Karlsruher Institut für Technologie

Grundlagen der Künstlichen Intelligenz

Wintersemester 25/26

Vorlesung 6

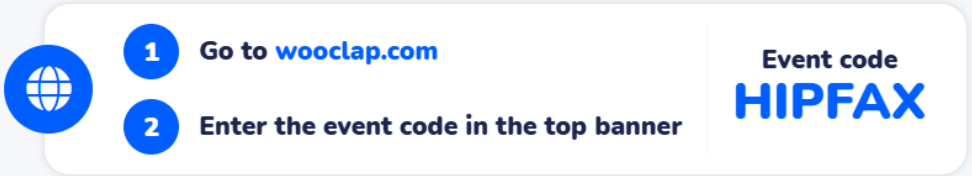
Generalisierung, Hyperparameter, Regularisierung, Ensembles

1. Dezember 2025

T.T.-Prof. Dr. Peer Nowack
Prof. Dr. Pascal Friederich

Wir werden immer mal wieder Umfragen machen...

Sie können sich bereits einloggen. Nutzen Sie das KIT Wi-Fi bei schlechtem Empfang.



Eine wichtige Anmerkung zu den Übungen...

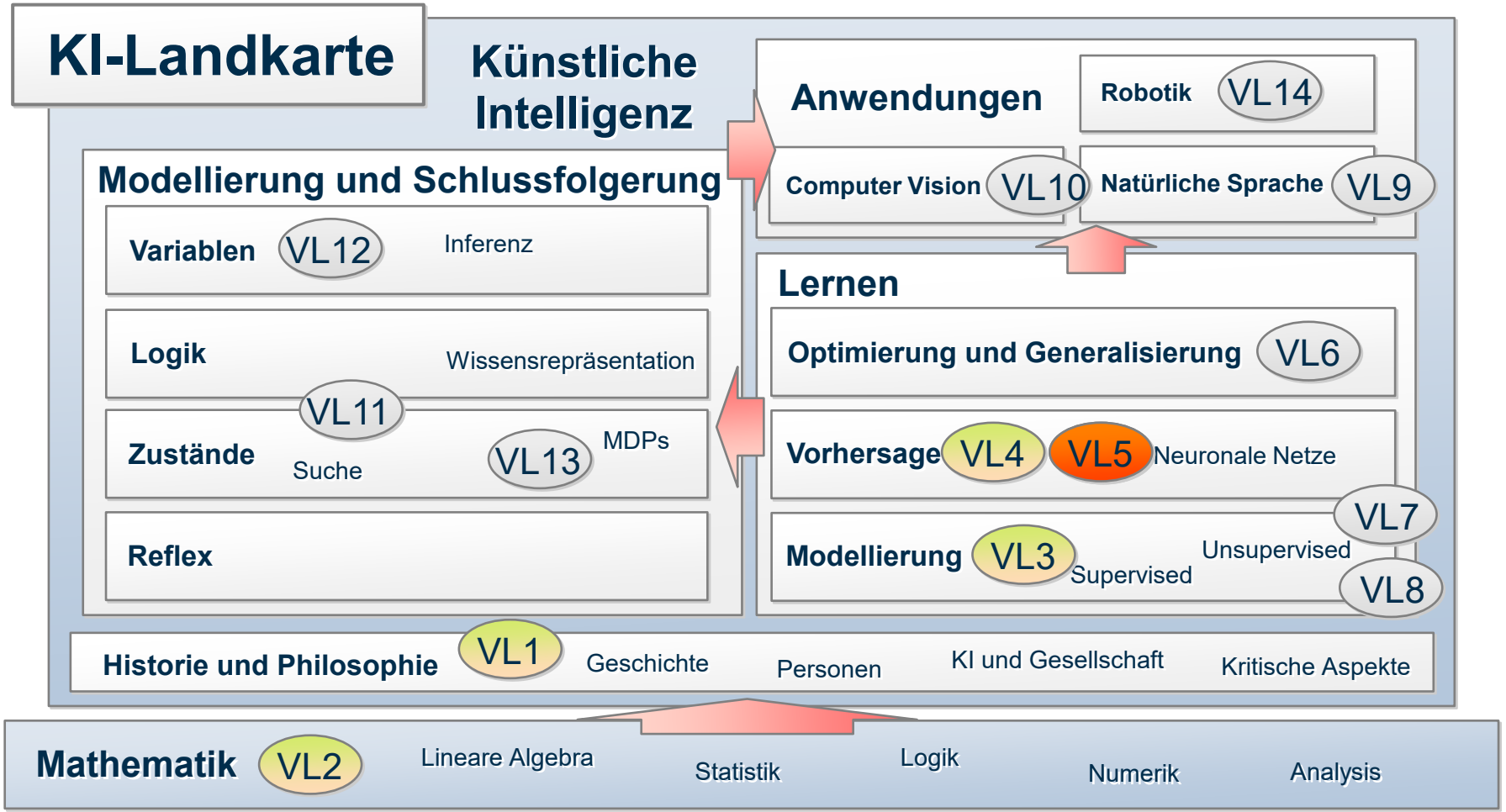
Sie können gerne in **Gruppen von bis zu drei Studierenden** arbeiten!

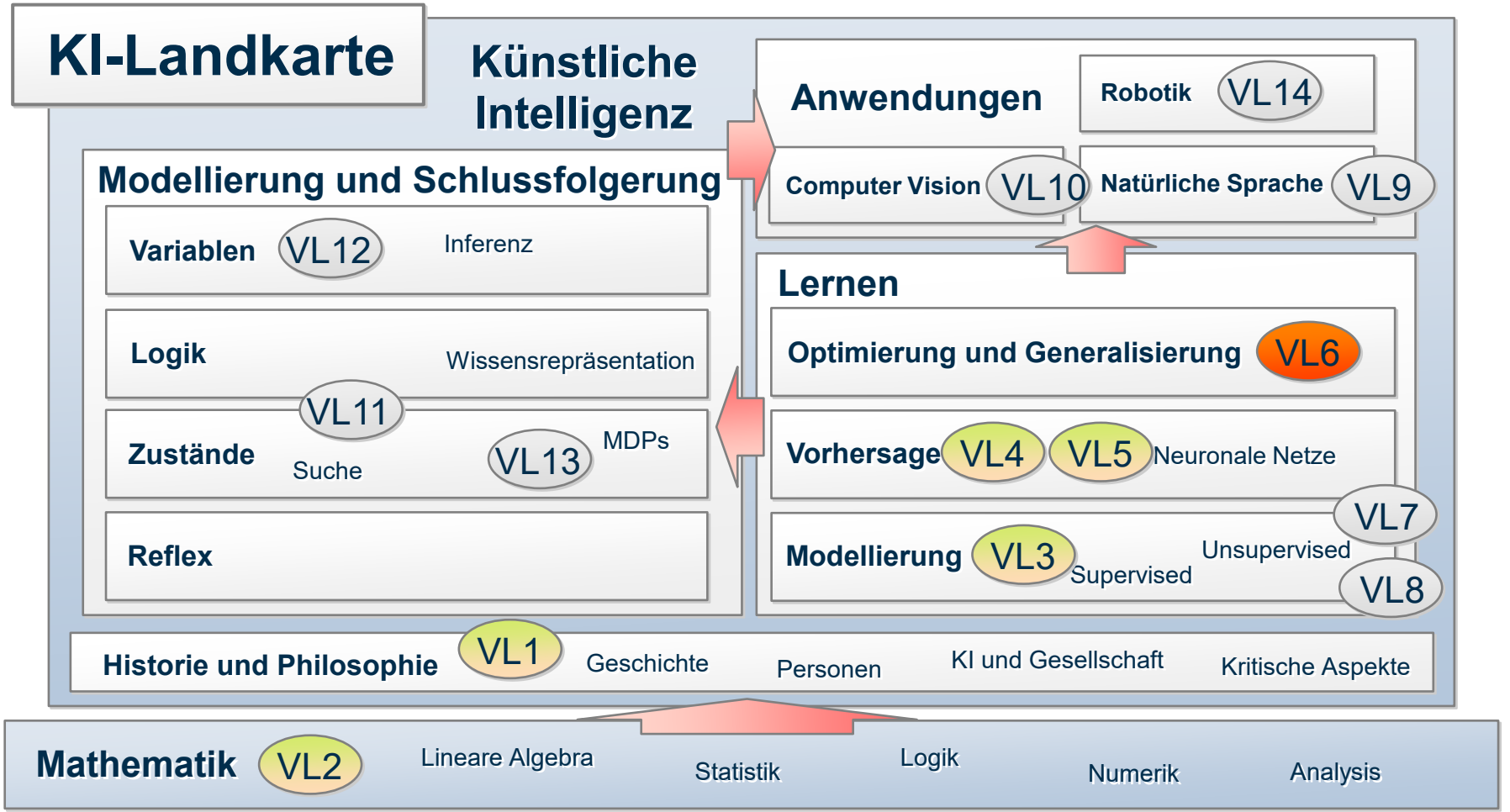
In dem Fall, bitte **im Jupyter Notebook** am markierten Ort ihre **u-Kürzel** angeben.

Sie müssen dennoch **jeweils eine Lösung einreichen.**

Die Markierung im Notebook hilft unserem Übungsteam den Korrekturprozess der handschriftlichen Übungen zu beschleunigen! (die gleichen Lösungen müssen nicht mehrfach korrigiert werden)

Vielen Dank!


KI-Landkarte
Künstliche Intelligenz
Anwendungen
Robotik
VL14
Computer Vision
VL10
Natürliche Sprache
VL9
Modellierung und Schlussfolgerung
Variablen
VL12
Inferenz
Logik
Wissensrepräsentation
VL11
Zustände
Suche
VL13
MDPs
Reflex
Lernen
Optimierung und Generalisierung
VL6
Vorhersage
VL4
VL5
Neuronale Netze
VL7
Modellierung
VL3
Supervised
Unsupervised
VL8
Historie und Philosophie
VL1
Geschichte
Personen
KI und Gesellschaft
Kritische Aspekte
Mathematik
VL2
Lineare Algebra
Statistik
Logik
Numerik
Analysis


KI-Landkarte
Künstliche Intelligenz
Anwendungen
Robotik
VL14
Computer Vision
VL10
Natürliche Sprache
VL9
Modellierung und Schlussfolgerung
Variablen
VL12
Inferenz
Logik
Wissensrepräsentation
VL11
Zustände
Suche
VL13
MDPs
Reflex
Lernen
Optimierung und Generalisierung
VL6
Vorhersage
VL4
VL5
Neuronale Netze
VL7
Modellierung
VL3
Supervised
Unsupervised
VL8
Historie und Philosophie
VL1
Geschichte
Personen
KI und Gesellschaft
Kritische Aspekte
Mathematik
VL2
Lineare Algebra
Statistik
Logik
Numerik
Analysis

Ziele der heutigen Vorlesung

Generalisierbarkeit auf neuen Testdaten ist für den Erfolg vom Maschinellen Lernen (ML) fundamental.

Nach dieser Vorlesung werden Sie in der Lage sein:

- Zu erklären, wie ML-Methoden auf Generalisierbarkeit geprüft und ausgewählt werden können, durch **Validierungsmethoden.**
- Beispiele für **Hyperparameter** zu beschreiben, die für die Generalisierbarkeit „getuned“ werden können.
- **Regularisierungstechniken** zu unterscheiden, die dafür genutzt werden können.
- Die praktische Bedeutung und den Erfolg von **Ensemble-Methoden** zu bewerten.

Überblick

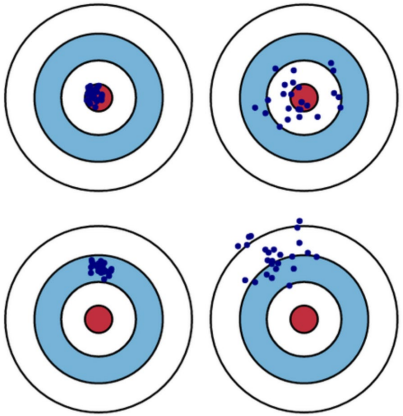
1) Generalisierung

2) Hyperparameter

3) Regularisierung

4) Ensemble Methoden

Teil 1



1) Generalisierung

2) Hyperparameter

3) Regularisierung

4) Ensemble Methoden

Generalisierung

- Herausforderung des maschinellen Lernens (ML): Generalisierung

→ Gute Voraussagen auch für ungesehene Daten (Fehler auf dem Testset)
→ **Generalisierungsfehler**

- Hauptunterschied zwischen ML und Optimierung
- Beispiel: Lineare Regression

Training: Minimiere $\frac{1}{m^{(\text{train})}} \left\| \boldsymbol{X}^{(\text{train})} \boldsymbol{w} - \boldsymbol{y}^{(\text{train})} \right\|_2^2$ durch Optimierung von $\boldsymbol{w}$

Eigentliches Ziel: Minimiere $\frac{1}{m^{(\text{test})}} \left\| \boldsymbol{X}^{(\text{test})} \boldsymbol{w} - \boldsymbol{y}^{(\text{test})} \right\|_2^2$

Lösung nur möglich unter der Annahme, dass beide Datensätze aus der gleichen Verteilung gezogen werden.

Kapazität, Überanpassung und Unteranpassung

Herangehensweise

1. Minimiere den Fehler auf den Trainingsdaten

→ Optimierung der Modellparameter, Backpropagation

2. Minimiere die Lücke zwischen Trainings- und Testdaten

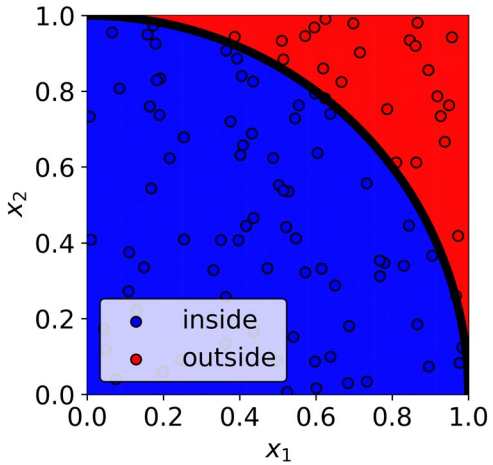
→ Optimierung des Modells selbst (Auswahl)

- Kontrolle des Generalisierungsfehlers: Anpassung der **Kapazität** des Modells
  - Kapazität: Möglichkeit eines Modells unterschiedliche Funktionen zu beschreiben
  - Niedrige Kapazität: Tendenz zur Unteranpassung
  - Hohe Kapazität: Tendenz zur Überanpassung

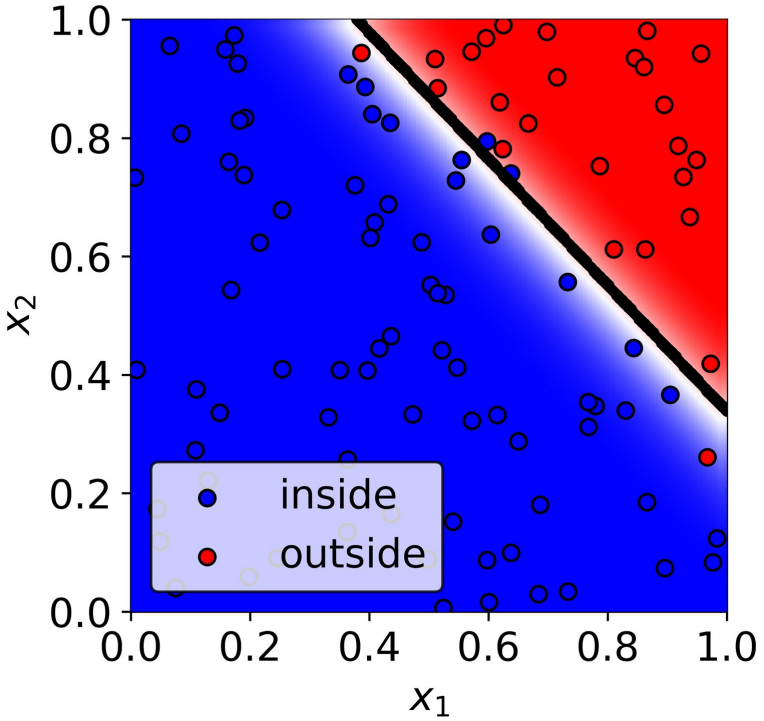
Kapazität, Überanpassung und Unteranpassung

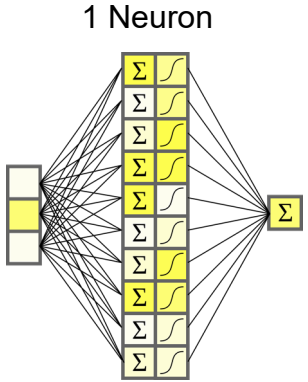
- **Beispiel**

Ziel



Tatsächliches Ergebnis: 1 Neuron

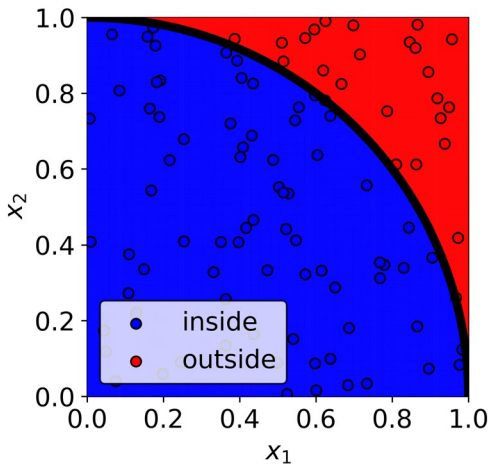




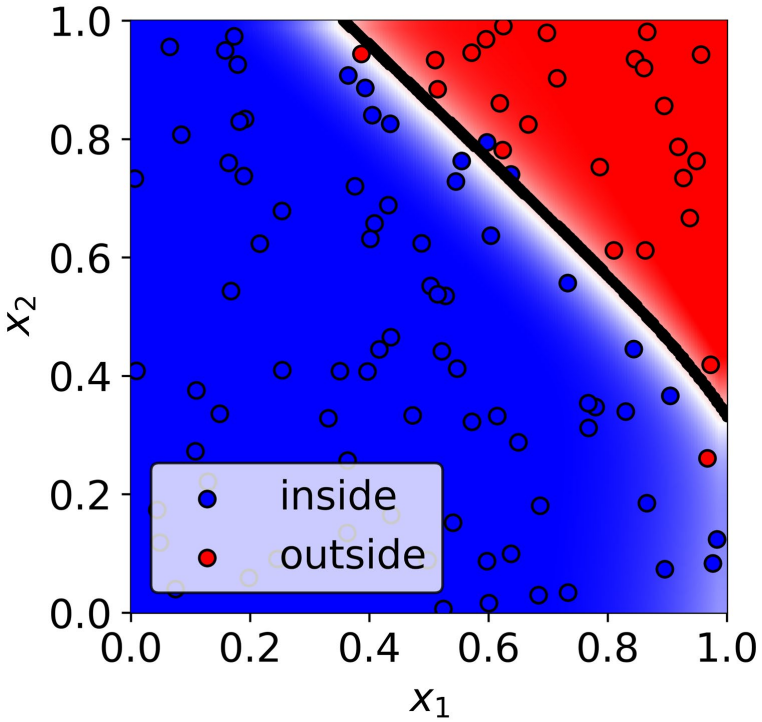
Kapazität, Überanpassung und Unteranpassung

- **Beispiel**

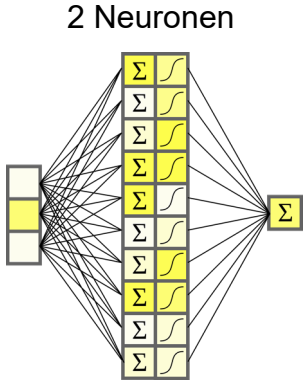
Ziel



Tatsächliches Ergebnis: 2 Neuronen



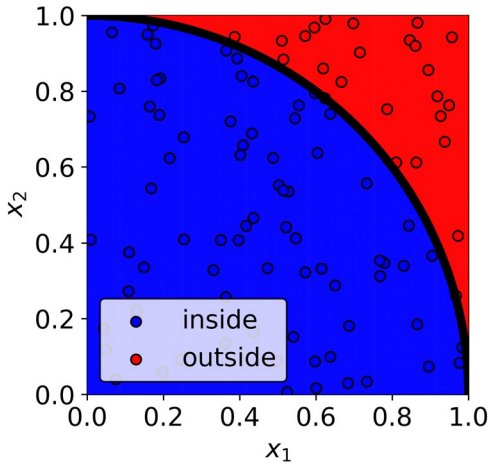
2 Neuronen



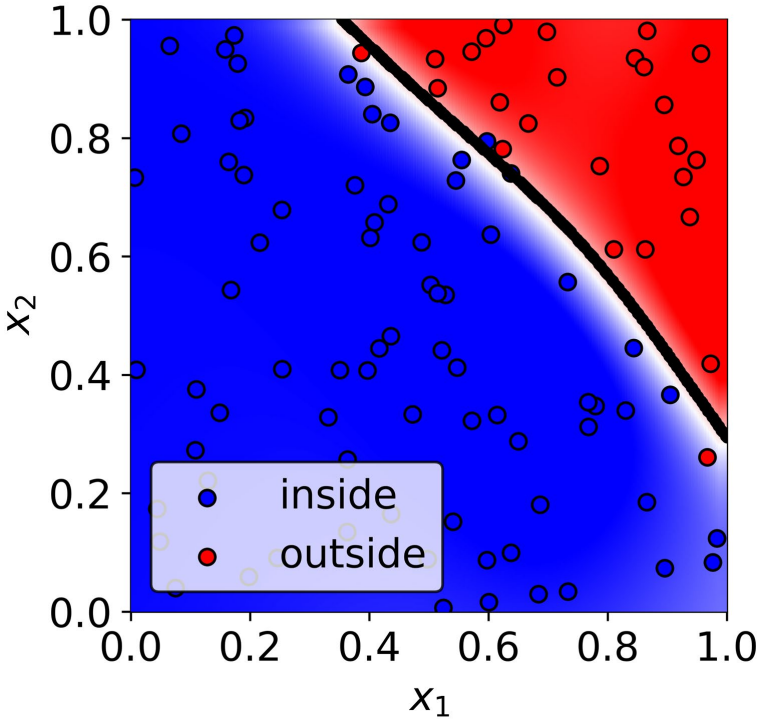
Kapazität, Überanpassung und Unteranpassung

