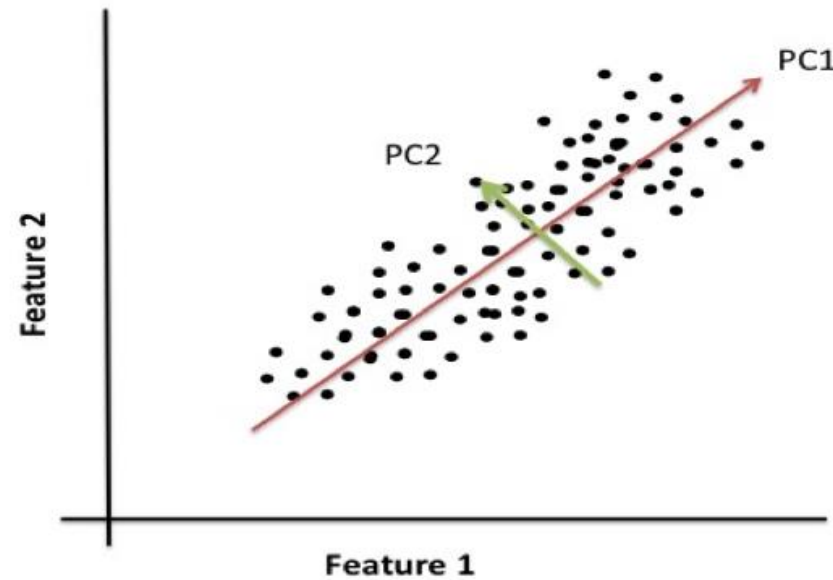
- Bisher haben wir nur Unsupervised-Learning-Algorithmen gesehen, die Cluster finden
- Es gibt aber auch Unsupervised Algorithmen, die neue Datenrepräsentationen lernen
- Ermöglicht das Erlernen von niedrig-dimensionalen (kompakteren) Repräsentationen
- Z.B. PCA, Autoencoder

# PRINCIPAL COMPONENT ANALYSIS (PCA)

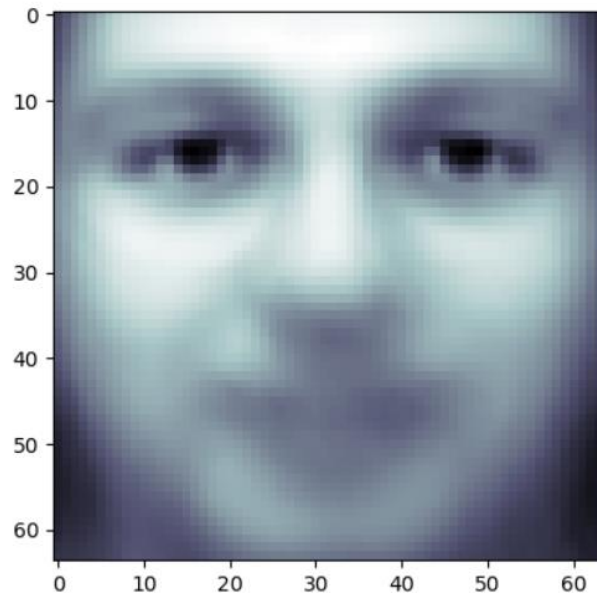
- Ziel: Finde eine kompaktere (niedriger dimensionale) Feature-Repräsentation für Daten
- Schritt 1: Subtrahiere den Mittelwert pro Feature von jedem Feature
- Schritt 2: Bestimme die Kovarianzmatrix dieser Daten
- Schritt 3: Bestimme die Eigenvektoren und dazugehörige Eigenwerte dieser Matrix
- Schritt 4: Projiziere die Daten auf die  $m$  Eigenvektoren mit den höchsten Eigenwerten (über Matrixmultiplikation der Daten mit dem neuem Eigenvektorraum)
- Diese werden principal components genannt und decken pro Achse die meisten Variationen ab.



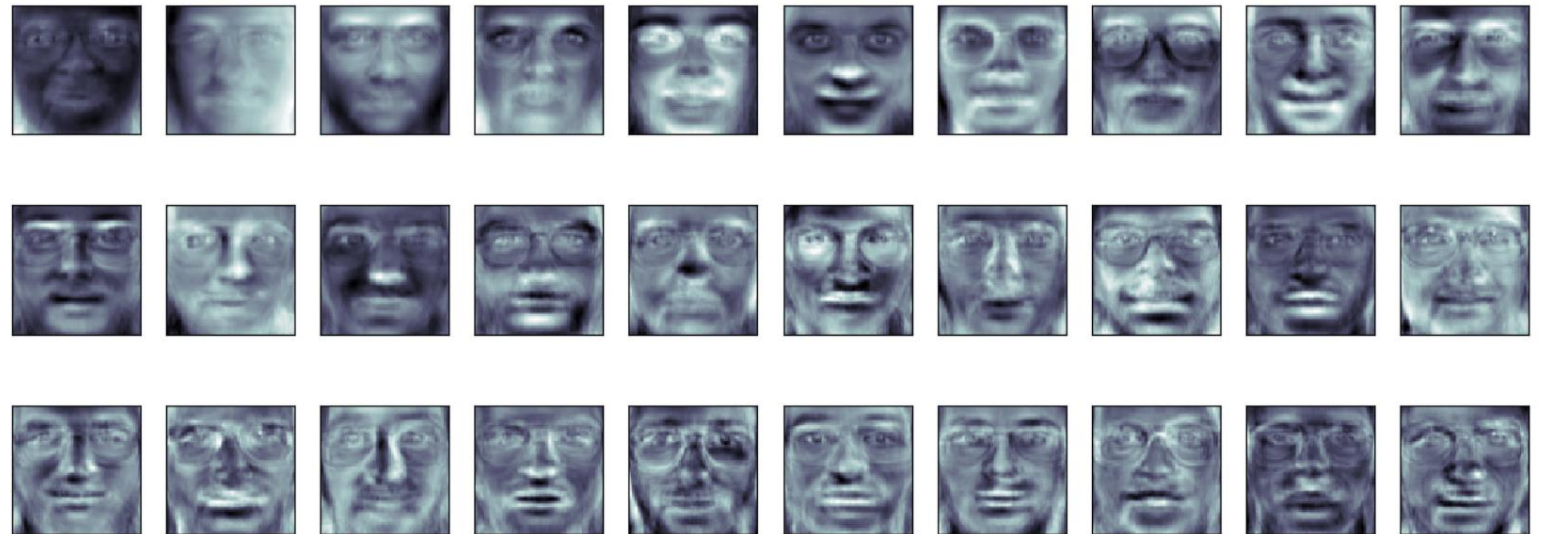
# PRINCIPAL COMPONENT ANALYSIS (PCA)



# PRINCIPAL COMPONENT ANALYSIS (PCA)



Durchschnittsgesicht



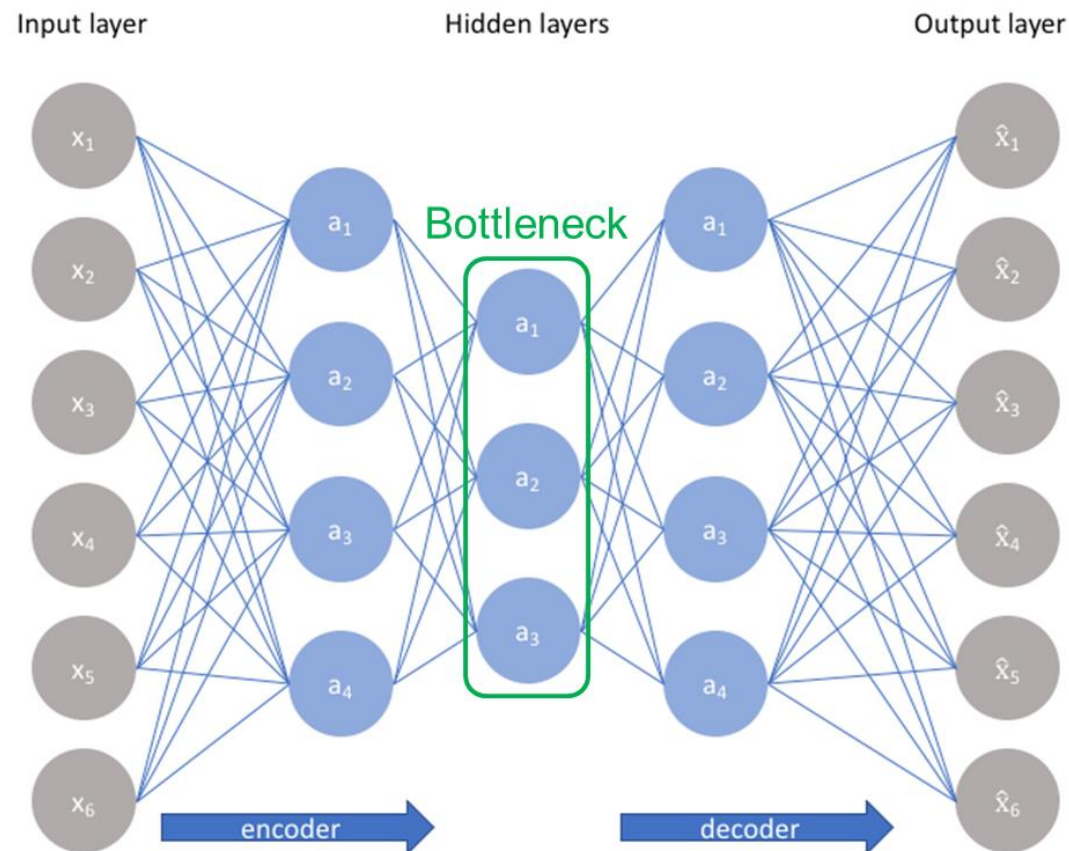
Eigenvektoren  
sortiert nach Einfluss von oben links nach unten rechts

[https://scipy-lectures.org/packages/scikit-learn/auto\\_examples/plot\\_eigenfaces.html](https://scipy-lectures.org/packages/scikit-learn/auto_examples/plot_eigenfaces.html)

# AUTOENCODER

- Neuronales Netz, das für das Kodieren komprimierter Repräsentationen genutzt wird.
- Idee: Im Training versucht man, den Input als Output zu rekonstruieren und verwendet dabei eine Netzstruktur mit einem Bottleneck. Die Features in diesem Bottleneck stellen eine komprimierte Repräsentation dar.
- Im Training: Encoder → komprimierte Repräsentation → Decoder
- Im Anwendungsfall: Encoder → komprimierte Repräsentation

# AUTOENCODER



# AUTOENCODER VS PCA

## Autoencoder

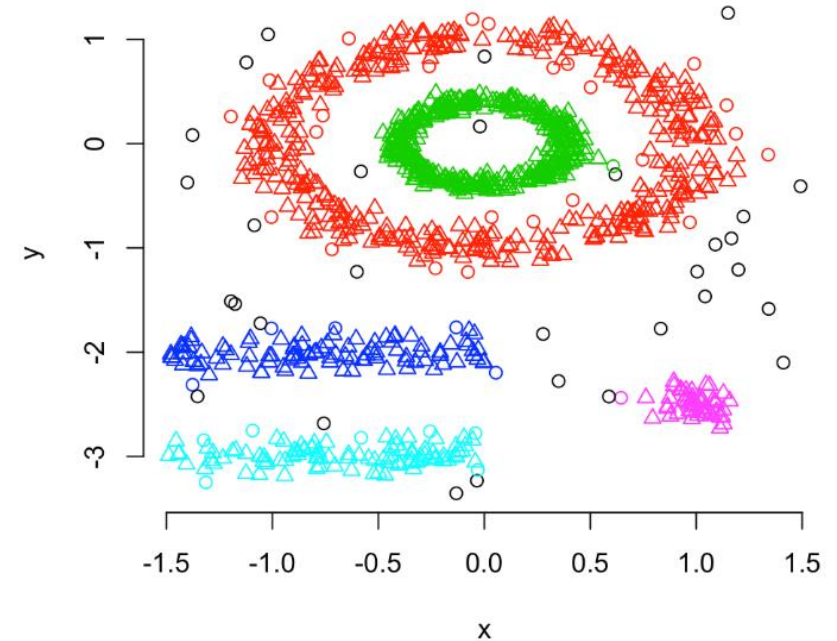
- + Lernen nicht-linearer Ebenen ist möglich
- Komplizierte Hyperparameteroptimierung notwendig
- Aufwendiges Training