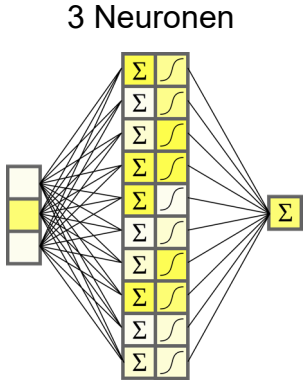
- **Beispiel**

Ziel



Tatsächliches Ergebnis: 3 Neuronen

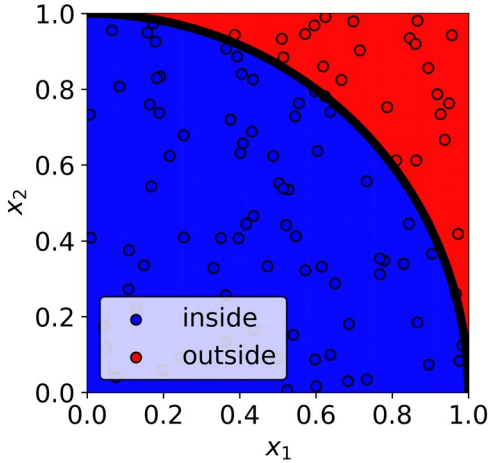




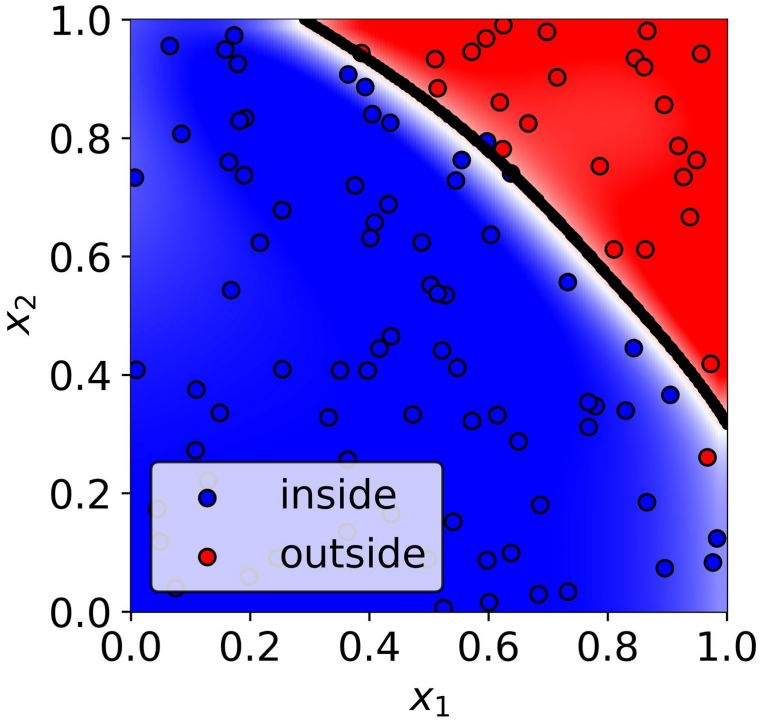
Kapazität, Überanpassung und Unteranpassung

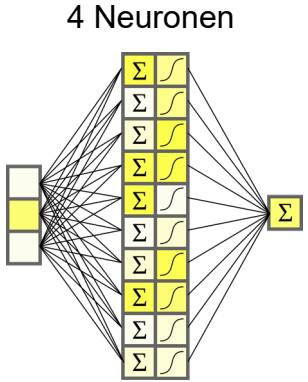
- **Beispiel**

Ziel



Tatsächliches Ergebnis: 4 Neuronen

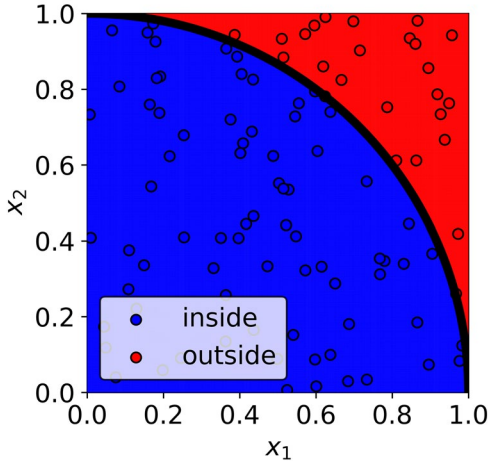




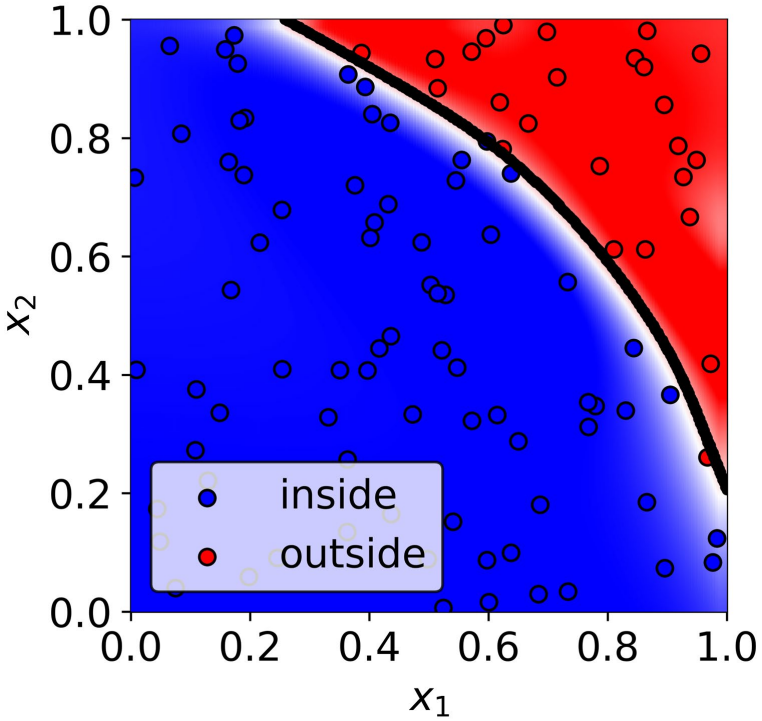
Kapazität, Überanpassung und Unteranpassung

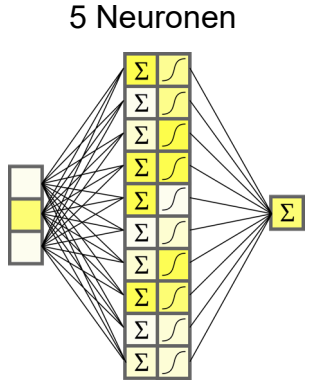
- **Beispiel**

Ziel



Tatsächliches Ergebnis: 5 Neuronen

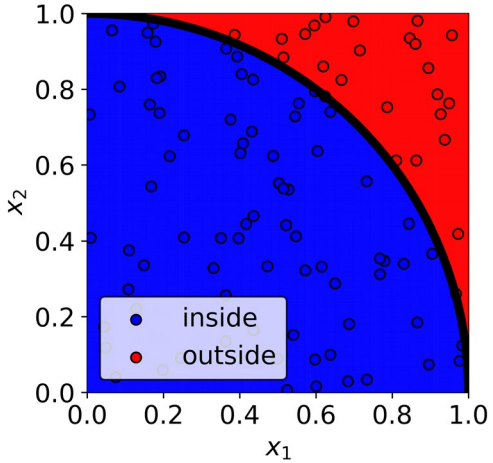




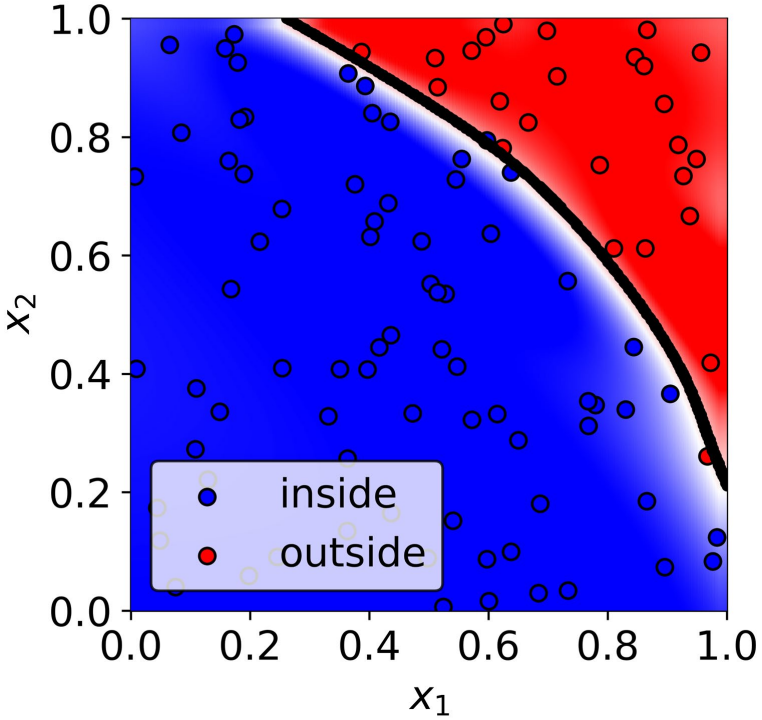
Kapazität, Überanpassung und Unteranpassung

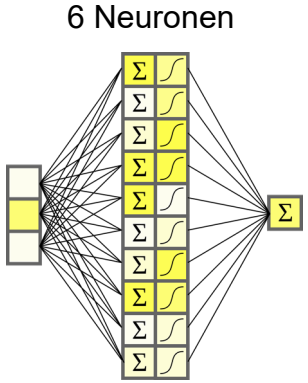
- **Beispiel**

Ziel



Tatsächliches Ergebnis: 6 Neuronen

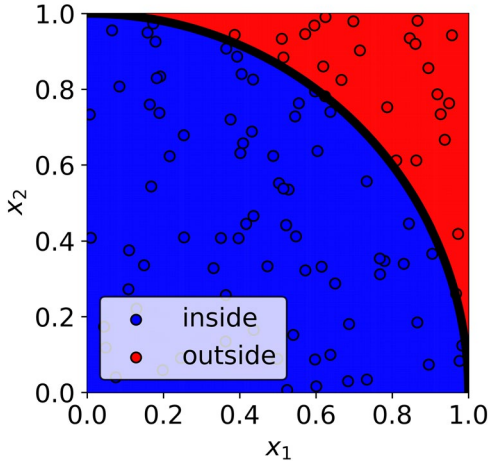




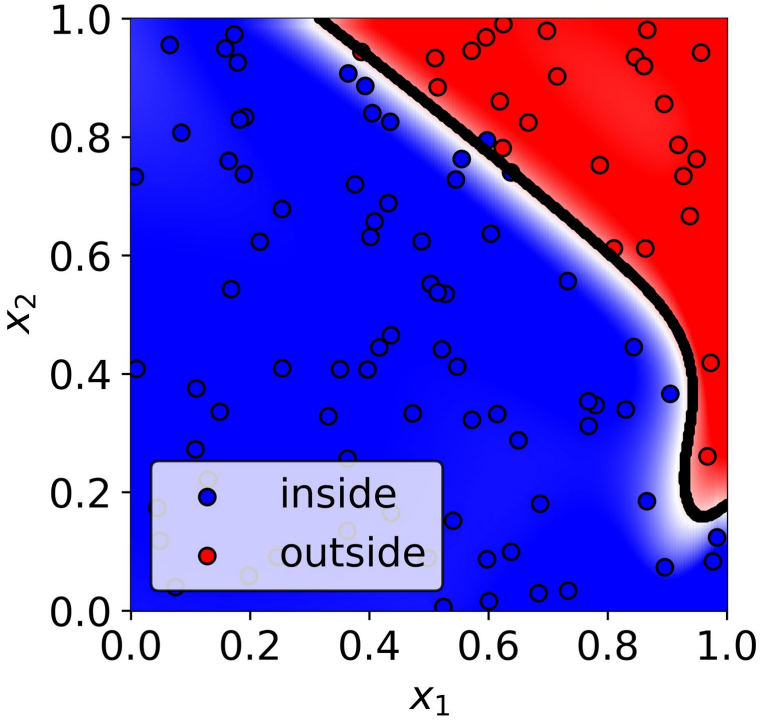
Kapazität, Überanpassung und Unteranpassung

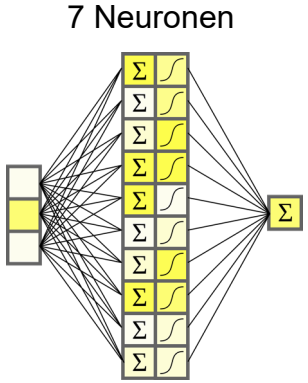
- **Beispiel**

Ziel



Tatsächliches Ergebnis: 7 Neuronen

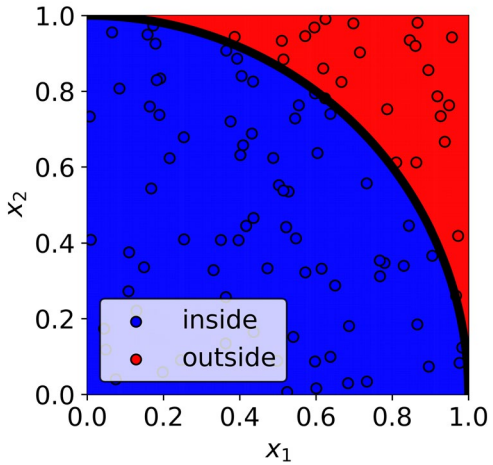




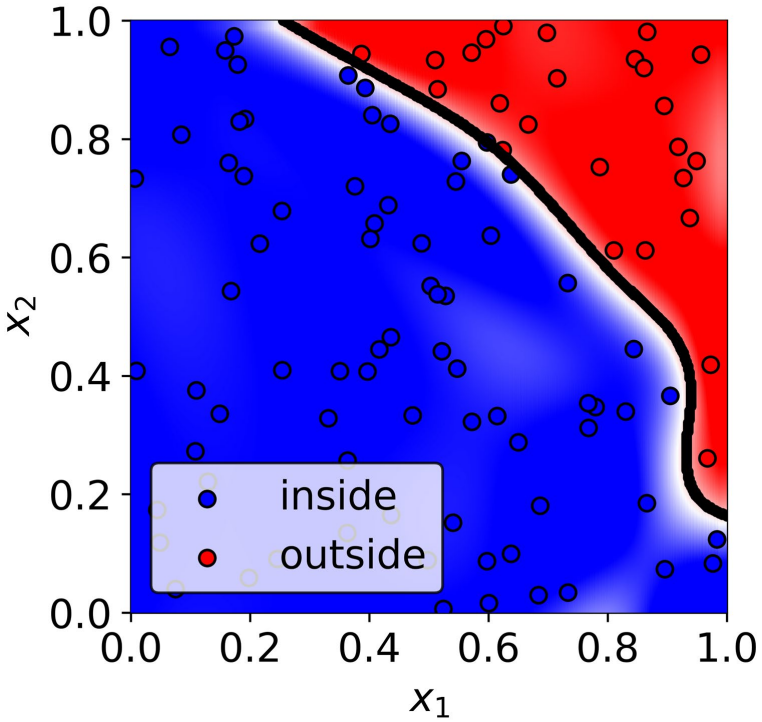
Kapazität, Überanpassung und Unteranpassung

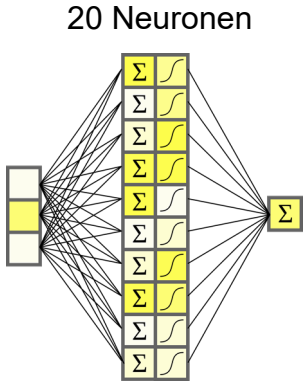
- **Beispiel**

Ziel

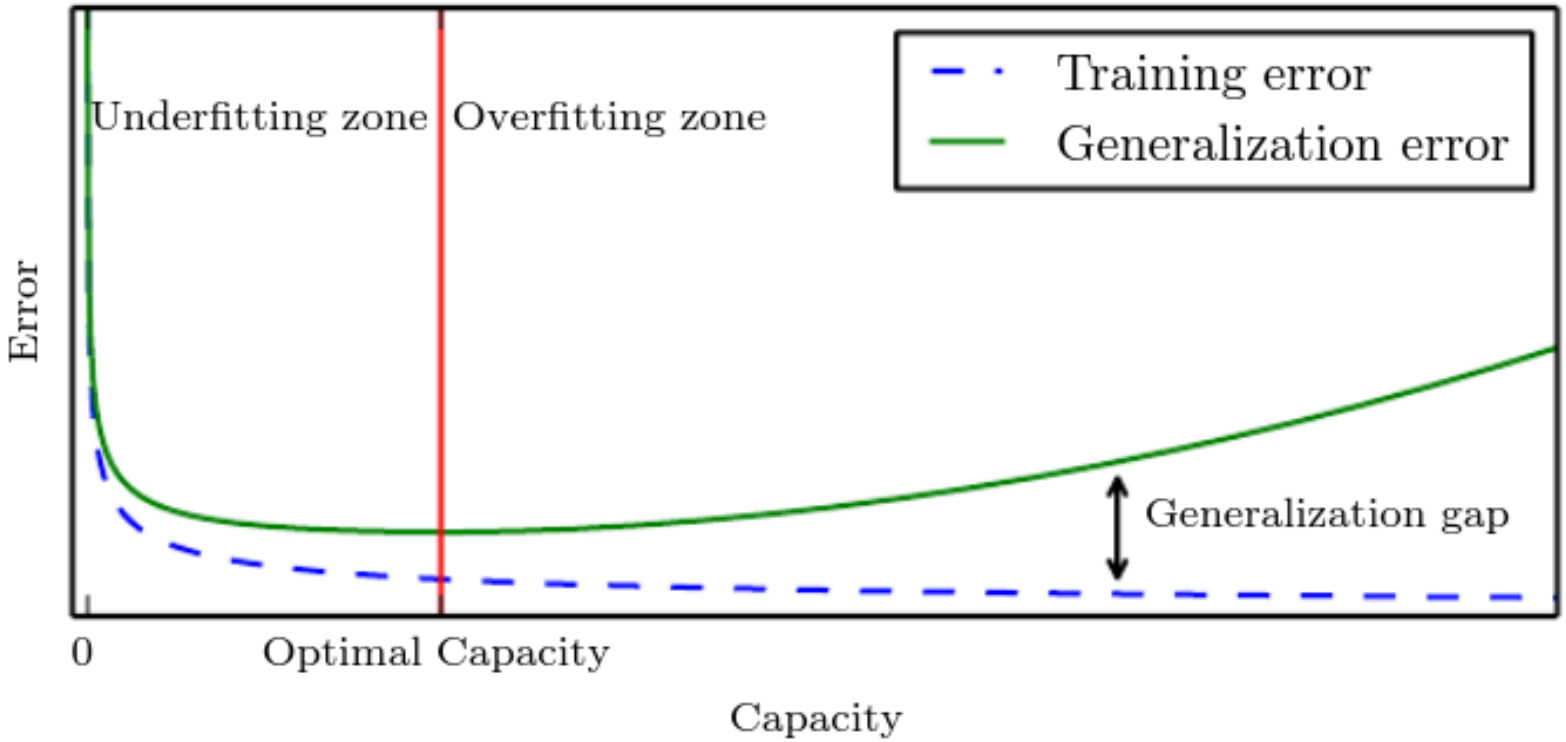


Tatsächliches Ergebnis: 20 Neuronen





Kapazität, Überanpassung und Unteranpassung



Einschub: Erwarteter Modellfehler, Bias und Varianz

Gegeben: Inputs $x_1, \ldots, x_n$

Labels $y_1, \ldots, y_n$ mit $y_i = f(x_i) + \epsilon$ $\left(\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)\right)$

Gesucht: Funktion $\hat{f}(x)$, welche die Verlustfunktion $L = (y - \hat{f}(x))^2$ minimiert.