## PCA

- + Einfaches finden der Hyperparameter
- + stabileres Training
- Nur lineare Komponenten lernbar (ohne Kernel-Trick)

# ANOMALIE-DETEKTION

- Anomalie-Detektion beschreibt den Prozess, unerwartete Gegenstände oder Events in einem Datenset zu finden.
- Besonderheiten von Anomalien:
  - Sie treten sehr selten auf
  - Sie unterscheiden sich signifikant von ihren Eigenschaften
- Anomalie-Detektion haben wir bereits mit DBSCAN kennengelernt







KAPITEL 1.3

# WEITERE LERNKONZEPTE

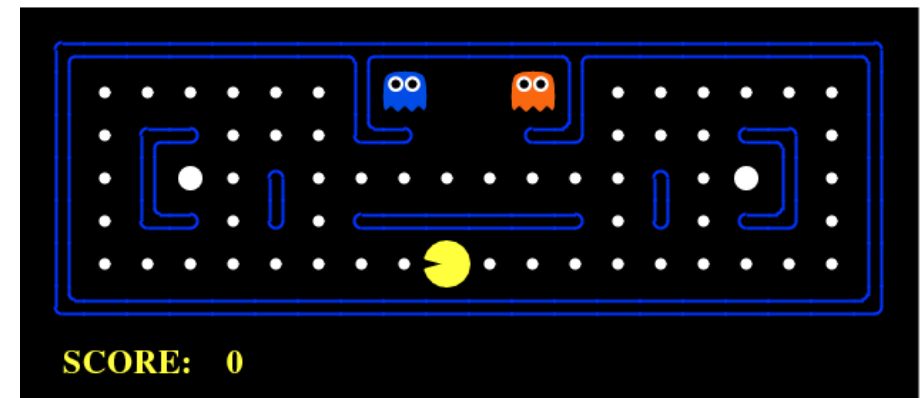
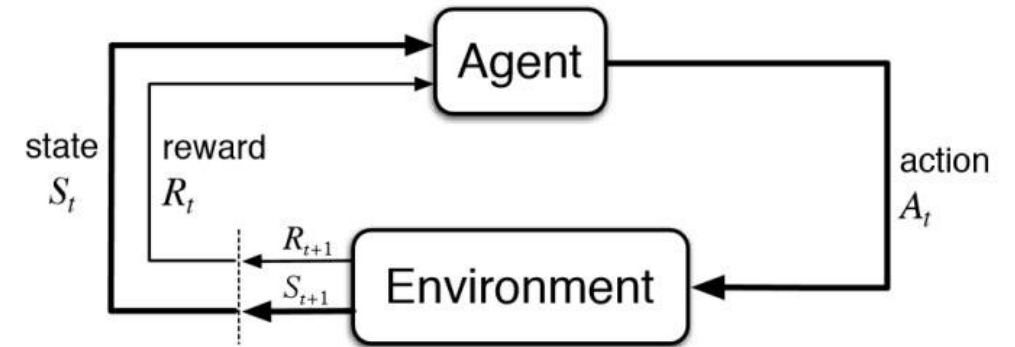
# SEMI-SUPERVISED LEARNING

- In den meisten Fällen ist es sehr aufwändig, Labels für seine Datenpunkte zu bekommen.
- Die Folge ist, dass man nur einen Teil der Daten labeln kann, man aber möglichst alle Daten zum Training nutzen möchte.
- Diese Art von Lernalgorithmen wird als semi-supervised Learning bezeichnet.
- Sie sind in der Lage, Daten mit und ohne Labels zum Trainieren zu verwenden, um ein deutlich besseres Ergebnis zu erlangen, als wenn man nur die gelabelten Daten verwendet.



# REINFORCEMENT LEARNING

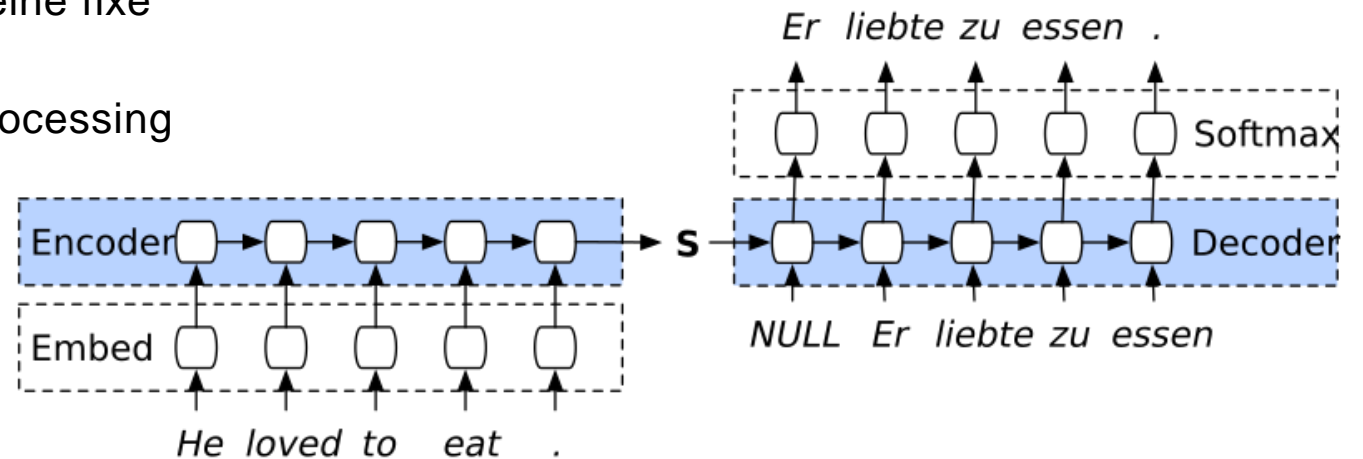
- Reinforcement Learning erlaubt es, das Verhalten (actions) eines Agenten in einer interaktiven Umgebung zu lernen
- Dabei lernt es mit Trial und Error über ein Feedback (reward) Mechanismus
- Erlaubt das Erlernen von Situationen und Spielen wie Schach, Go, Pacman und vielen mehr...
- Aml-Situation: Staubsaugroboter, der alleine die Wohnung saugt



[kdnuggets.com/2018/03/5-things-reinforcement-learning](https://kdnuggets.com/2018/03/5-things-reinforcement-learning)

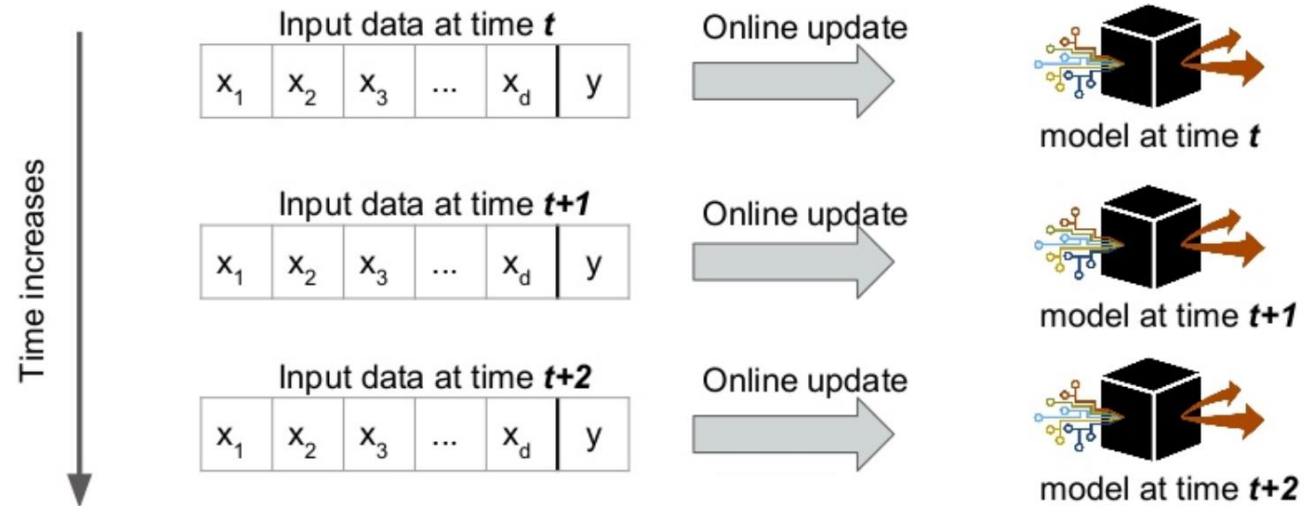
# SEQUENCE LEARNING

- Beim sequentiellen Lernen wird eine Sequenz von Daten in das Modell gegeben (anstatt von Feature-Vektoren fixer Größe)
- Diese Daten haben eine beliebige Länge, aber eine fixe Anordnung!
- Beispiele können aus dem Natural Language Processing oder aus der Sensorik kommen:
  - Frage-Antwort Probleme
  - Auswertung von Sensordaten



# ONLINE-LEARNING

- Im Klassische ML wird ein Modell vor der Laufzeit trainiert und bleibt während der Laufzeit unverändert
- Im Online-Learning lernt das Modell während der Laufzeit weiter



[slideshare.net/queirozfc/online-machine-learning-introduction-and-examples](https://www.slideshare.net/queirozfc/online-machine-learning-introduction-and-examples)

# ACTIVE LEARNING (AL)

- Active Learning ist ein Spezialfall von Semi-Supervised Learning
- Ein AL Modell ist in der Lage, interaktiv mit dem Nutzer zu kommunizieren, um Labels für neue Datenpunkte zu erhalten
- Dies erlaubt es dem Modell, mit wenig Datenpunkten eine hohe Aussagekraft (z.B. durch eine Gute Decision Boundary) zu erlernen.

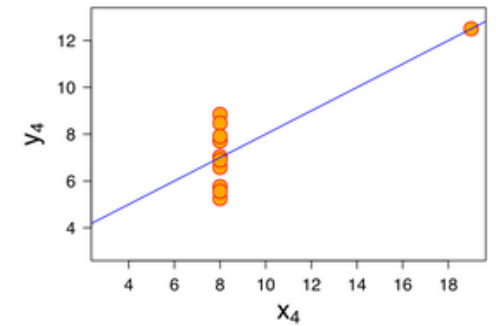
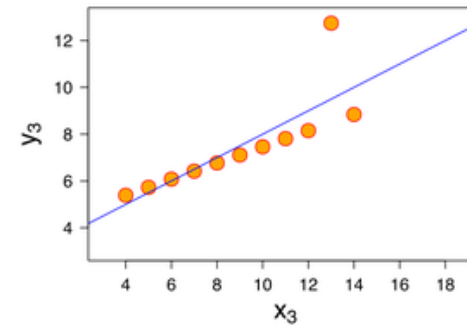
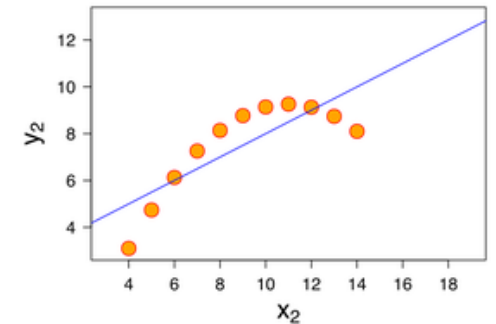
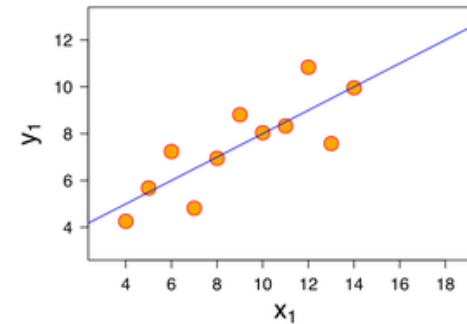


KAPITEL 2

# TIPPS UND TRICKS

# ANSCOMBE'S QUARTETT

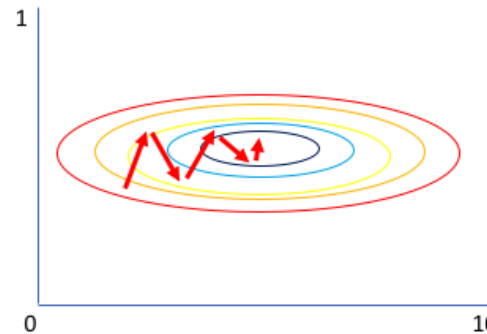
- Alle vier Verteilungen besitzen den selben Mittelwert, Standardabweichung und Korrelationswerte
- Aber die wichtigsten Information fehlen!
- Die Annahme einer Normalverteilung ist eine Vereinfachung, deren Sinnhaftigkeit immer überprüft werden sollte!