Frage: Was ist der Erwartungswert von L :

$$\mathrm{E}[L] = \mathrm{E}\left[(y - \hat{f})^2\right]$$

Einschub: Bias und Varianz

$$\mathrm{E}\left[(y-\hat{f})^2\right] = \mathrm{E}\left[y^2+\hat{f}^2-2y\hat{f}\right] = \mathrm{E}\left[y^2\right] + \mathrm{E}\left[\hat{f}^2\right] - \mathrm{E}[2y\hat{f}]$$

(1) (1) (2)

$$= \mathrm{Var}[y] + \mathrm{E}[y]^2 + \mathrm{Var}[\hat{f}] + \mathrm{E}[\hat{f}]^2 - 2f\mathrm{E}[\hat{f}] = \mathrm{Var}[y] + \mathrm{Var}[\hat{f}] + (f - \mathrm{E}[\hat{f}])^2$$

(3)

$$= \mathrm{Var}[y] + \mathrm{Var}[\hat{f}] + \mathrm{E}[\hat{f}-f]^2 = \sigma^2 + \mathrm{Var}[\hat{f}] + \mathrm{Bias}[\hat{f}]^2$$

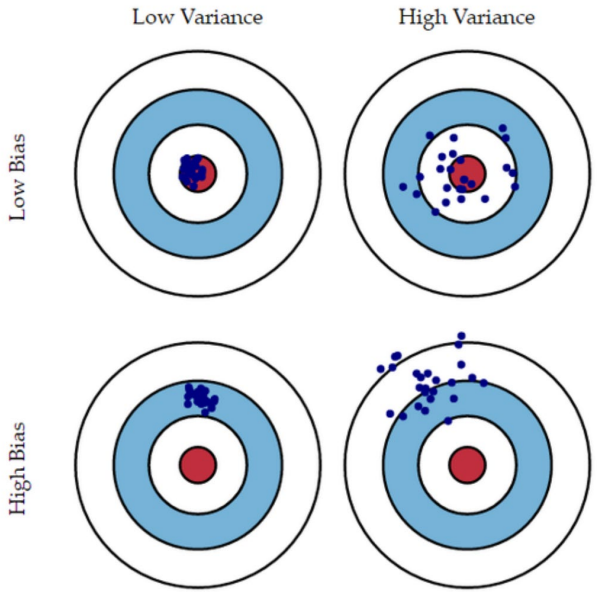
(1) $\mathrm{E}\left[X^2\right] = \mathrm{Var}[X] + \mathrm{E}[X]^2$ (Verschiebungssatz)

(2) $\mathrm{E}[y] = \mathrm{E}[f+\epsilon] = \mathrm{E}[f] = f$ ($\mathrm{E}[\epsilon] = 0$)

(3) $\mathrm{Var}[y] = \mathrm{E}\left[(y-\mathrm{E}[y])^2\right] = \mathrm{E}\left[(y-f)^2\right] = \mathrm{E}\left[(f+\epsilon-f)^2\right] = \mathrm{E}\left[\epsilon^2\right] = \mathrm{Var}[\epsilon] + \mathrm{E}[\epsilon]^2 = \sigma^2$

Verbindung zu Bias und Varianz

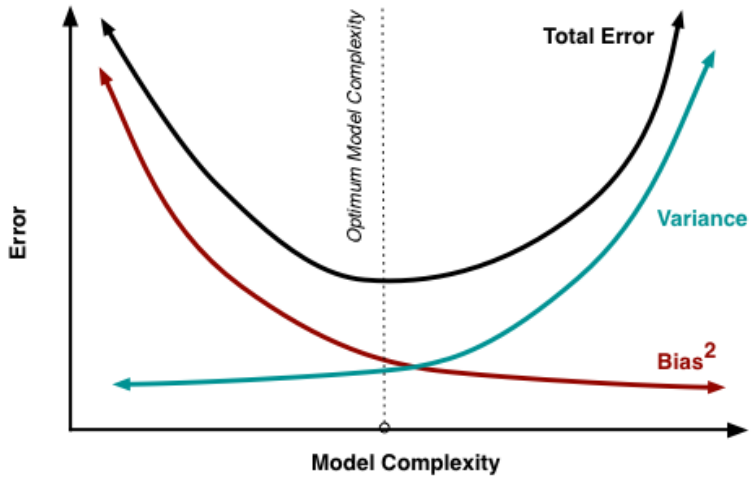
- Hoher Bias: Unteranpassung
- Hohe Varianz: Überanpassung



$$\mathrm{E}\left[(y - \hat{f}(x))^2\right] = \mathrm{Bias}[\hat{f}(x)]^2 + \mathrm{Var}[\hat{f}(x)] + \sigma^2$$

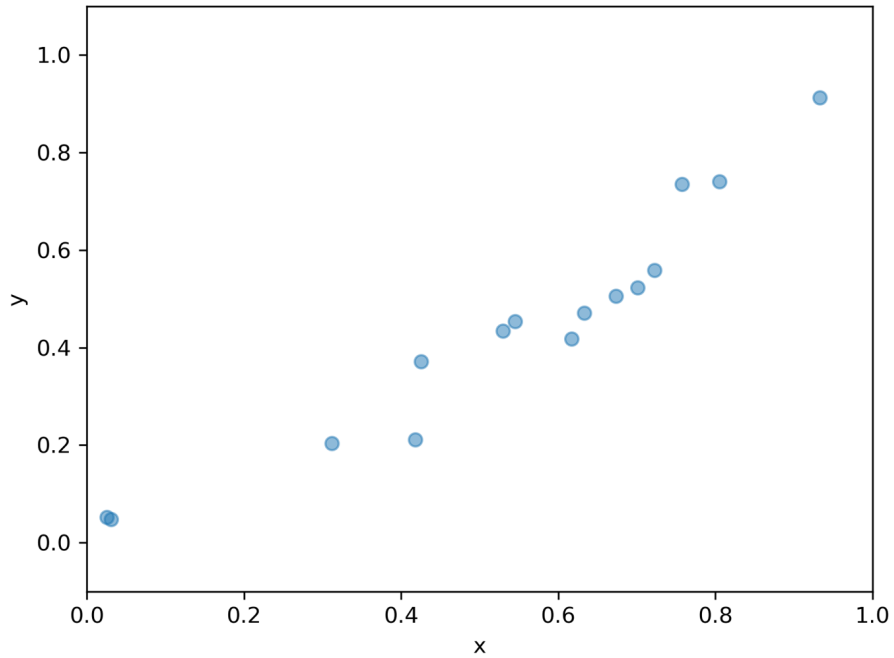
$$\mathrm{Bias}[\hat{f}(x)] = \mathrm{E}[\hat{f}(x) - f(x)]$$

$$\mathrm{Var}[\hat{f}(x)] = \mathrm{E}\left[(\hat{f}(x) - \mathrm{E}[\hat{f}(x)])^2\right]$$



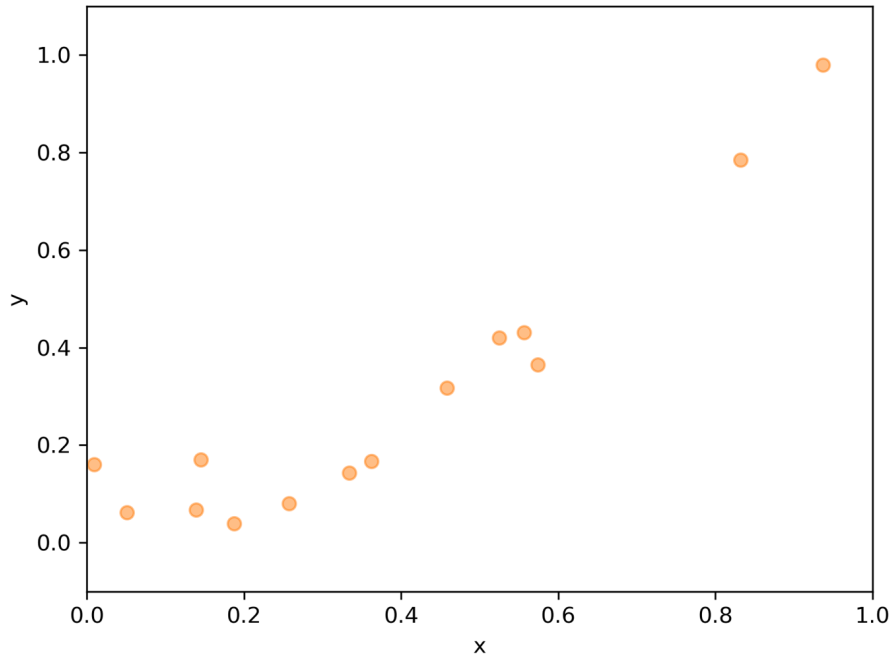
Einschub: Bias und Varianz - Beispiel

Nur ein Feature/Attribut: x ist 1D, $y = x^2 + \epsilon$



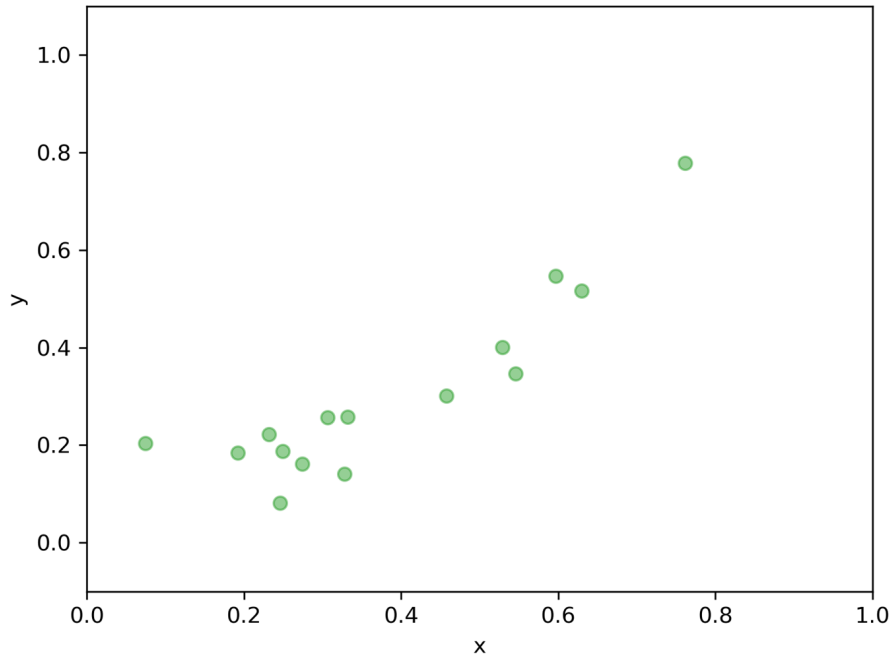
Einschub: Bias und Varianz - Beispiel

Nur ein Feature/Attribut: x ist 1D, $y = x^2 + \epsilon$



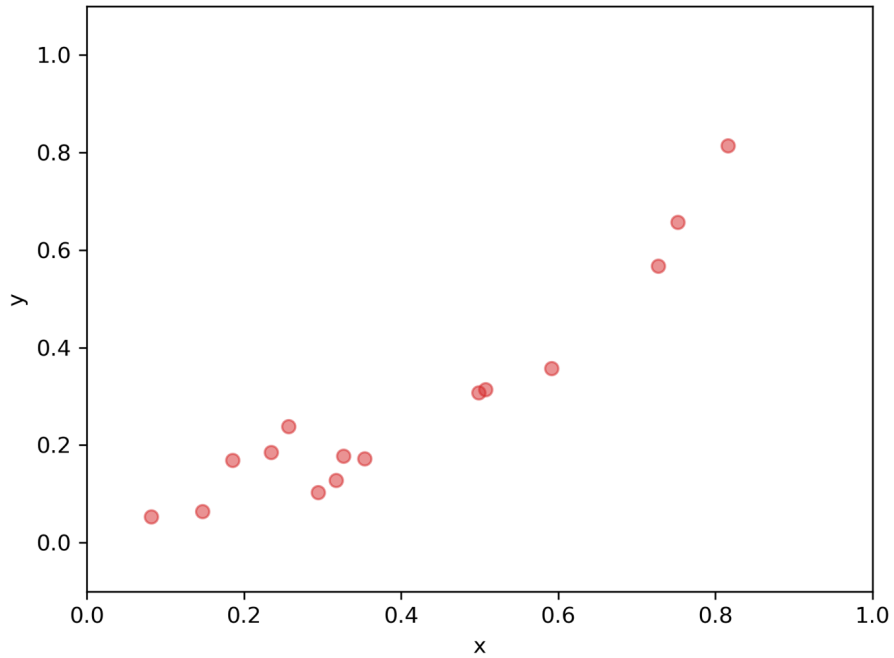
Einschub: Bias und Varianz - Beispiel

Nur ein Feature/Attribut: x ist 1D, $y = x^2 + \epsilon$



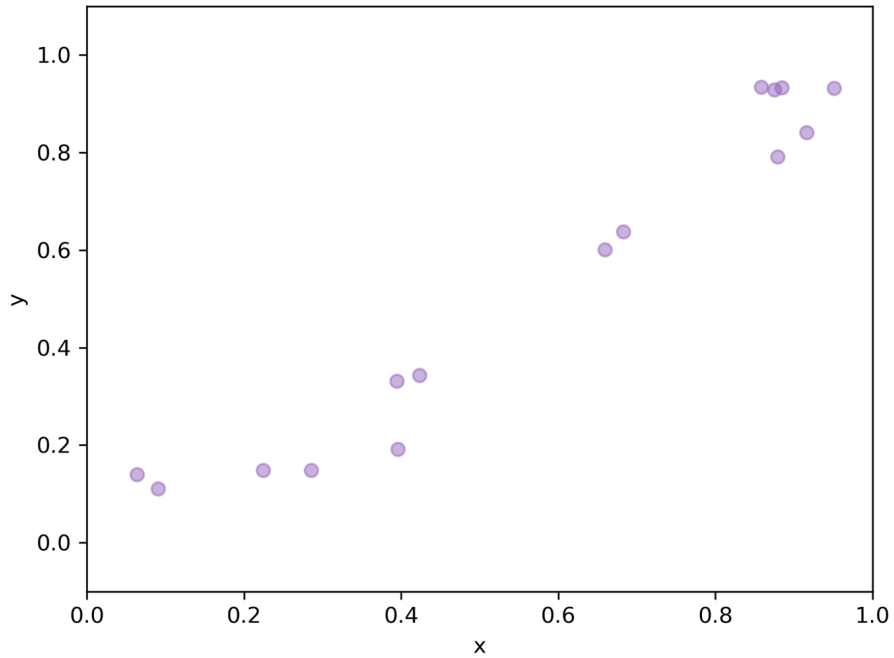
Einschub: Bias und Varianz - Beispiel

Nur ein Feature/Attribut: x ist 1D, $y = x^2 + \epsilon$



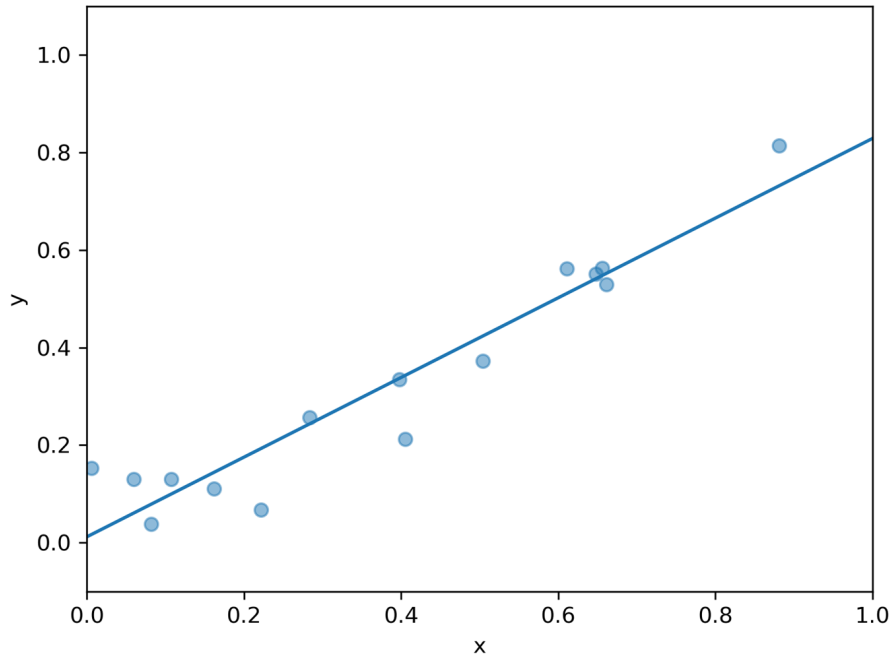
Einschub: Bias und Varianz - Beispiel

Nur ein Feature/Attribut: x ist 1D, $y = x^2 + \epsilon$



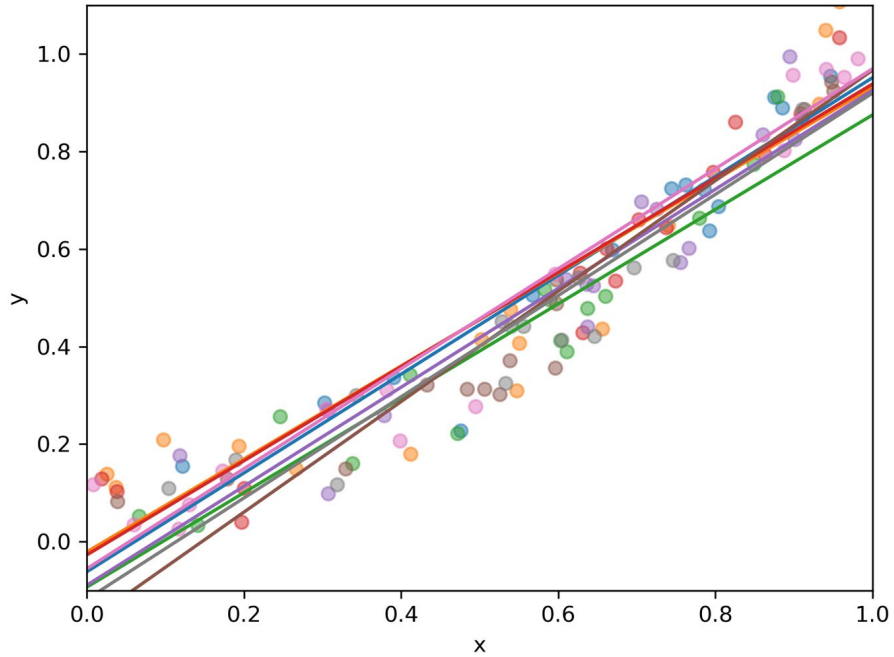
Einschub: Bias und Varianz - Beispiel

Modell: Linear, zwei Parameter



Einschub: Bias und Varianz - Beispiel

Modell: Linear, zwei Parameter

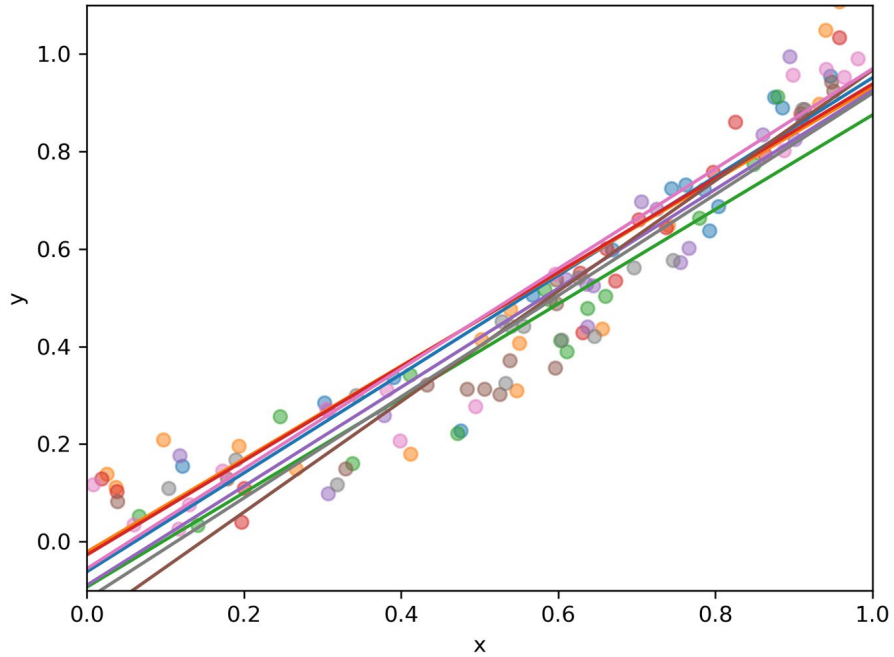


Wie hoch ist

(a) Varianz?
(b) Bias?