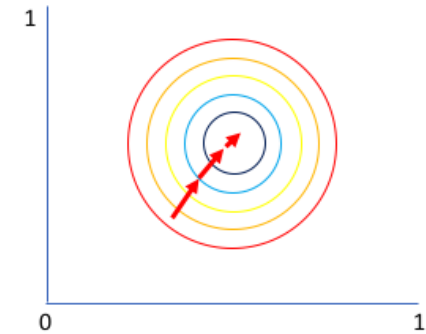
# FEATURE SCALING

- Verbessert die Trainingseffizienz
- Schnelleres Training und einfacheres Finden von Optima
- Min-Max Normalisierung
- Bringt jedes Feature in den Wertebereich [0,1]

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$



Gradient of larger parameter  
dominates the update

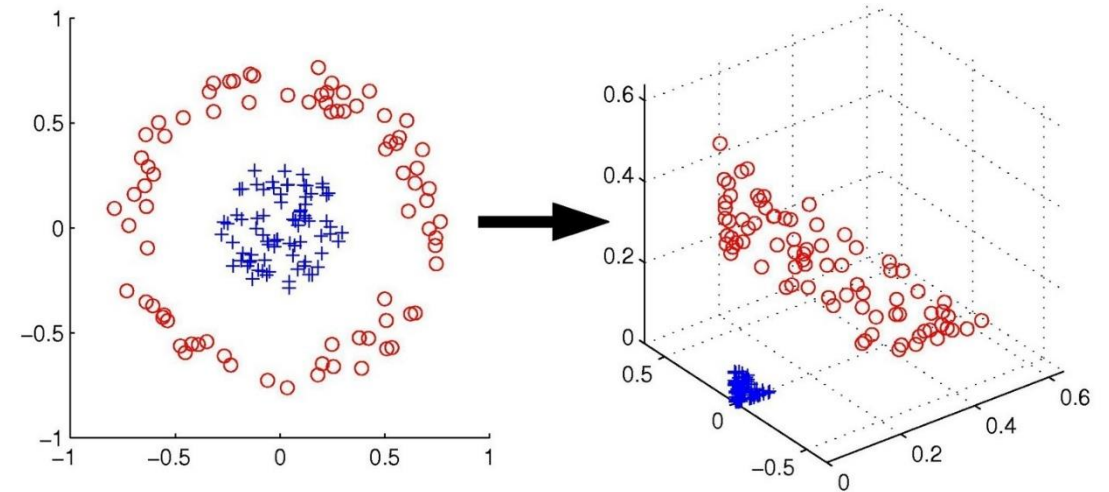


Both parameters can be  
updated in equal proportions

Towardsdatascience.com

# KERNEL TRICK

- Löse nicht-linear Separierbarkeit:  
Transformiere Daten in einen höherdimensionalen Raum,  
in dem das Problem linear separierbar ist.
- Problem: Das Auffinden einer geeigneten Transformation ist  
aufwändig.





# KERNEL TRICK

- Lösung: Kernel Trick
- Ersetze Skalarprodukt  $x^T y$  durch einen Kernel  $K(x, y)$ 
  - Ein Kernel ist eine Ähnlichkeitsfunktion
- Ohne weiteren Rechenaufwand werden alle Transformationen abgedeckt, deren Skalarprodukt den Kernel ergeben!
- Beispiel: Polynomial-Kernel zweiten Grades

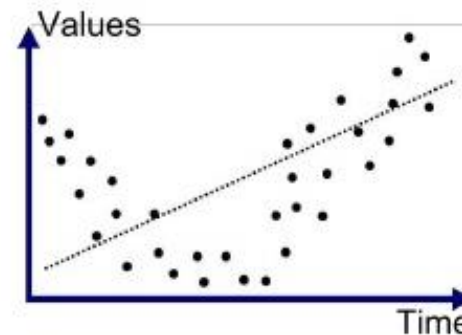
$$\hat{L}(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (\phi(x_j)^T \phi(x_i))$$

$$K(x, z) = \phi(x)^T \phi(z)$$

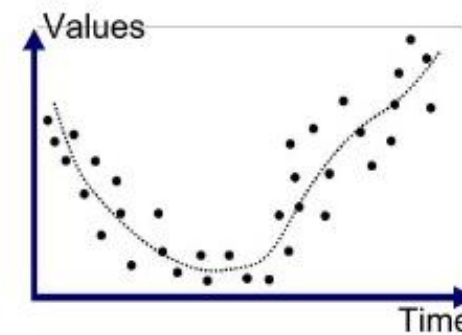
$$K(x, y) = (x^T y)^2 = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}^T \cdot \begin{bmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_1^2 - x_2^2 \\ 2x_1x_2 \\ x_1^2 + x_2^2 \end{bmatrix}^T \cdot \begin{bmatrix} y_1^2 - y_2^2 \\ 2y_1y_2 \\ y_1^2 + y_2^2 \end{bmatrix} = \phi(x)^T \cdot \phi(y)$$