Einschub: Bias und Varianz - Beispiel

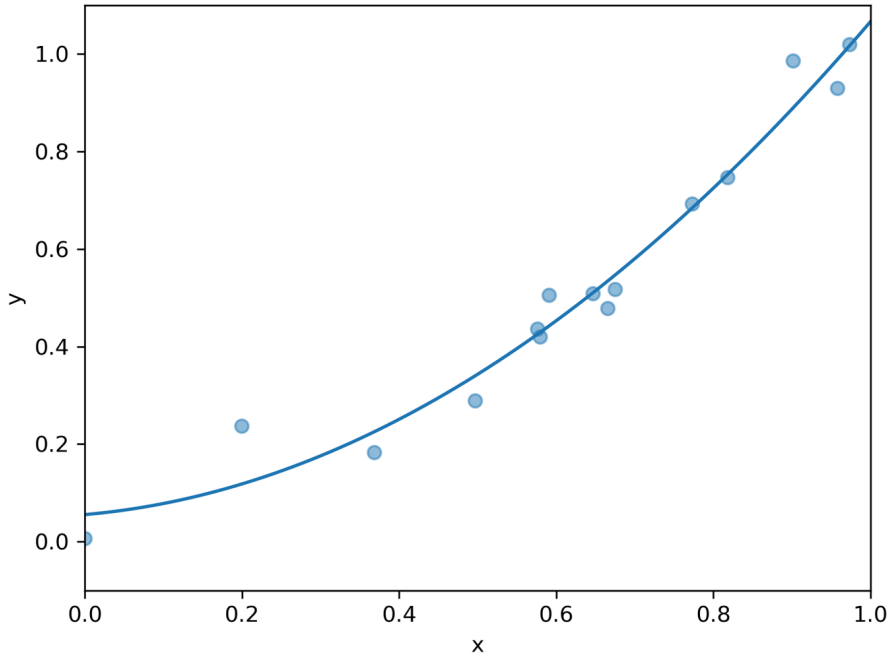
Modell: Linear, zwei Parameter



Niedrige Varianz
Hoher Bias

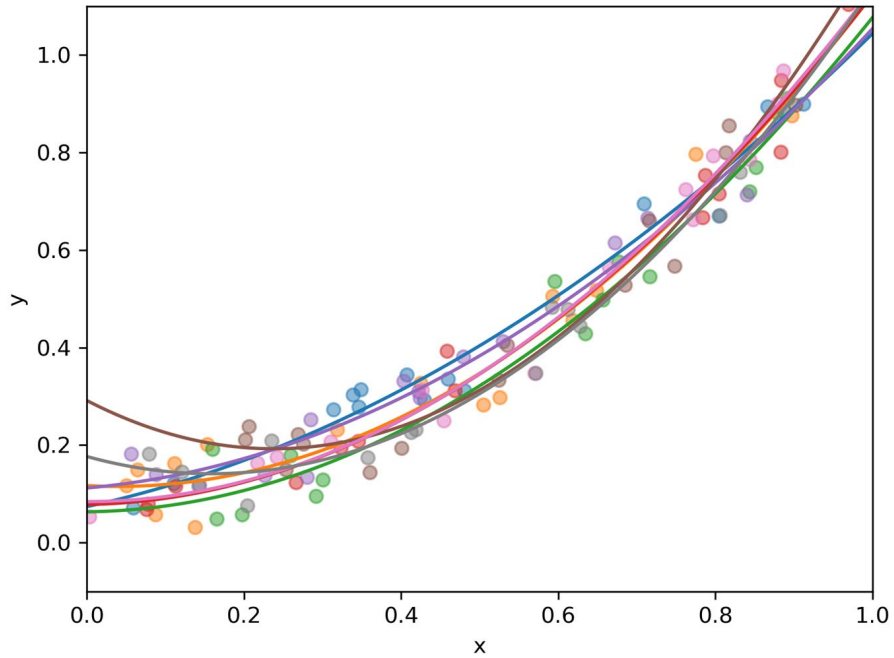
Einschub: Bias und Varianz - Beispiel

Modell: Quadratisch, drei Parameter



Einschub: Bias und Varianz - Beispiel

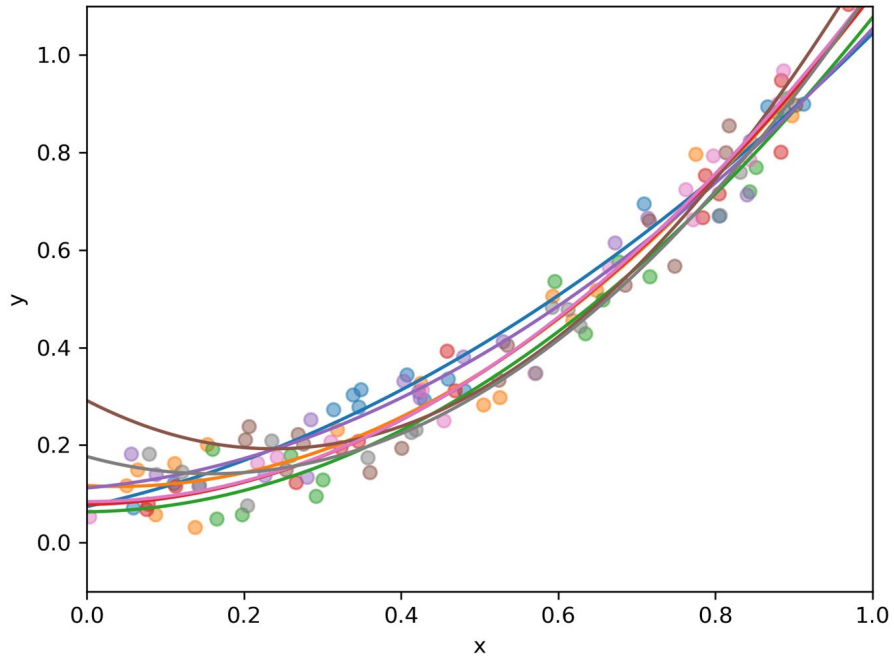
Modell: Quadratisch, drei Parameter



Varianz?
Bias?

Einschub: Bias und Varianz - Beispiel

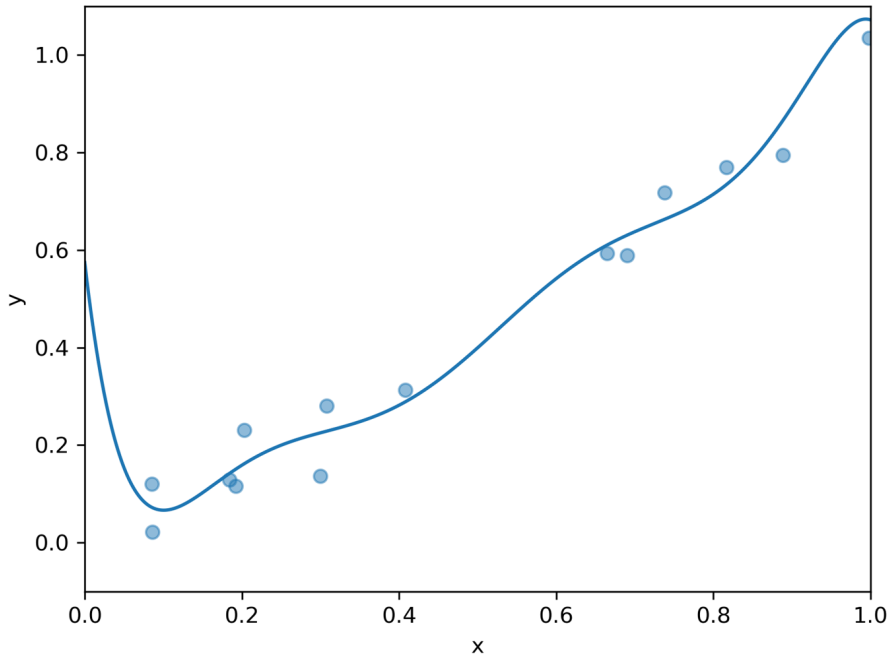
Modell: Quadratisch, drei Parameter



Niedrige Varianz
Niedriger Bias

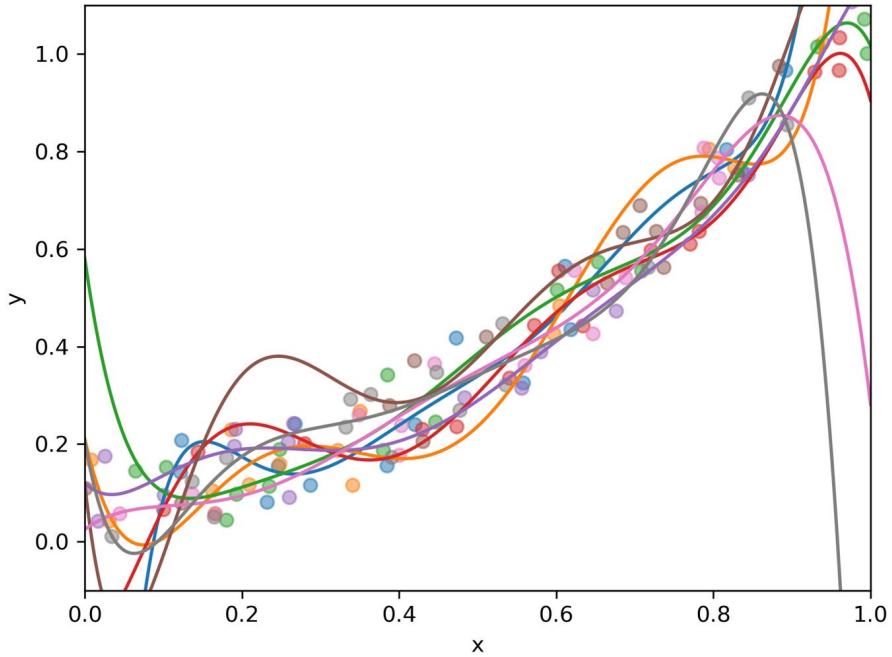
Einschub: Bias und Varianz - Beispiel

Modell: Polynom 7ten Grades, 8 Parameter



Einschub: Bias und Varianz - Beispiel

Modell: Polynom 7ten Grades, 8 Parameter

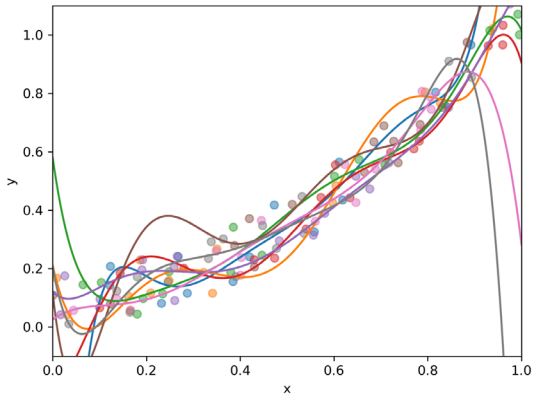


Varianz?
Bias?

You cannot vote anymore

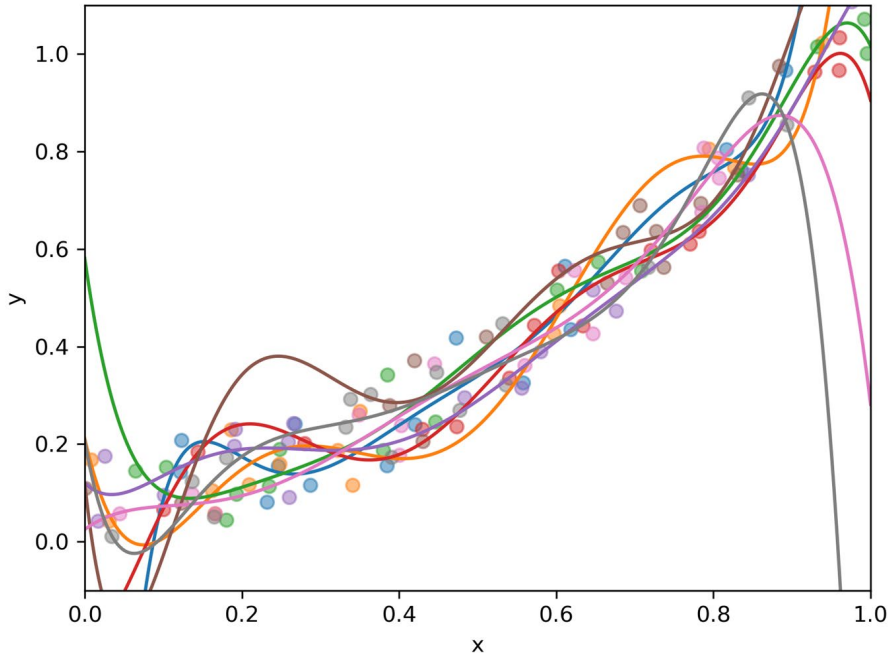
Für das Polynom 7ten Grades, haben wir...?

1. Hohe Varianz, niedriger Bias
2. Hohe Varianz, hohen Bias
3. Niedrige Varianz, niedriger Bias
4. Niedrige Varianz, hoher Bias



Einschub: Bias und Varianz - Beispiel

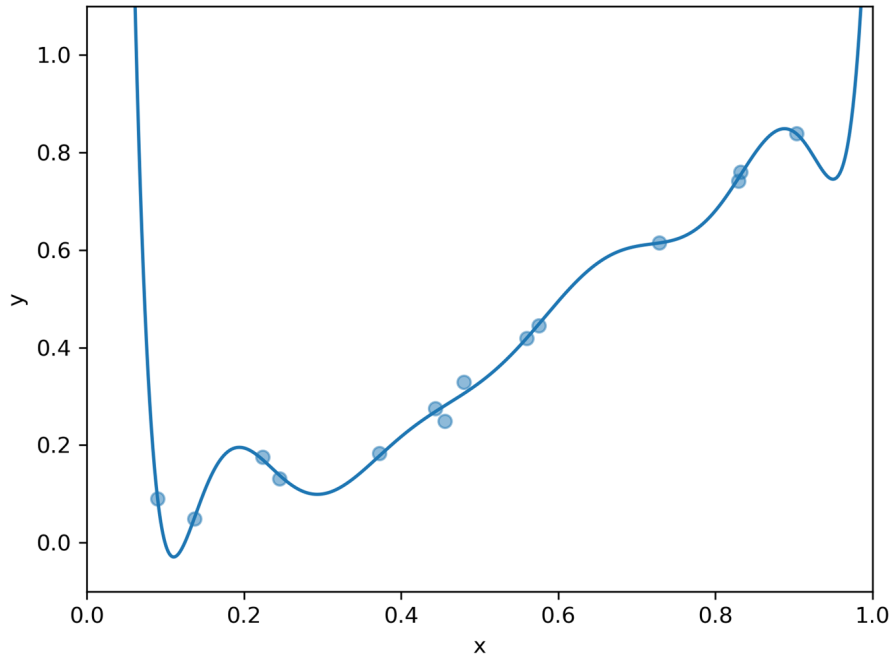
Modell: Polynom 7ten Grades, 8 Parameter



Hohe Varianz
Niedriger Bias

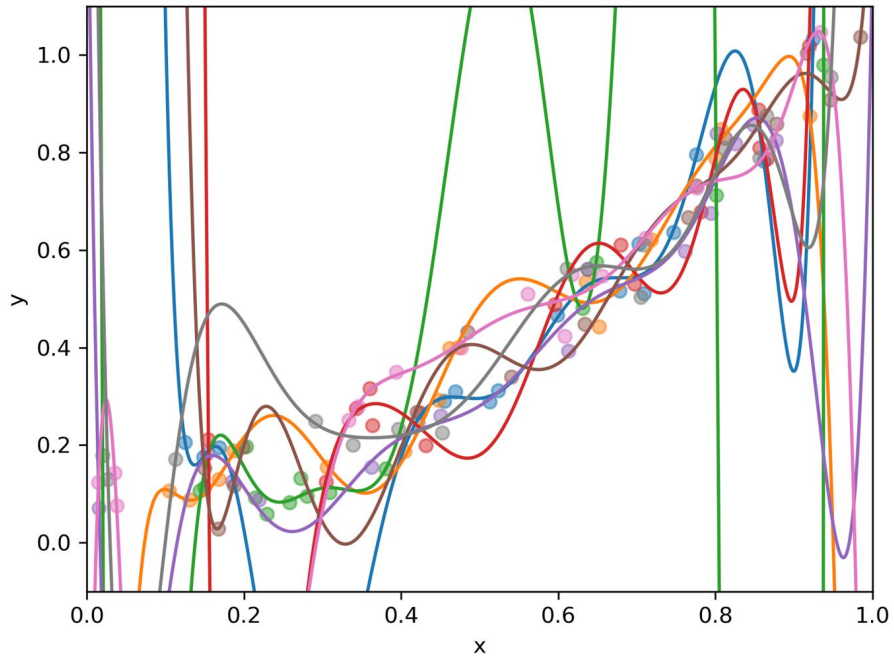
Einschub: Bias und Varianz - Beispiel

Modell: Polynom 10ten Grades, 11 Parameter



Einschub: Bias und Varianz - Beispiel

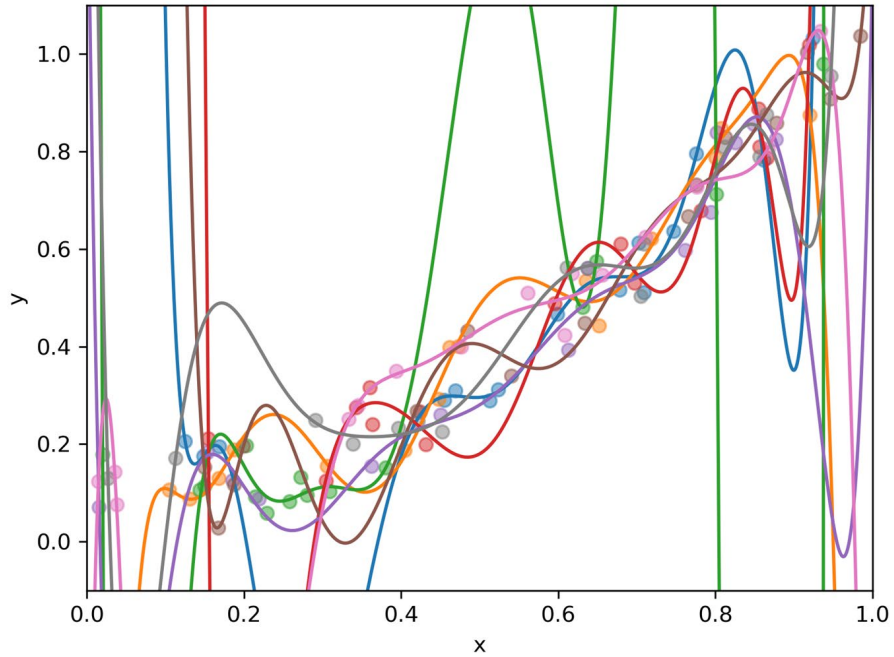
Modell: Polynom 10ten Grades, 11 Parameter



Varianz?
Bias?

Einschub: Bias und Varianz - Beispiel

Modell: Polynom 10ten Grades, 11 Parameter



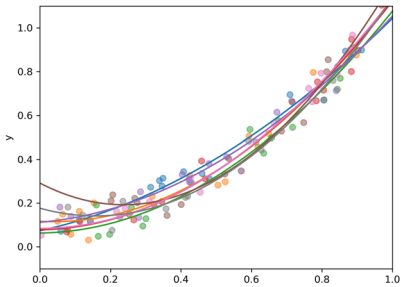
Riesige Varianz
Niedriger Bias

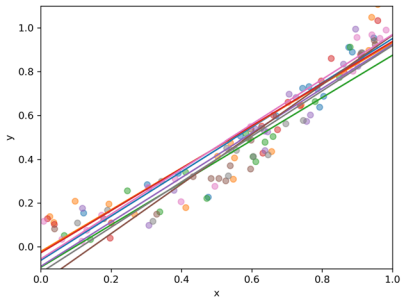
Verbindung zu Bias und Varianz

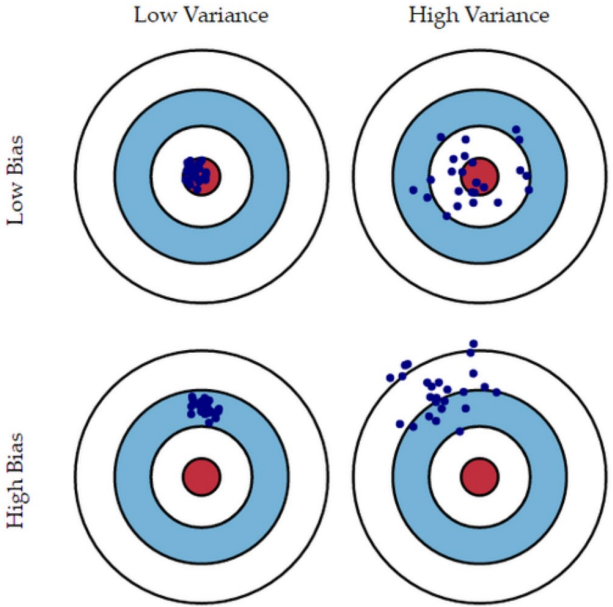
- Hoher Bias: Unteranpassung
- Hohe Varianz: Überanpassung

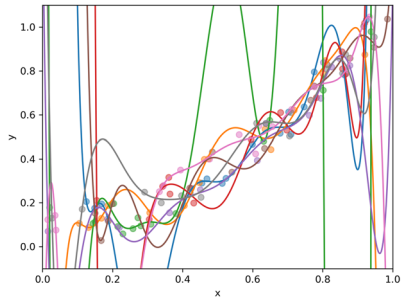
$$\text{Bias}[\hat{f}(x)] = \text{E}[\hat{f}(x) - f(x)]$$

$$\text{Var}[\hat{f}(x)] = \text{E}\left[(\hat{f}(x) - \text{E}[\hat{f}(x)])^2\right]$$





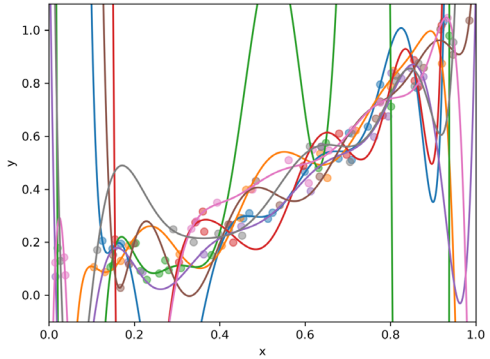




?

Fragen?

Teil 2



1) Generalisierung

2) Hyperparameter

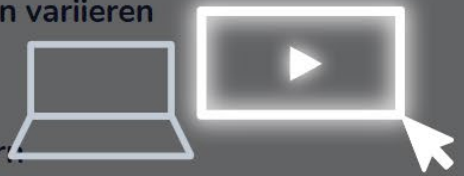
3) Regularisierung

4) Ensemble Methoden

You cannot vote anymore

Wie können wir aktiv Bias und Varianz in neuronalen Netzwerken beeinflussen? ...

1. Regularisierung der Kostenfunktion, z.B. wie in Ridge Regression
2. Anzahl Neuronen und Schichten variieren
3. Aktivierungsfunktion(en) ändern
4. Abbruchkriterium in Stochastic Gradient Descent ändern
5. Lernrate ändern



Click on the projected screen to start the question

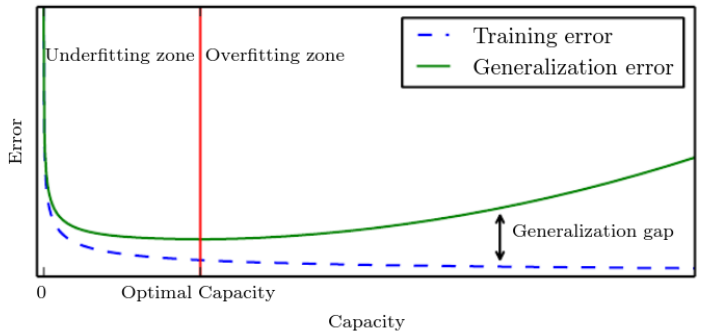
Erinnerung: Regularisierung zur Begrenzung der Kapazität

Regularisierung

- Bevorzugung weniger flexibler/einfacherer Funktionen
- Bestrafung großer Parameter im Modell („weight decay“)

$$L(\boldsymbol{w}) = \mathrm{MSE}_{\mathrm{train}} + \lambda \boldsymbol{w}^\top \boldsymbol{w}$$

- Dadurch kommt es zu einer Präferenz kleinerer Gewichte
- Was wiederrum zu weniger „steilen“ Funktionen und einfacheren Modellen führt



Parameter und Hyperparameter

Parameter

Automatisch optimiert: Gewichte w Bias b

Hyperparameter

Vorher festgelegt: Regularisierung λ Lernrate α

Wie können die Hyperparameter gewählt und optimiert werden?

Hyperparameter-Optimierung!

Fazit - Hyperparameter

Ziele (alle ungefähr äquivalent)

- Gute Generalisierung
- Kleine Lücke zwischen Trainings- und Testdaten
- Möglichst kleiner Fehler auf Testdaten

Ansätze

- Modellkapazität optimieren (Zahl und Größe der Layer anpassen)
- Regularisierung (Modellkapazität implizit beeinflussen)

Sonstige Design Entscheidungen

- Aktivierungsfunktionen, Lernrate, Optimierungsalgorithmus, …

→ Sehr viele Möglichkeiten, viele Entscheidungen!
→ Rechenaufwand limitiert, was wir versuchen können

Training, Validierung, Test

Problem

- Optimierung von Hyperparametern auf Trainingsdaten nicht möglich
- Warum? Größere Modellkapazität → Kleinerer Trainingsfehler → $\lambda = 0$

Lösung

- Zusätzlicher Datensatz zur **Validierung**
- Bisher: Daten werden in Trainings- und Testdaten eingeteilt
- Jetzt: **Trainingsdaten, Validierungsdaten und Testdaten**

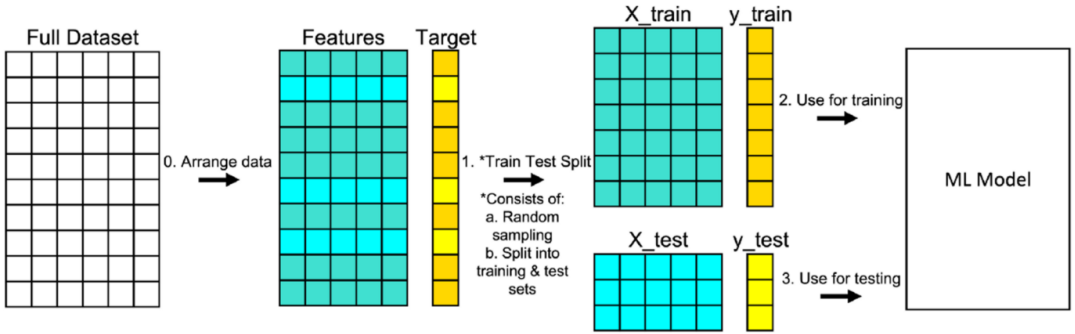
Vorgehen

- Bsp.: 80% der bisherigen Trainingsdaten werden zum Training und 20% zur Validierung benutzt → z.B. 60:20:20 Split (stark abhängig von Datensatz!)
- **Optimierung der Hyperparameter**: Minimierung des Validierungsfehlers
- Validierungsfehler unterschätzt möglicherweise den Generalisierungsfehler (auf Testdatensatz), da die Hyperparameter auf Validierungsdaten angepasst sind

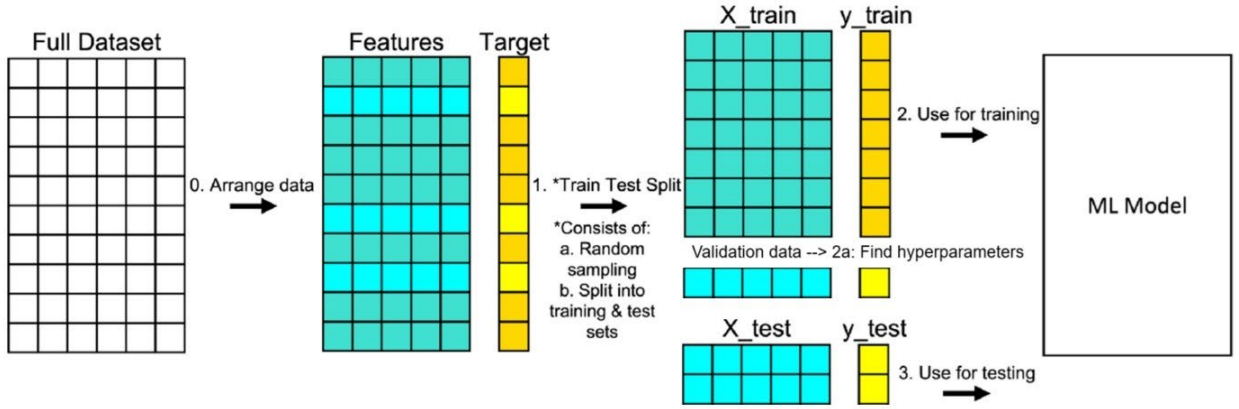
Test-Train-**Validation** split

Michael Galarnyk

Teile Daten auf in Training und Test Daten



Teile Daten auf in Training, Validierung und Test Daten

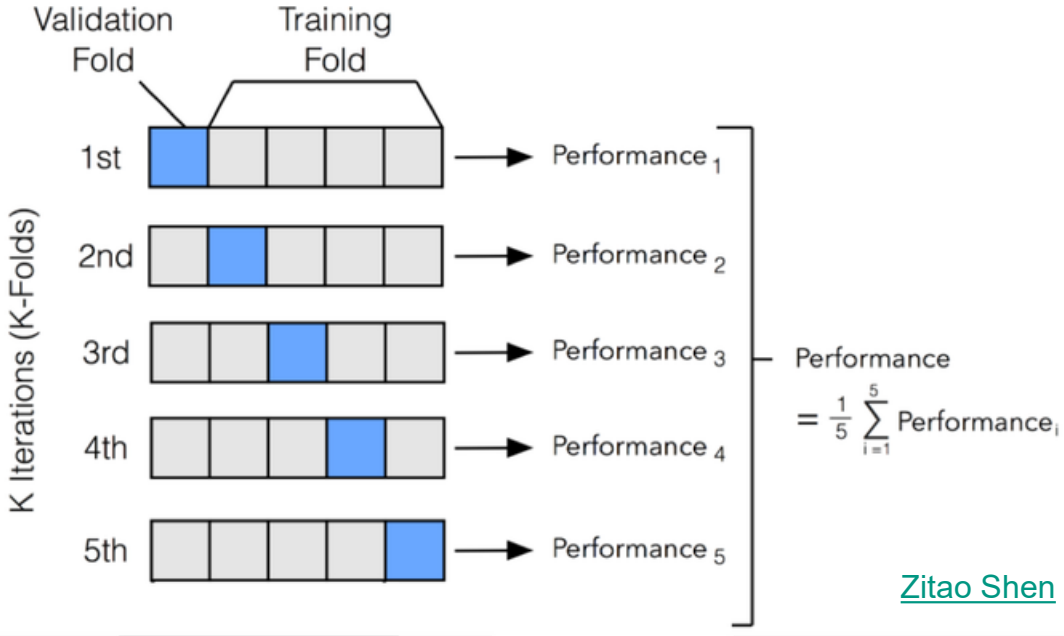


Kreuzvalidierung (Cross-Validation)

Gerade wenn wir wenig Daten haben, oder um eine besonders robuste Abschätzung der Hyperparameter zu treffen.

Lösung: Nutze unterschiedliche Teile des Datensatzes während des Trainings, z.B. je ein Fünftel der Daten für Validierung
→ „5-fold cross-validation“

Allgemein:
k-fold cross-validation



Zitao Shen

Hyperparameteroptimierung

Herausforderung

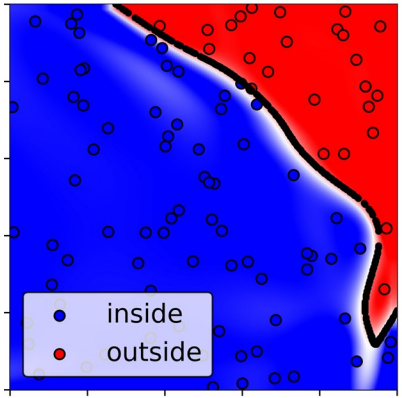
- Möglicherweise viele Hyperparameter
- Lernrate, Netztiefe, Zahl der Neuronen, Aktivierungsfunktion, Regularisierung, …

Mögliche Ansätze

- Manuelle Optimierung: Iterativ ausprobieren → Ggf. nicht systematisch
- Grid Search: Systematisches Testen aller Kombinationen → Teuer
- Random Search: Zufälliges Ausprobieren von Kombinationen
  → Effizienter (mehr Werte pro Hyperparameter), aber schwieriger zu interpretieren
- Bayes‘sche Optimierung: Iterativer Ansatz, in jedem Schritt wird der Informationsgehalt möglicher weiterer Experimente quantifiziert
- Evolutionäre Optimierung, …

Fragen?

Teil 3



1) Generalisierung

2) Hyperparameter

3) Regularisierung

4) Ensemble Methoden

Regularisierung

- Regularisierung zur Verbesserung des Generalisierungsfehlers
- Regularisierungsparameter sind eine wichtige Form von Hyperparameter, die wir z.B. durch Kreuzvalidierung optimieren können.
- Typischerweise: Verkleinerung des Modells
  → Verringerung der Anzahl an Modellparametern

Neuronale Netze und Deep Learning

- Häufig Herangehensweise:
  Nicht: Modell der “richtigen” Größe/Kapazität finden (Anzahl Neuronen, Schichten)
  Sondern: Großes Modell angemessen regularisieren

Regularisierung: Beispiele

Bestrafung der Parameternorm

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \lambda \Omega(\boldsymbol{\theta})$$

- Bei neuronalen Netzen werden oft nur die Gewichte regularisiert, nicht die Bias Parameter (es gibt sehr viel mehr Gewichtsparameter als Bias Parameter)
- Beispiele: L1- und L2-Regularisierung

L2-Regularisierung

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \lambda \Omega(\boldsymbol{\theta})$$

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2} \|\boldsymbol{w}\|_2^2$$

Alle Parameter

Gewichte

Damit für Gradient Descent:

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\lambda}{2} \boldsymbol{w}^\top \boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

$$\nabla_{\boldsymbol{w}} \tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \lambda \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

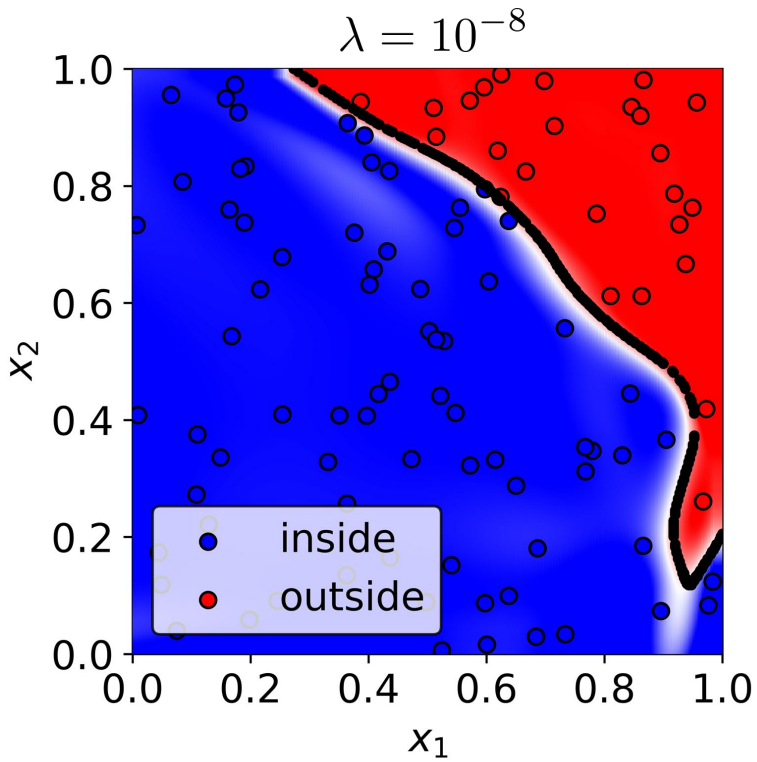
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \left(\lambda \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) \right)$$

„Weight decay“

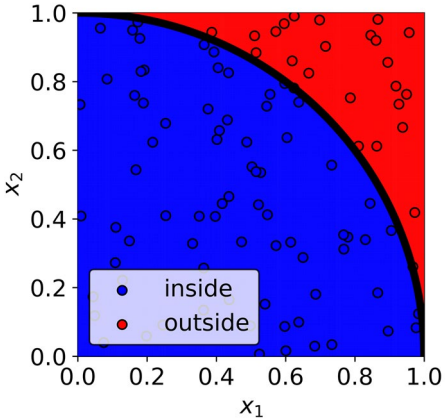
$$\boldsymbol{w} \leftarrow (1 - \alpha\lambda) \boldsymbol{w} - \alpha \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$$

L2-Regularisierung

Beispiel: Neuronales Netz (20 hidden Neuronen)

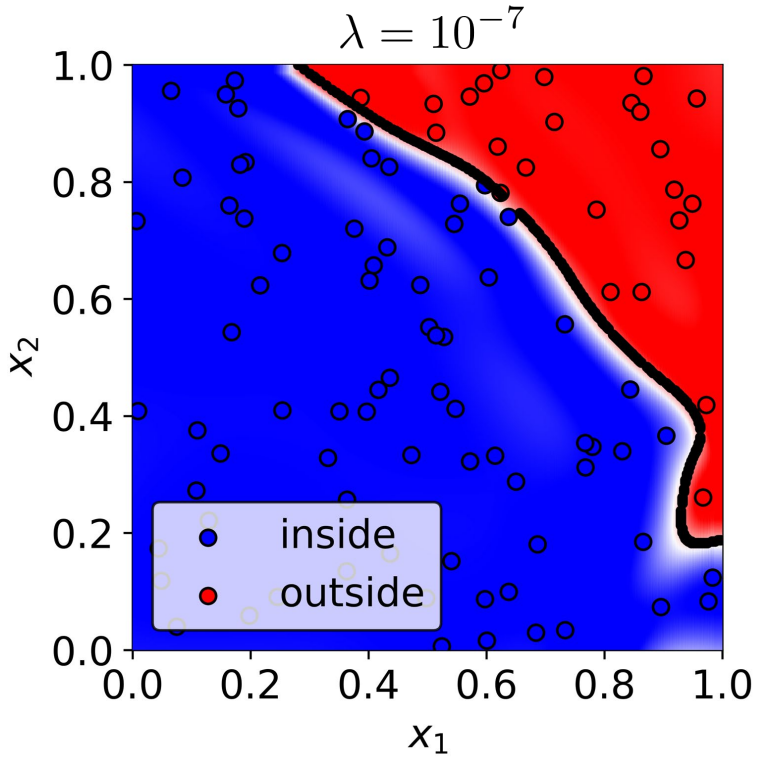


Ziel

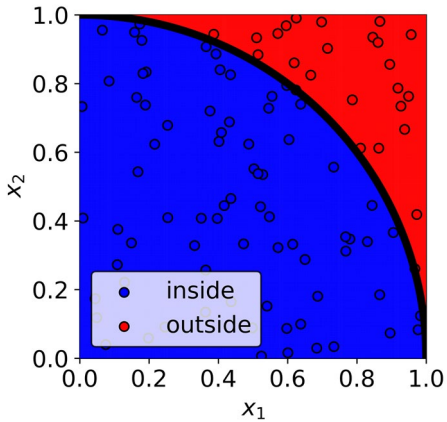


L2-Regularisierung

Beispiel: Neuronales Netz (20 hidden Neuronen)