# BIAS-VARIANCE TRADEOFF

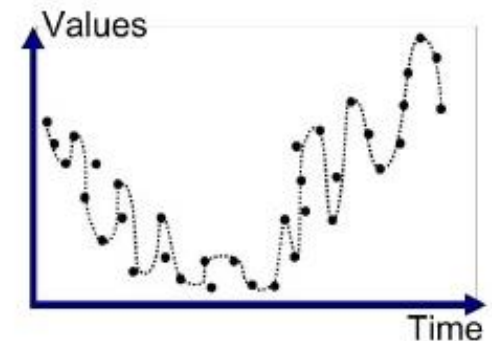
- **High Bias / Low Variance:**
  - Modell zu einfach
  - Komplexität der Daten wird nicht erfasst
- **Medium Bias / Medium Variance:**
  - Guter Tradeoff
- **Low Bias / High Variance:**
  - Modell ist zu komplex
  - Datenpunkte werden auswendig gelernt  
→ Overfitting



high bias  
low variance



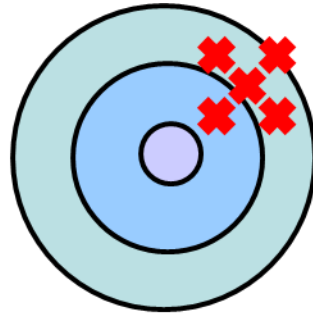
medium bias  
medium variance



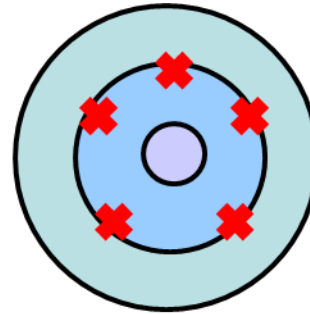
low bias  
high variance

# BIAS-VARIANCE TRADEOFF

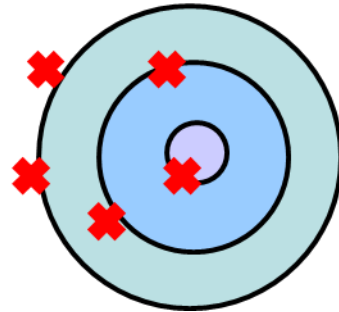
High Bias  
Low Variance



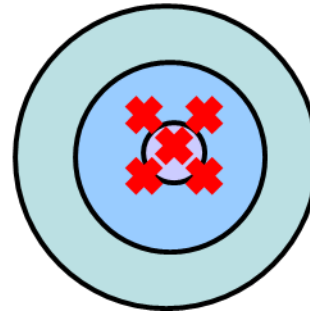
Low Bias  
High Variance



High Bias  
High Variance

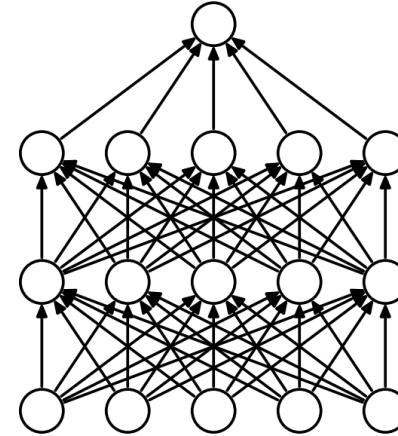


Low Bias  
Low Variance

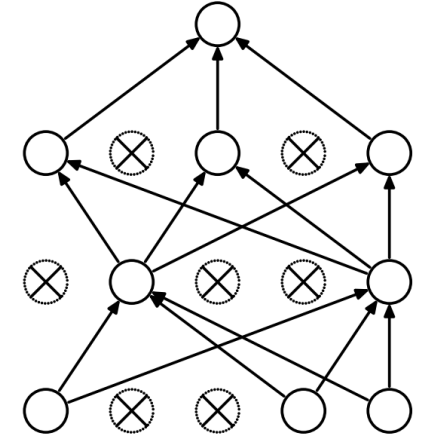


# REGULARISIERUNG - DROPOUT

- Idee
  - Trainiere pro Batch zufällige Subnetze
  - Subnetze werden durch das zufällige Löschen ( $p=0.5$ ) von Knoten und deren Kanten erstellt
  - Nach jeder Epoche werden die Gewichte gemittelt
- → Effekt: Intelligenz der Masse / Ensemble Learning
- → Ein Netz lernt zu Generalisieren



(a) Standard Neural Net



(b) After applying dropout.

<http://jmlr.org/papers/v15/srivastava14a.html>

# „DOUBLE-DESCENT“ PHÄNOMEN

- Classic ML: folgt dem Bias-Variance Tradeoff
  - Zu „schwache“ Modelle können nicht vollständig die Struktur der Daten erfassen, während zu „mächtige“ Modelle overfitten und daher schlecht generalisieren.
- Praxis zeigt: Deep Learning Modelle sind oft massiv überparametrisiert und verbessern sich trotzdem auf dem Test-set.
- Eine Erklärung :
  - Deep Double Descent: Where Bigger Models and More Data Hurt (Nakkiran – Dez 2019)

# „DOUBLE-DESCENT“ PHÄNOMEN

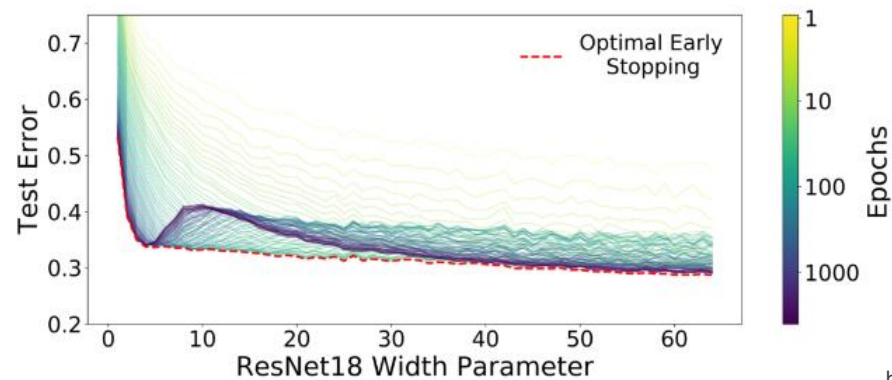
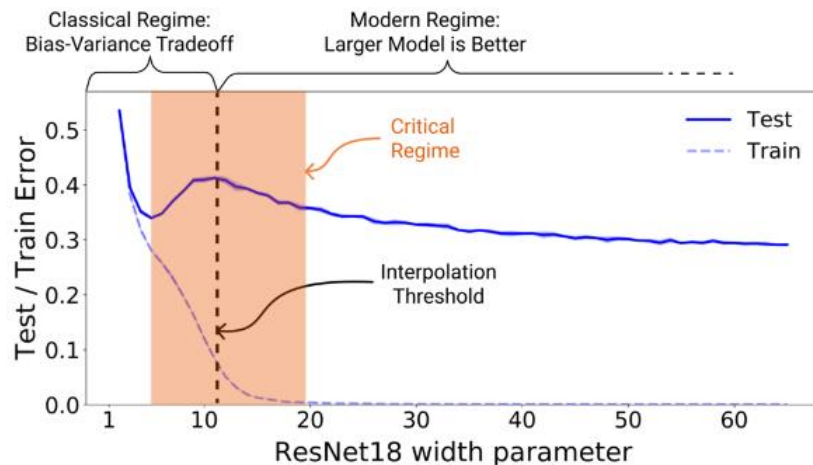
- By considering larger function classes, which contain more candidate predictors compatible with the data, we are able to find interpolating functions that have smaller norm and are thus “simpler”. Thus, increasing function class capacity improves performance of classifiers.