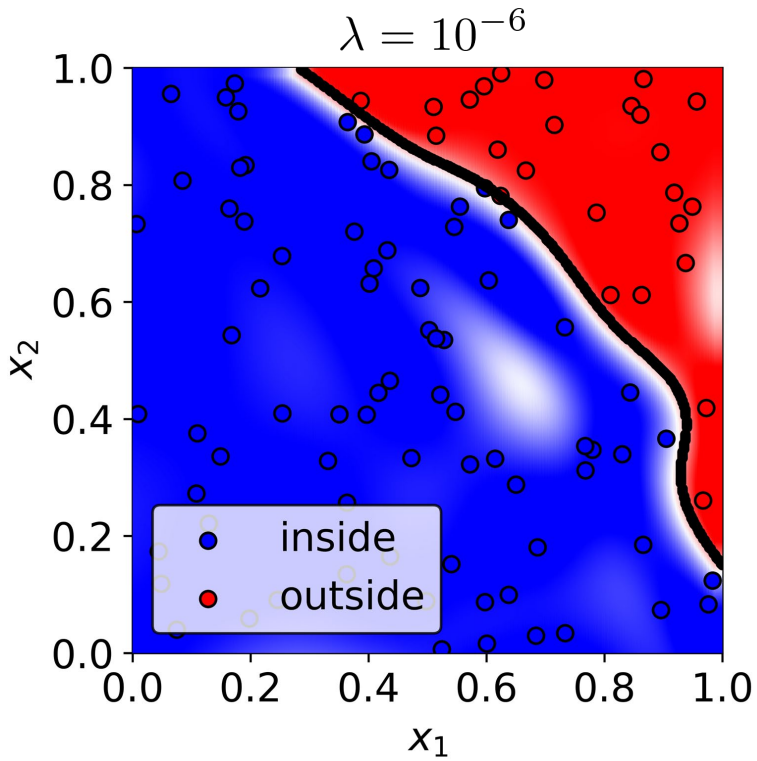


Ziel

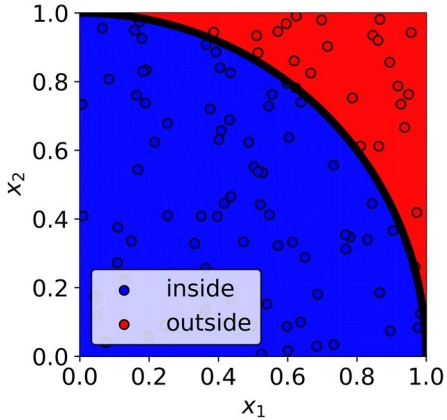


L2-Regularisierung

Beispiel: Neuronales Netz (20 hidden Neuronen)

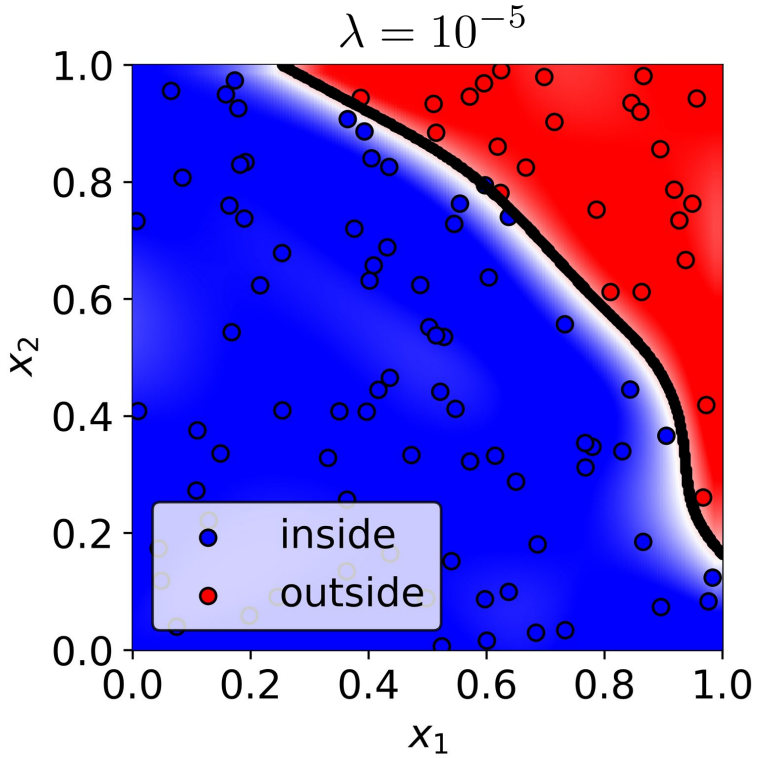


Ziel

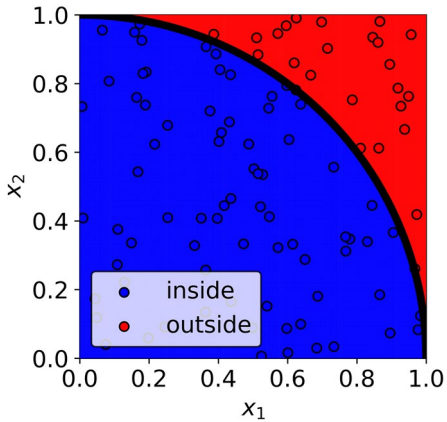


L2-Regularisierung

Beispiel: Neuronales Netz (20 hidden Neuronen)

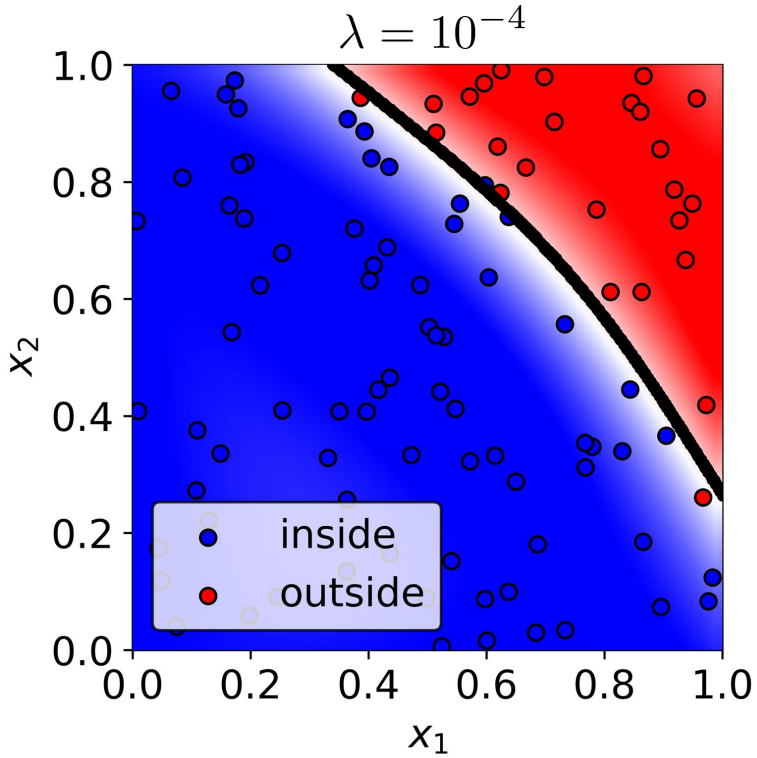


Ziel

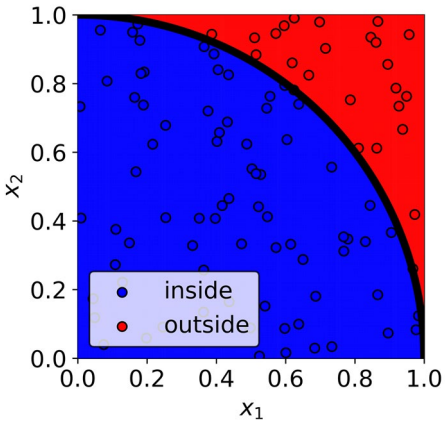


L2-Regularisierung

Beispiel: Neuronales Netz (20 hidden Neuronen)

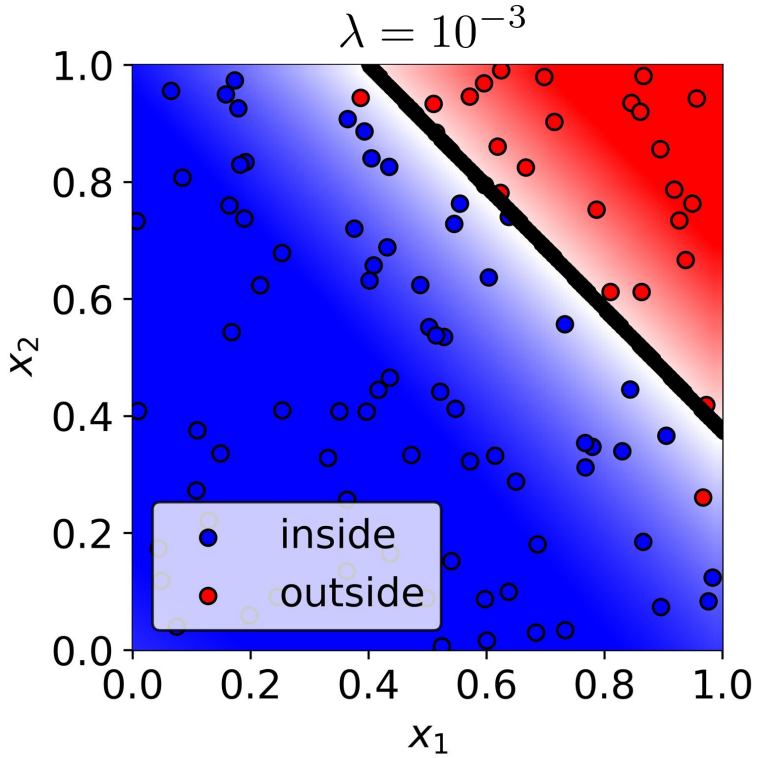


Ziel

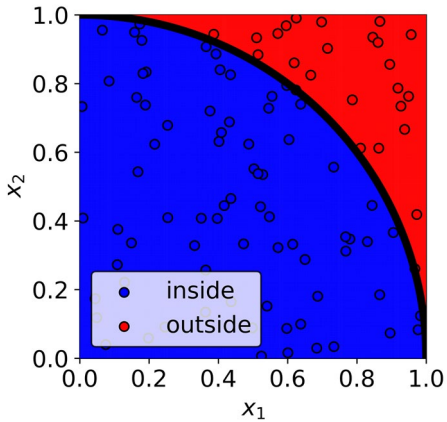


L2-Regularisierung

Beispiel: Neuronales Netz (20 hidden Neuronen)

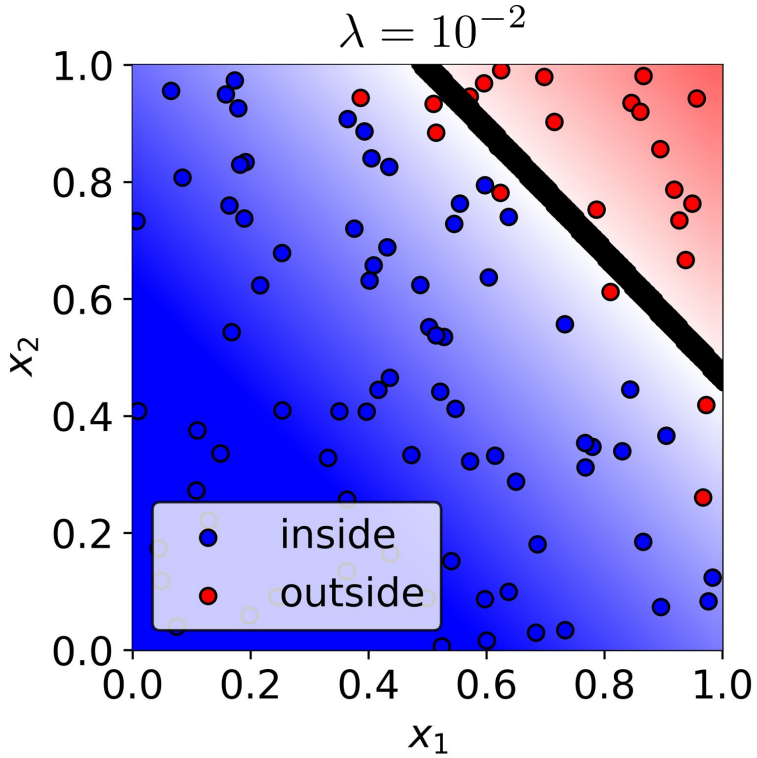


Ziel

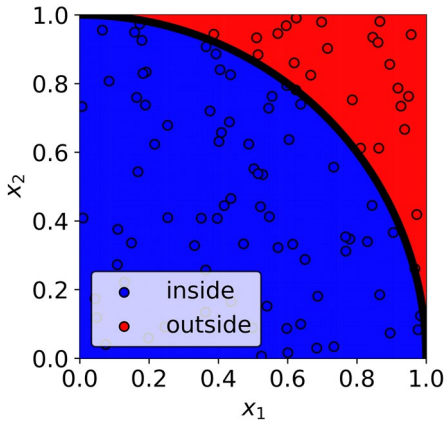


L2-Regularisierung

Beispiel: Neuronales Netz (20 hidden Neuronen)



Ziel



L1-Regularisierung

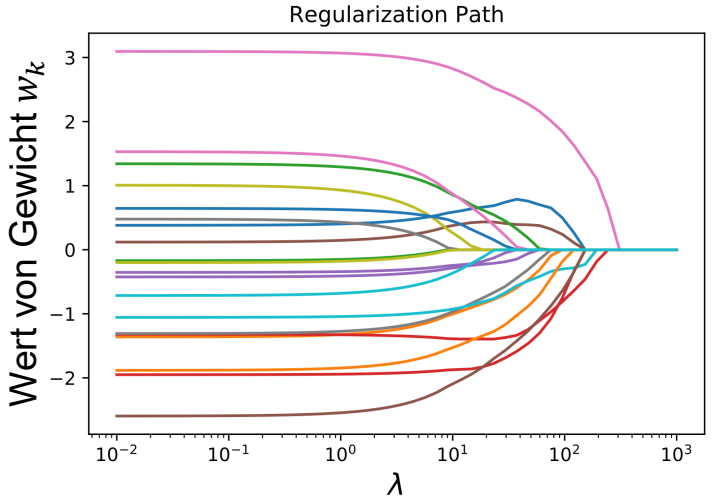
Bestrafung der **Absolutwerte** der Gewichte

$$\Omega(\boldsymbol{\theta}) = \|\boldsymbol{w}\|_1 = \sum_i |w_i|$$

- Effekt: Einige/viele Gewichte werden auf 0 gesetzt → Sparsification
- Daher oft genutzt zur Selektion von Attributen/Features → Interpretierbarkeit
- Lineares Beispiel: „LASSO“ Regression
- Bei L2-Regularisierung kann eine „sparse“ Lösung nur durch weiteres „thresholding“ erreicht werden

Regularisierung: LASSO

- Erinnerung: Lineare Regression: $w = argmin \sum_{i=1}^{N} \left(y_i - \sum_{k=0}^{K} x_{ik}\ w_k\right)^2$
- Wenn wir nun eine dünnbesetzte (sparse) Lösung für die Gewichte wollen, müssen wir die Zahl der Koeffizienten/Parameter begrenzen
  (hier lineare Koeffizienten, bei Neuronalen Netzen die Gewichte im Netz)
- z.B. L2-Norm: $\sqrt{\sum_{k=0}^{K} w_k^2} \leq \lambda$
- oder L1-Norm: $\sum_{k=0}^{K} |w_k| \leq \lambda$
- Bekannt als LASSO-Regularisierung
  (Least Absolute Shrinkage and Selection Operator)



https://www.cvxpy.org/examples/machine_learning/lasso_regression.html

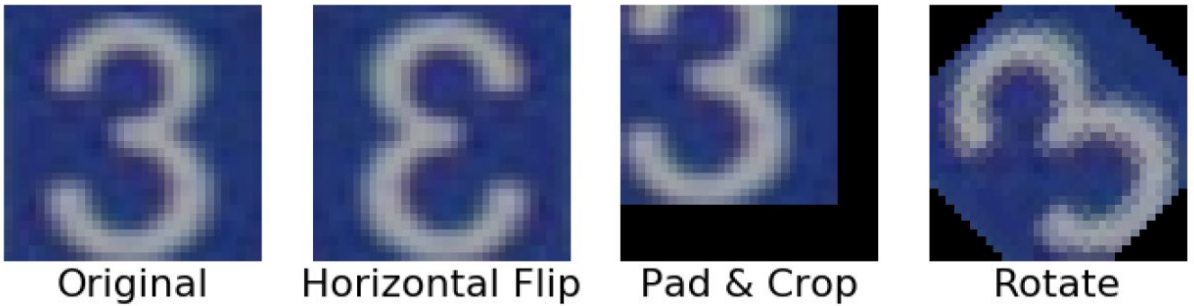
Regularisierung durch Erweiterung des Trainingssets

„Data augmentation“

- Hilfreich, wenn bei einem Klassifizierungsalgorithmus Invarianzen bekannt sind

Beispiel:

- **Drehung/Verzerrung** von Bilddaten **ohne Änderung des Labels**
- „Ein gestauchter/gespiegelter/verschobener/gedrehter Hund bleibt ein Hund.“
- Gut für Bilddaten, leider nicht einfach übertragbar auf andere ML-Tasks

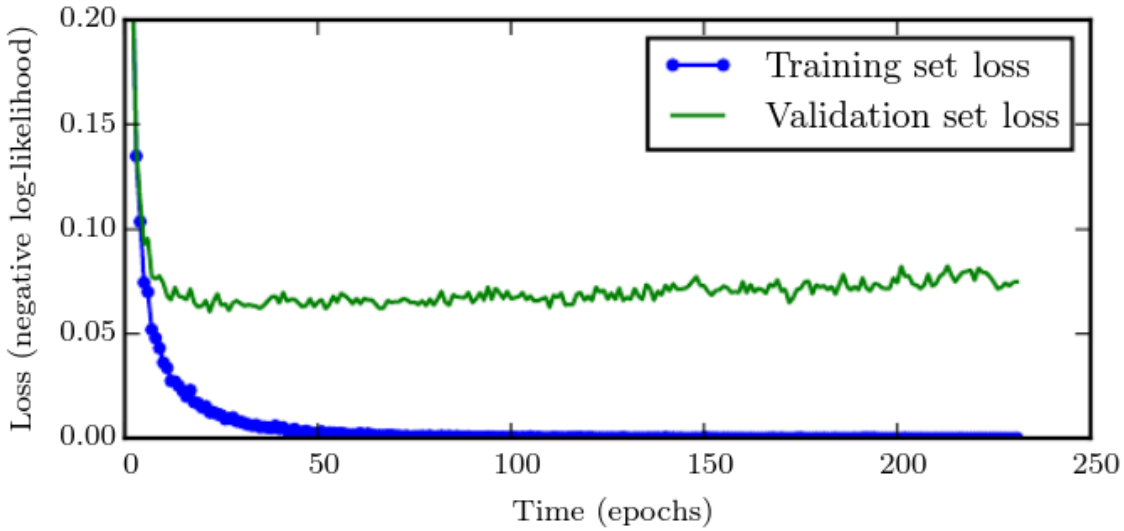


https://bair.berkeley.edu/blog/2019/06/07/data_aug/

Early Stopping – Wann höre ich auf zu trainieren?

Bei zu hoher Modellkapazität

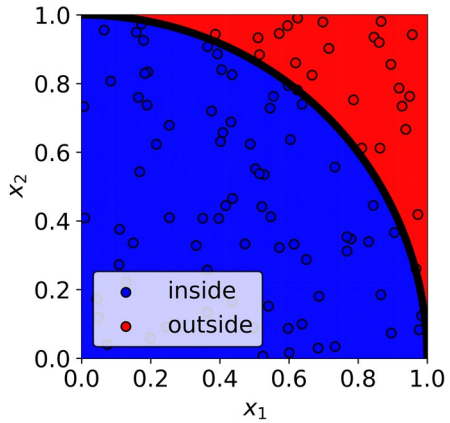
- Trainingsfehler sinkt, Validierungsfehler steigt
- Stopp des Trainings wenn über eine bestimmte Zeit („patience“) keine Verbesserung des Validierungsfehlers beobachtet wurde; hilfreich um Trainingszeit einzuschränken.



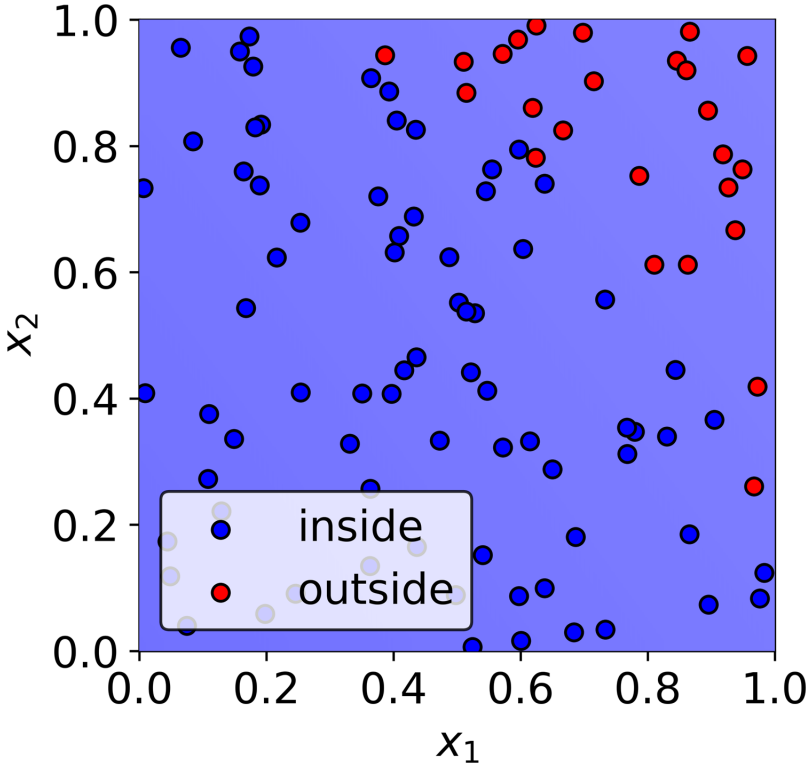
Early stopping: Beispiel

Neuronales Netz

Ziel



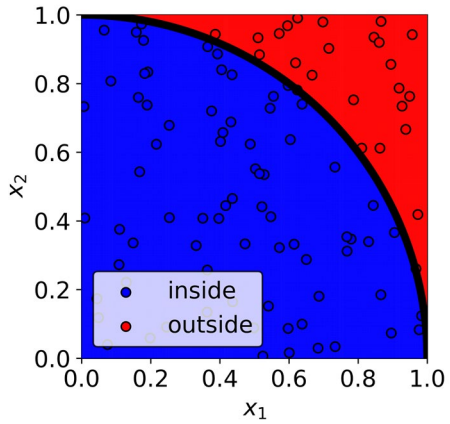
Tatsächliches Ergebnis: Epoche 1



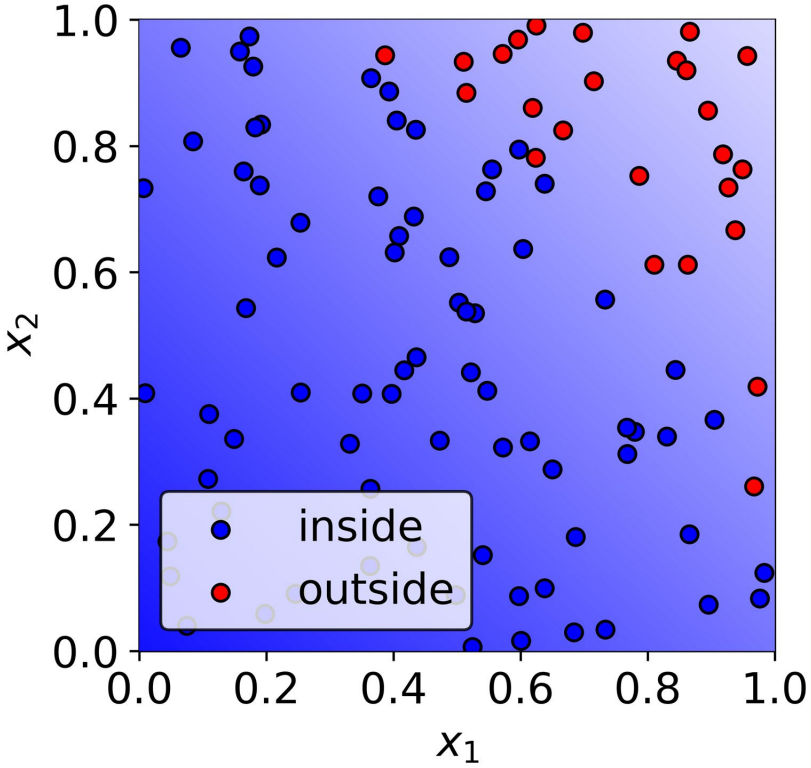
Early stopping: Beispiel

Neuronales Netz

Ziel



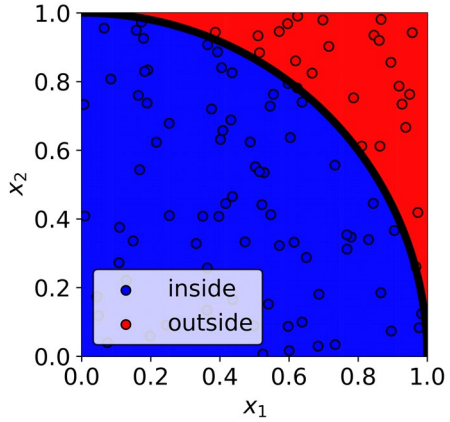
Tatsächliches Ergebnis: Epoche 46



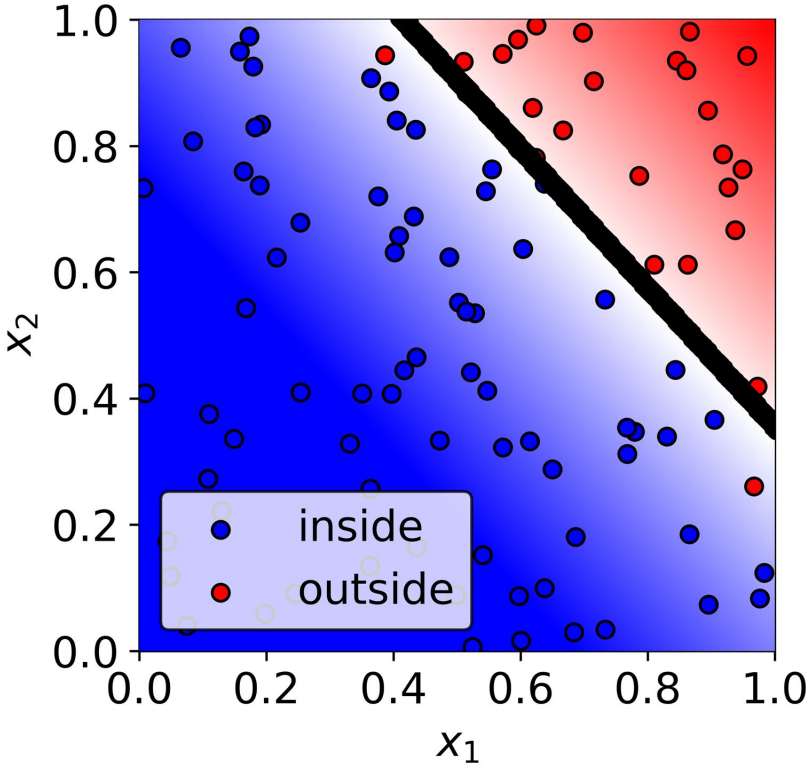
Early stopping: Beispiel

Neuronales Netz

Ziel



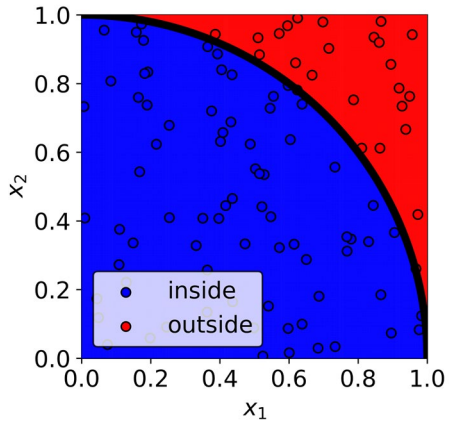
Tatsächliches Ergebnis: Epoche 166



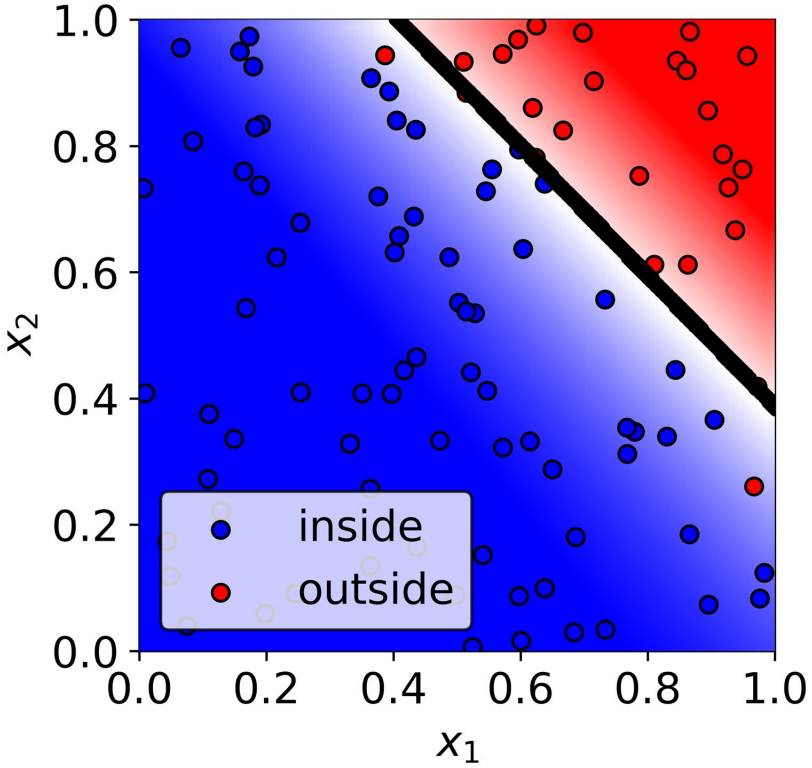
Early stopping: Beispiel

Neuronales Netz

Ziel



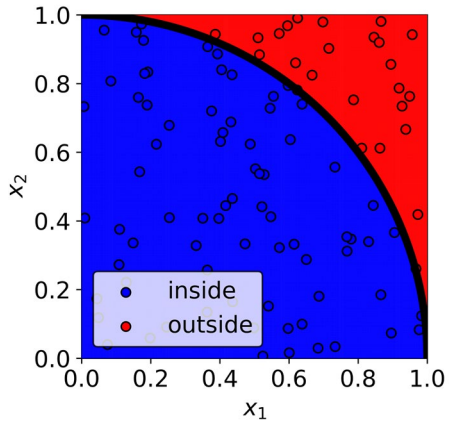
Tatsächliches Ergebnis: Epoche 599



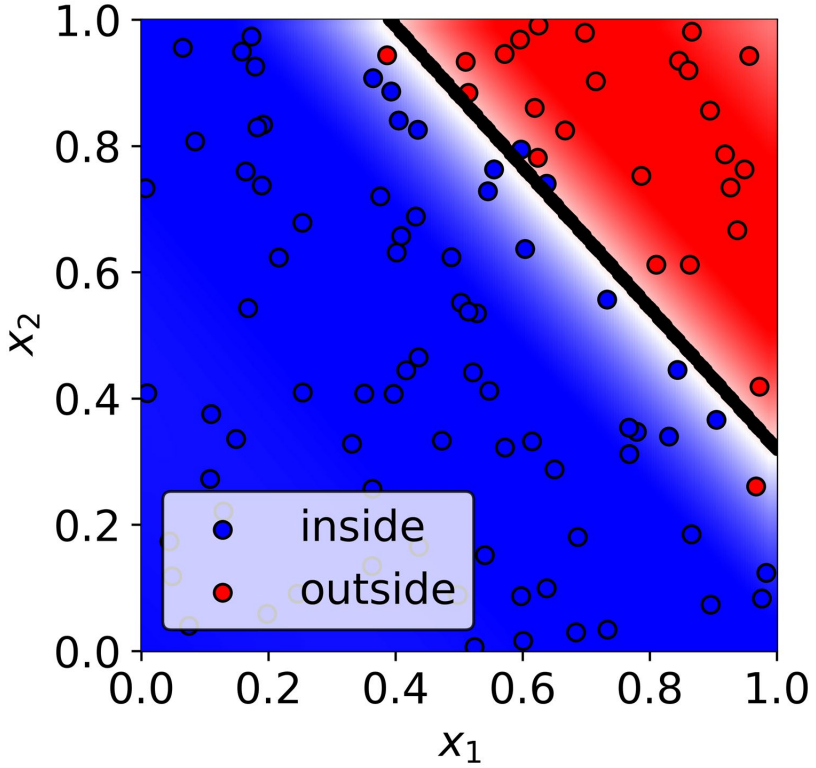
Early stopping: Beispiel

Neuronales Netz

Ziel



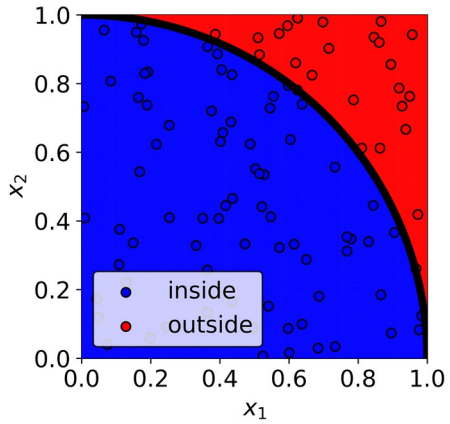
Tatsächliches Ergebnis: Epoche 2154



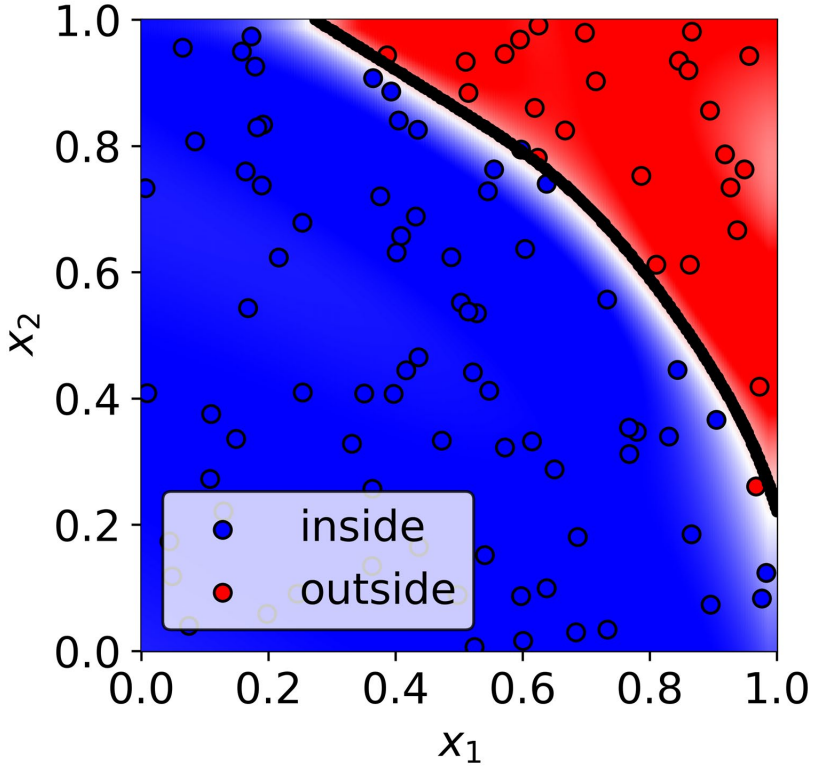
Early stopping: Beispiel

Neuronales Netz

Ziel



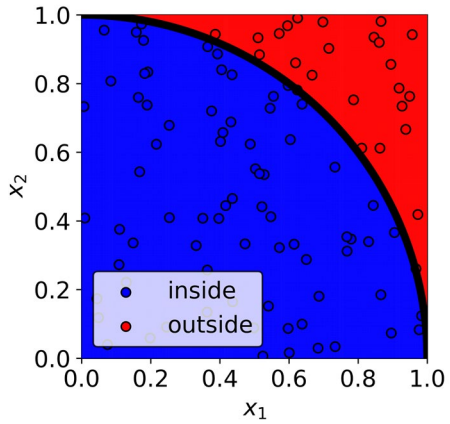
Tatsächliches Ergebnis: Epoche 7742



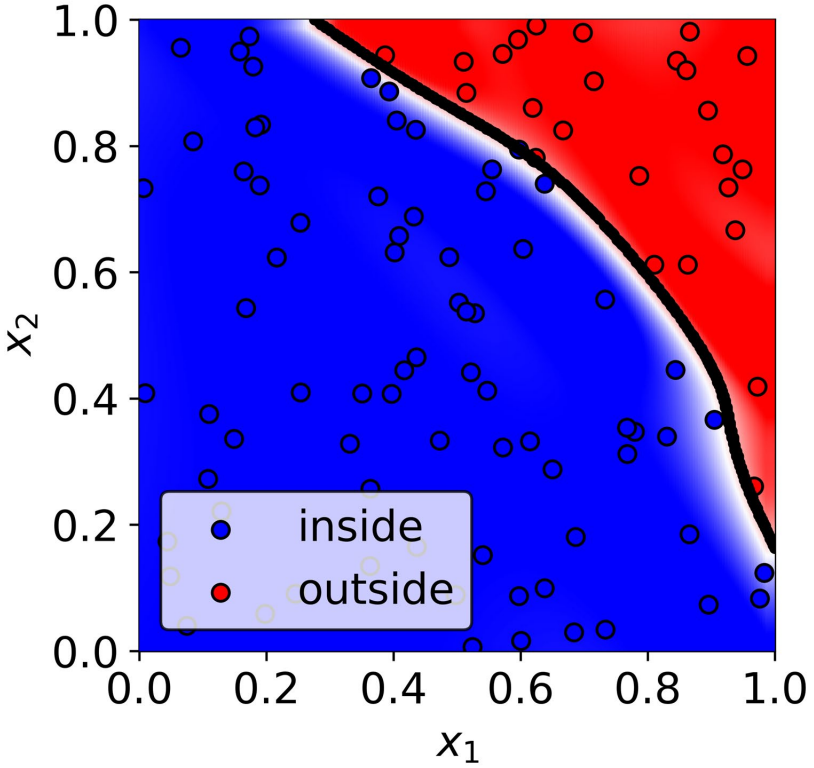
Early stopping: Beispiel

Neuronales Netz

Ziel



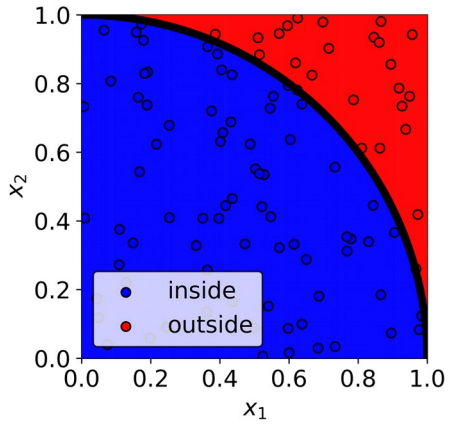
Tatsächliches Ergebnis: Epoche 27825



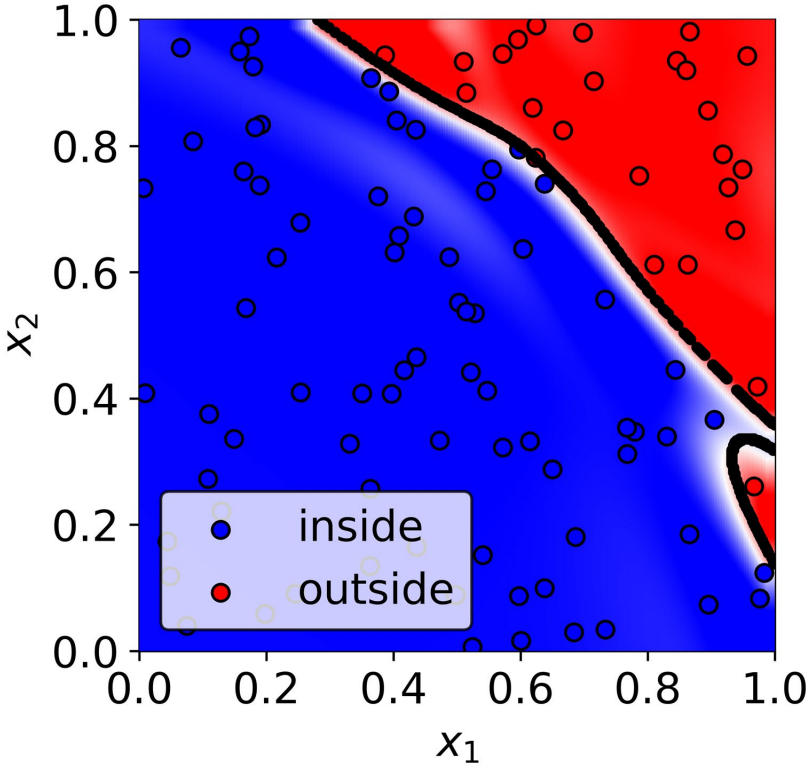
Early stopping: Beispiel

Neuronales Netz

Ziel



Tatsächliches Ergebnis: Epoche 100000



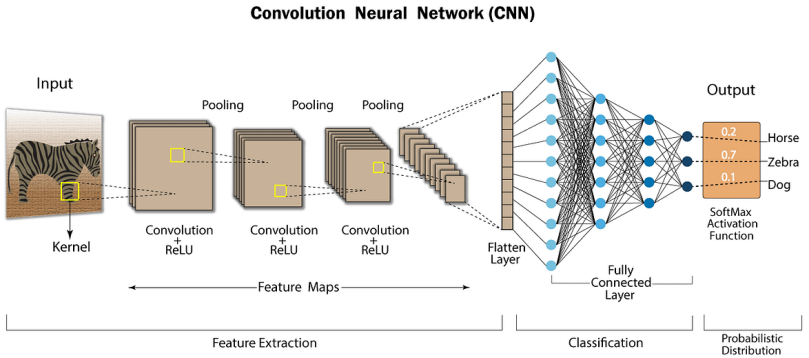
Parameter Sharing

- Das **Gleichsetzen von Parametern** lässt sich als eine Form der Regularisierung interpretieren
- Prominentestes Beispiel: **Convolutional Neural Networks (CNNs)**

→ Weniger Parameter insgesamt

→ Ausnutzung von wenigen Transformationen auf vielen Attributen

→ Mehr dazu in CNN-Vorlesung 9.