—Belkin et al. 2018

- Größere Modelle sind besser! (Aber aufwändiger zu trainieren)
- Gilt für DL Ansätze als auch für klassische ML Modelle wie z.B. Random Forest!

# „DOUBLE-DESCENT“ PHÄNOMEN

- DL Settings besitzt zwei relevante Regime:
  - Under-parametrized Regime: Modell Komplexität  $\ll$  #Trainingsdaten: U-Form Verhalten im Test-Error
  - Over-parametrized Regime: Modell Komplexität  $\gg$  # Trainingsdaten: Erreicht einen Trainingserror von fast 0 und dann verringert sich der Test-Error



<https://arxiv.org/pdf/1912.02292.pdf>





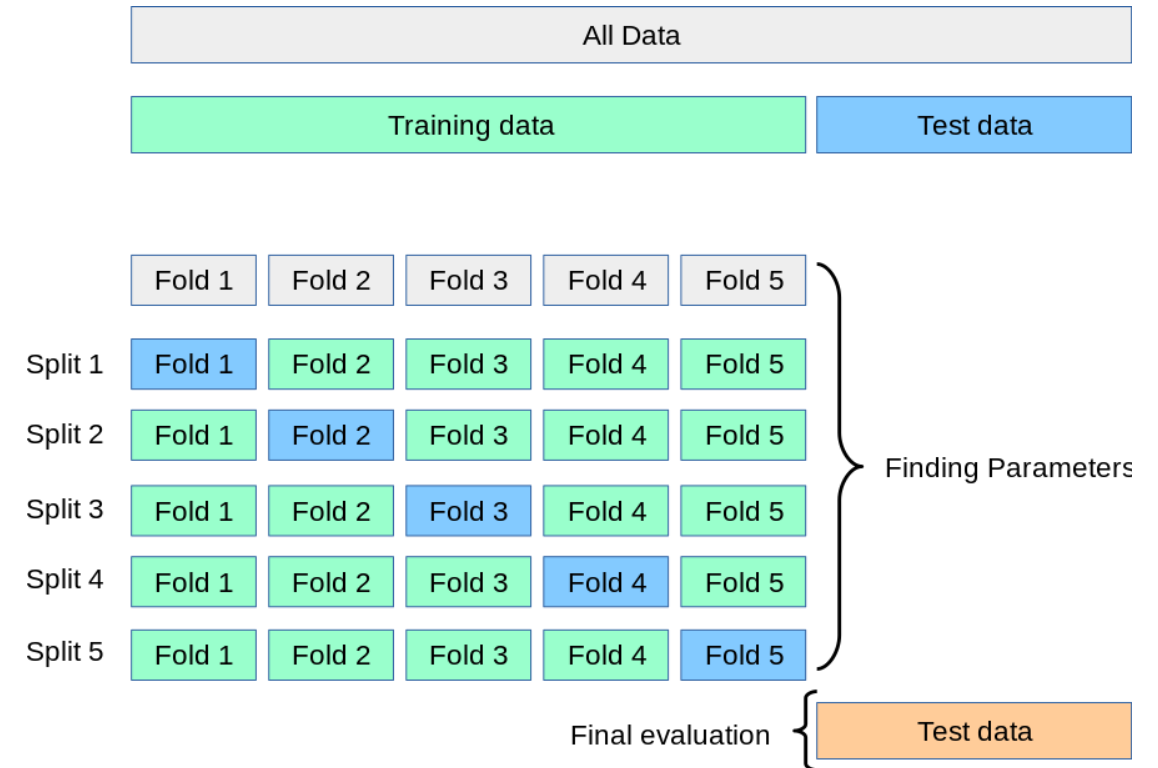
KAPITEL 3

# EVALUIERUNGSKONZEPTE



# K-FOLD CROSS-VALIDATION

- Um Overfitting zu vermeiden, sollten Training- und Testset immer unabhängig voneinander sein
- Um trotzdem das meiste aus seinen Daten rauszuholen:
- Cross-Validation:  
Daten werden in k Folds eingeteilt,
- Damit kann auch die mittlere Performanz mit Standardabweichung angegeben werden. (Mean und STD über die k Folds)



Scikit-learn.org



**AMBIENT INTELLIGENCE | TIPPS UND TRICKS**

# **EVALUATION VON KLASSIFIKATIONSPROBLEMEN**

# KONFUSIONSMATRIX

- Visualisiert die Performanz eines Modells

	positive classified	negative classified	
is positive	true positives ( $tp$ )	false negatives ( $fn$ )	$tp + fn = P$
is negative	false positives ( $fp$ )	true negatives ( $tn$ )	$fp + tn = N$
	$tp + fp$	$fn + tn$	$P + N =  E $

- True Positive: Das Label ist positiv und auch die Vorhersage
- True Negative: Das Label ist negativ und auch die Vorhersage
- False Positive: Das Modell predicted ein negatives Sample positiv
- False Negative: Das Modell predicted ein positives Sample negativ

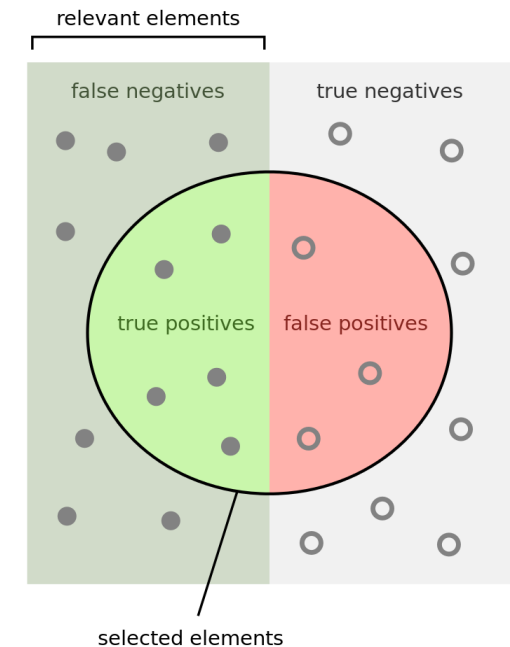
# PRECISION UND RECALL

- Precision
  - Anteil der positiven Vorhersagen, die korrekt sind
  - Fähigkeit des Modells nur relevante Elemente auszugeben

$$Precision = \frac{TP}{TP + FP}$$

- Recall
  - Anteil der positiven Daten, die korrekt vorhergesagt wurden
  - Fähigkeit des Modells, alle relevanten Elemente zu identifizieren

$$Recall = \frac{TP}{TP + FN}$$



How many selected items are relevant?

Precision =  $\frac{\text{green}}{\text{green} + \text{red}}$

How many relevant items are selected?

Recall =  $\frac{\text{green}}{\text{green} + \text{light blue}}$

# ROC-KURVE

- Receiver Operating Characteristic (ROC) Kurve
- Erlaubt es, einen binären Klassifizierer für alle Thresholds gleichzeitig zu untersuchen
- X-Achse: False Positive Rate (FPR)

- Wahrscheinlichkeit, dass ein negatives Sample fälschlicherweise als positiv klassifiziert wird

$$FPR = \frac{FP}{FP + TN}$$

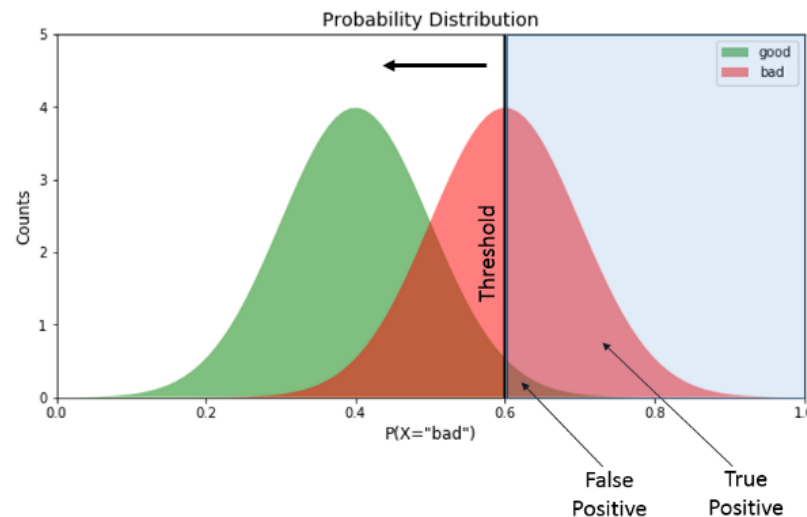
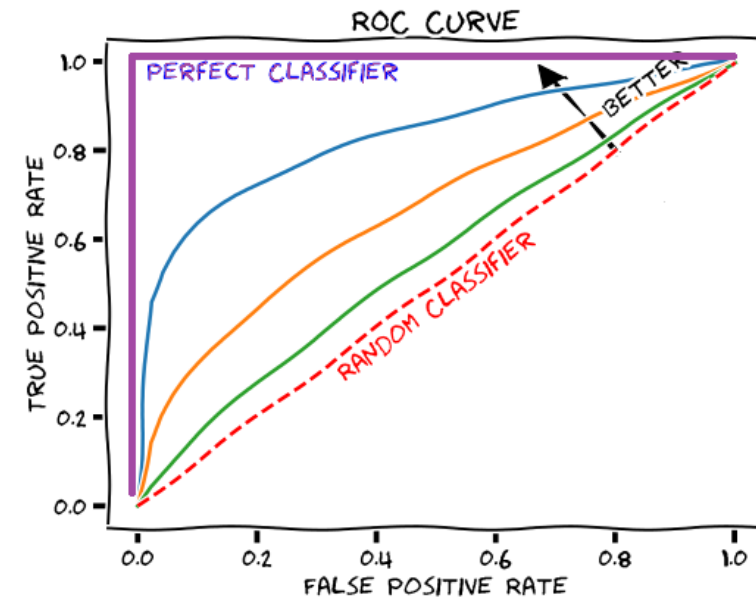
- Y-Achse: True Positive Rate (TPR)

- Wahrscheinlichkeit, dass ein positives Sample korrekt als positiv klassifiziert wird

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

# ROC-KURVE

- Area Under the Curve (AUC)
- Gibt den Flächeninhalt unter der ROC-Kurve an
- Allgemeines Evaluationskriterium eines binären Klassifizierers
- Gibt die Wahrscheinlichkeit an, dass ein positives und ein negatives Sample richtig erkannt werden





KAPITEL 4

# ZUSAMMENFASSUNG

# ZUSAMMENFASSUNG

- Maschinelles Lernen beschreibt die Erstellung einer Hypothese auf Basis von Daten
    - ML ist ein Optimierungsproblem!
  - Verschiedene maschinelle Lernkonzepte:
    - Unsupervised, Supervised, und viele mehr...
  - Evaluationskonzepte für Klassifikation
  - Hilfreiche Hinweise beim Anwenden ML Algorithmen
  - Z.B. Bias-Variance Tradeoff, Double-Descent, Kernel Trick
- 
- Dies war nur ein kleiner Überblick eines riesigen Feldes!



# LITERATUR

- Marc Peter Deisenroth et al.: ***Mathematics for Machine Learning***, <https://mml-book.github.io>
- Kevin Murphy: ***Probabilistic Machine Learning: An Introduction***, <https://probml.github.io/pml-book/book1.html>
- Kevin Murphy: ***Probabilistic Machine Learning: Advanced Topics***, <https://probml.github.io/pml-book/book2.html>
- Pedro Domingos: ***A Few Useful Things to Know About Machine Learning***, <https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf>
- Christopher M. Bishop: ***Pattern Recognition and Machine Learning***, <https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/>