A. Singh

Teil 4

$$\mathrm{Var}[\hat{f}_{\mathrm{ens}}] = \frac{1}{m}(1-\rho)\sigma^2 + \rho\sigma^2$$

1) Generalisierung

2) Hyperparameter

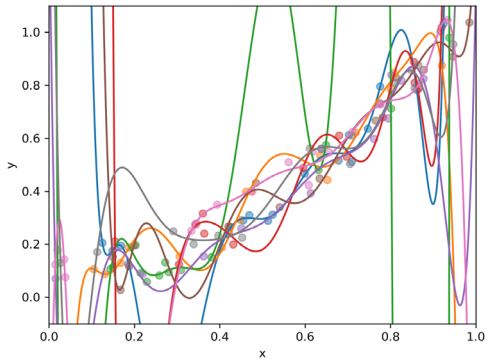
3) Regularisierung

4) Ensemble Methoden

Ensemble Methoden

Idee

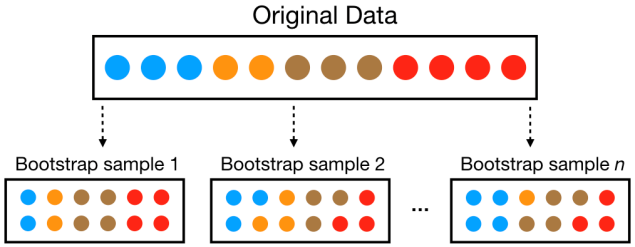
- Trainiere mehr als einen Regressor/Klassifikator
- Mittel über alle Vorhersagen der Einzelmodelle
- Erhalte leicht **bessere Ergebnisse als jedes einzelne Modell**
- Warum/wann funktioniert das?
- Unabhängigkeit der Modelle → Mittelung über verschiedene (falsche) Vorhersagen



Bagging und Boosting

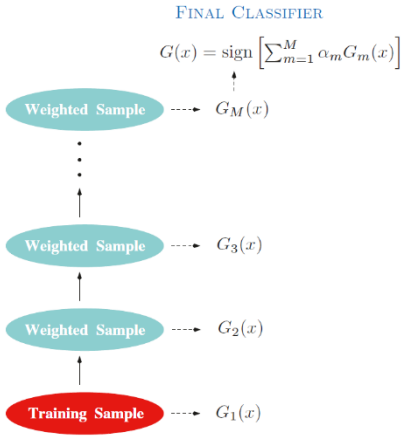
Wie wird Unabhängigkeit erreicht?

- **Bagging** (Bootstrap aggregation)
  - Paralleles Training von unabhängigen Modellen
  - Training auf zufälligen Subsets der Daten (gezogen mit Zurücklegen)



- **Boosting**
  - Sequentielles Training von Modellen
  - Fokus auf bisher falsch vorhergesagte Trainingspunkte

(später anhand von Entscheidungsbäumen gezeigt)



Kurze Erinnerung an Nomenklatur

Gegeben: Inputs $x_1, \ldots, x_n$

Labels $y_1, \ldots, y_n$ mit $y_i = f(x_i) + \epsilon$ $(\epsilon \sim \mathcal{N}(0, \sigma^2))$

Gesucht: Funktion $\hat{f}(x)$, welche die Verlustfunktion $L = (y - \hat{f}(x))^2$ minimiert.

Ensemble: Bias und Varianz

Einzelnes Modell

$$\mathbb{E}\left[\hat{f}_i\right] = \mu$$

$$\mathrm{Var}\left[\hat{f}_i\right] = \sigma^2$$

Gemitteltes Modell

$$\hat{f}_{\mathrm{ens}} = \frac{1}{m}\sum_{i=1}^{m}\hat{f}_i$$

$$\mathbb{E}[\hat{f}_{\mathrm{ens}}] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}\hat{f}_i\right] = \mu$$

Für ein immer größeres Ensemble m wird die Varianz immer kleiner!

$$\mathrm{Var}[\hat{f}_{\mathrm{ens}}] = \mathrm{Var}\left[\frac{1}{m}\sum_{i=1}^{m}\hat{f}_i\right] \overset{1)}{\underset{2)}{=}} \frac{1}{m^2}\sum_{i=1}^{m}\mathrm{Var}\left[\hat{f}_i\right] = \frac{1}{m}\sigma^2$$

(unabhängige Modelle)

$$\mathrm{Var}[\hat{f}_{\mathrm{ens}}] = \frac{1}{m}(1-\rho)\sigma^2 + \rho\sigma^2$$

(Korrelation ρ)

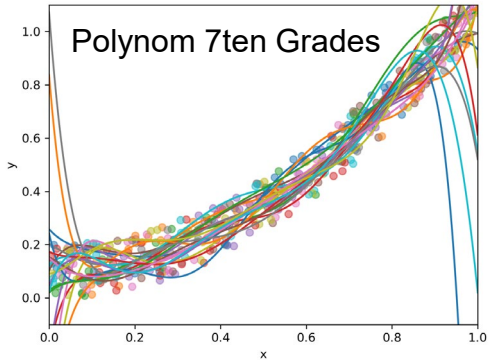
1) $\mathrm{Var}(aX) = \mathbb{E}\left((aX - a\mu)^2\right) = \mathbb{E}\left(a^2(X-\mu)^2\right) = a^2\,\mathrm{Var}(X)$

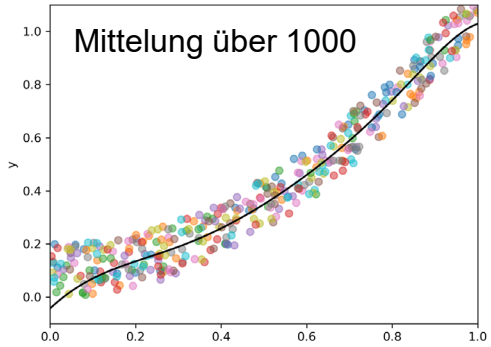
2) $\mathrm{Var}\left[\sum_i X_i\right] = \sum_i \mathrm{Var}\left[X_i\right]$ (nur wenn X_i unabhängig)

Ensemble: Bias und Varianz

Ergebnis

- Trainiere m unabhängige Modelle
- Für **unabhängige Modelle** ($\rho = 0$): **Reduktion der Varianz** auf 1/m
- **Bias** wird bei der Mittelung **nicht verändert**
- Falls die Modelle unabhängig sind (also unkorrelierte Fehler machen), wird das Gesamtmodell besser als jedes Einzelmodell
- Falls nicht, ist das Ensemble mindestens genauso gut wie jedes Einzelmodell

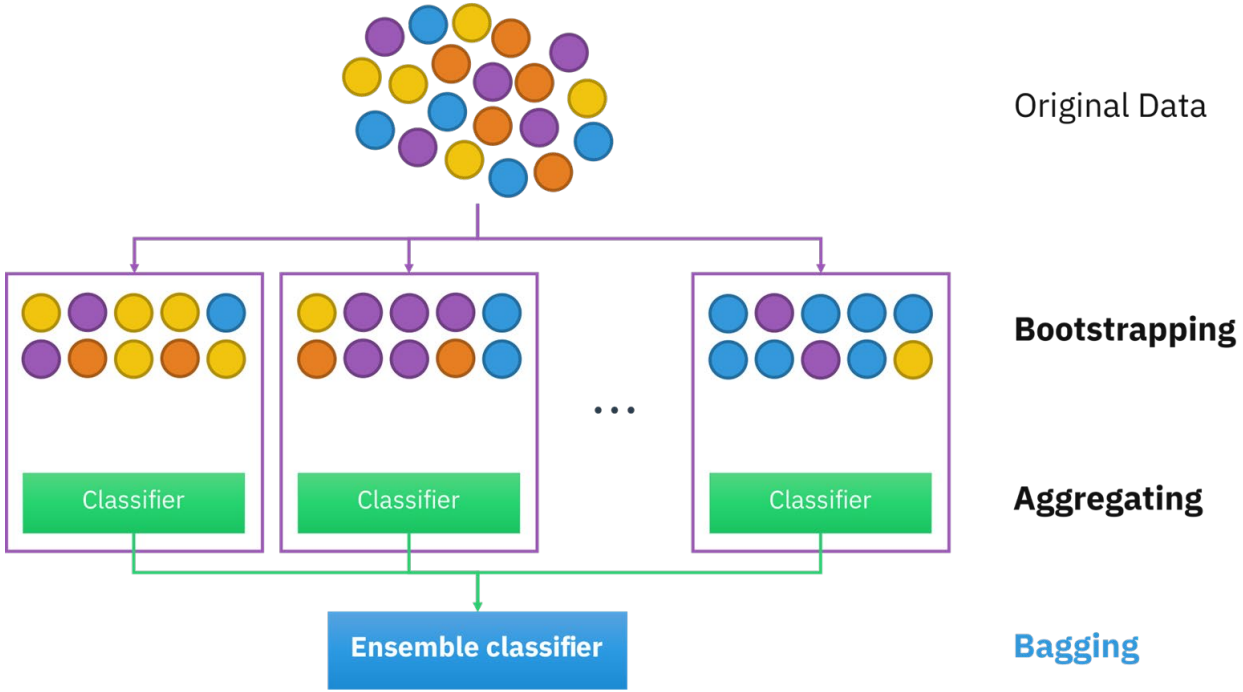




Zusammenfassung: Bootstrapping + Bagging

Bootstrapping: „neue“ Trainingsdaten aus „alten Daten“ generieren (Sub-sets, Variationen, …)

Bagging: Ensemble-Modell aggregieren

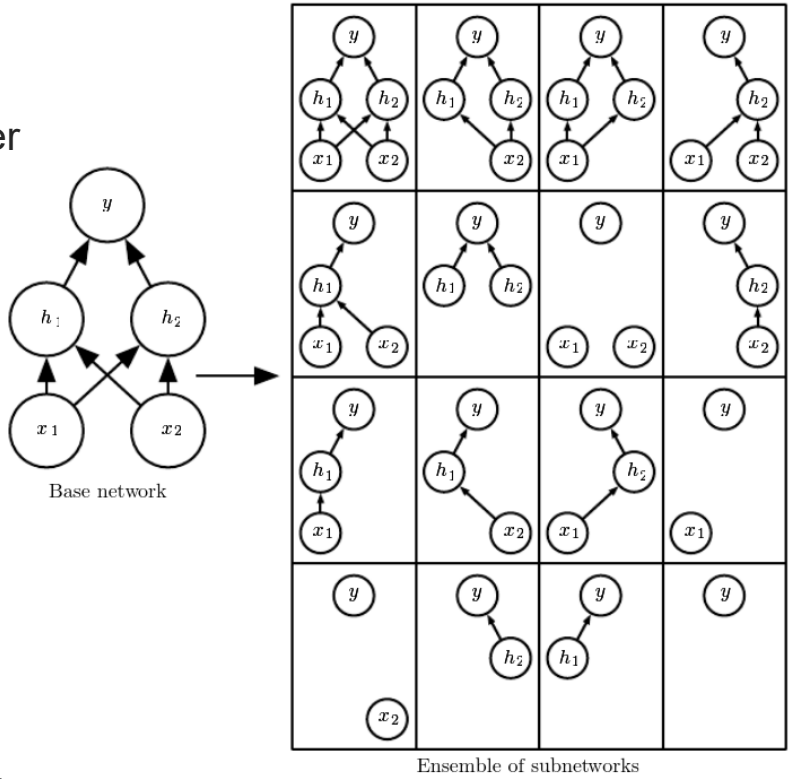


Sirakorn

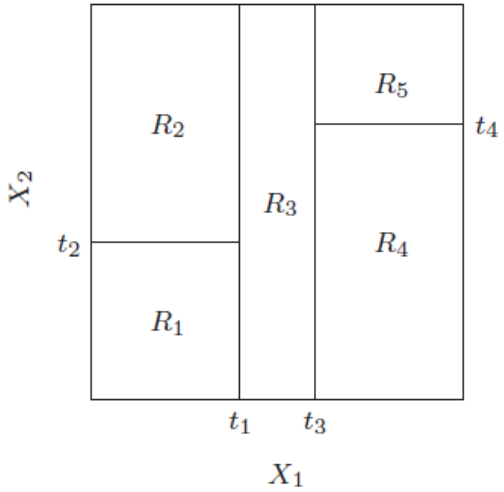
Dropout

Intuition

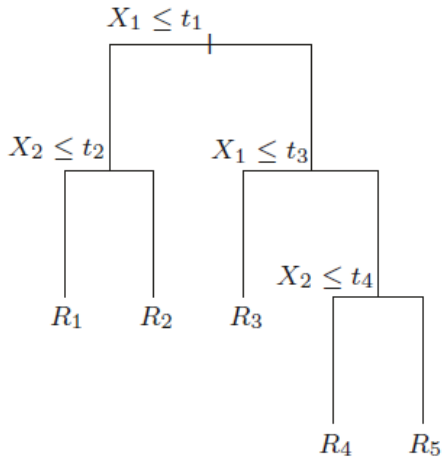
- **Bagging** von neuronalen Netzen
- Training von vielen neuronalen Netzen ist aber zu teuer
- Lösung: Betrachte ein neuronales Netz als Ensemble vieler Subnetze
- Für jeden Mini-Batch
  - Zufällige binäre Maske für alle Neuronen im neuronalen Netz
  - Update der Gewichte aller nicht-maskierten Neuronen
- Während Vorhersage
  - Keine Maske
  - Skalierung der Gewichte
  - → Effekt einer **Mittelung über alle Subnetze**

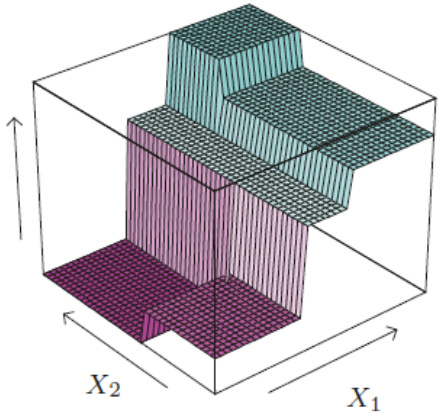


Ensemble-Lernen: Regressionsbäume



Für gegebenes X_1 and X_2, könnte das Regressionsmodell z.B. den Durchschnittswert von Y für die Trainingsdaten in R_m *vorhersagen*





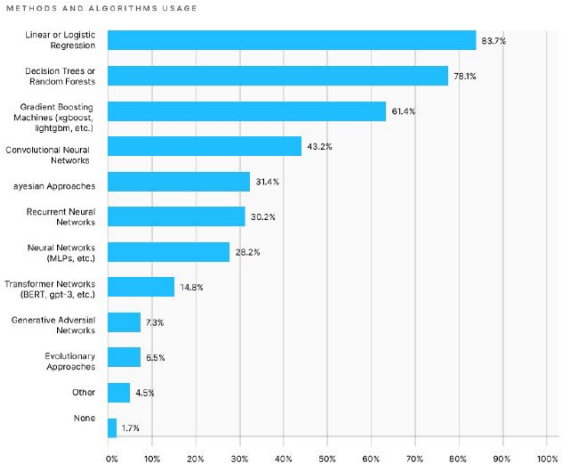
Wollen solche Bäume für **Boosting** nutzen!

Hastie et al. “The Elements of Statistical Learning.”

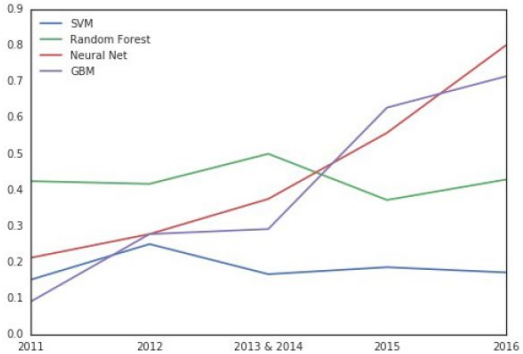
Ensemble-Lernen: Regressions-Bäume

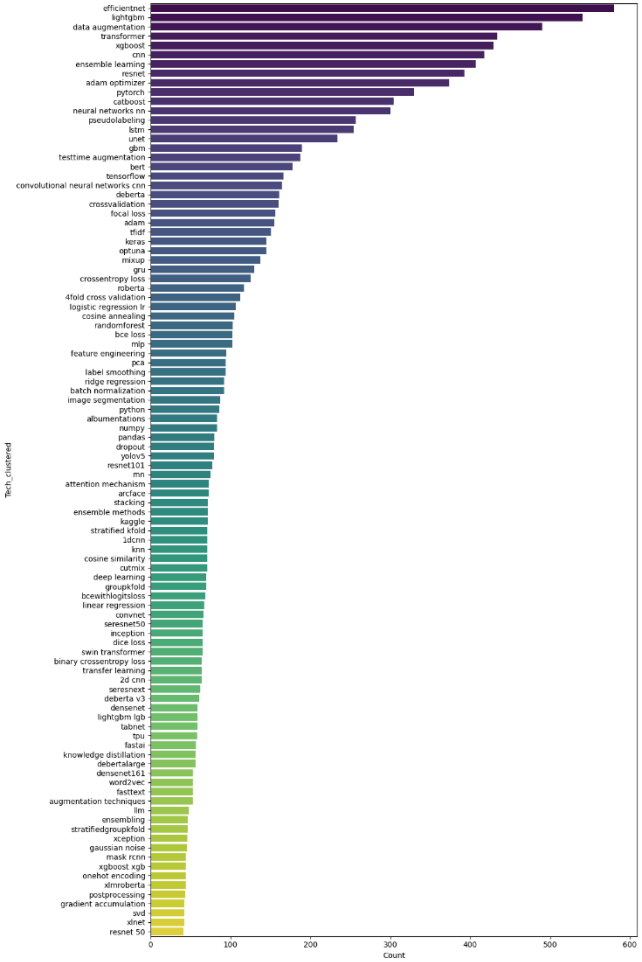
- Ziel: Zustandsraum in Regionen $R_1, R_2, \ldots, R_M$ aufteilen, sodass jede Region eine konstante Vorhersage erlaubt
- Model ist dann: $f(x) = \sum_{m=1} c_m I(x \in R_m)$
- Wir wählen die Vorhersagen als die Mittelwerte: $c_m = ave(y_i \vee x_i \in R_m)$
- Vorgehen für den Baum
  - Eine Variable auswählen, für die wir einen Trennwert finden wollen
  - Optimierungsproblem (z.B. mit MSE) lösen, um beste Trennung zu identifizieren
  - Wiederholen bis Baum auf gewünschte Größe gewachsen ist
  - Ggf. Baum wieder stutzen („pruning“), um Überanpassung zu verhindern.

Ensemble-Lernen: Motivation



Kaggle Gewinner Algorithmen

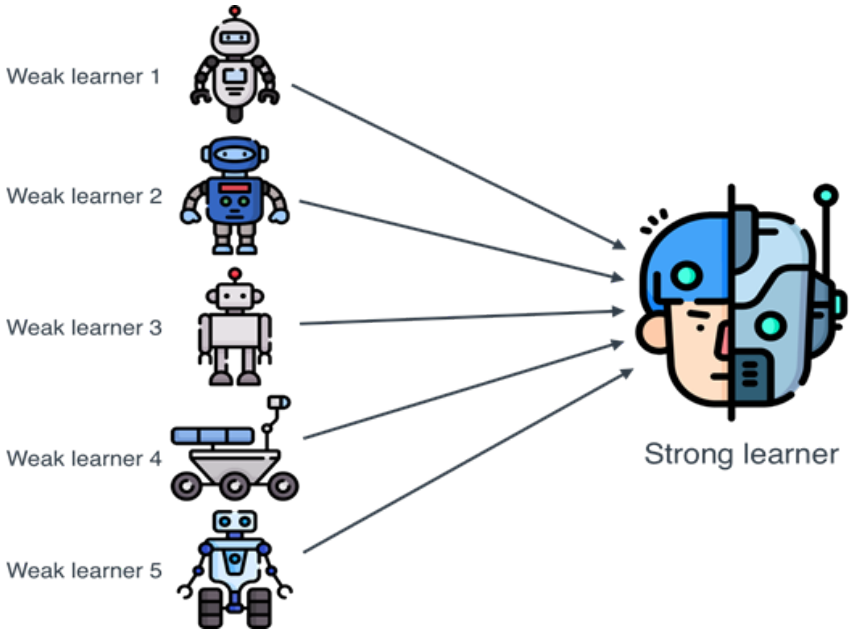




The top technologies mentioned in solution write-ups; July 2025

- Gewinner von Kaggle Challenges
- Neuronale Netze, aber auch Boosting-Methoden mit Bäumen werden viel benutzt!

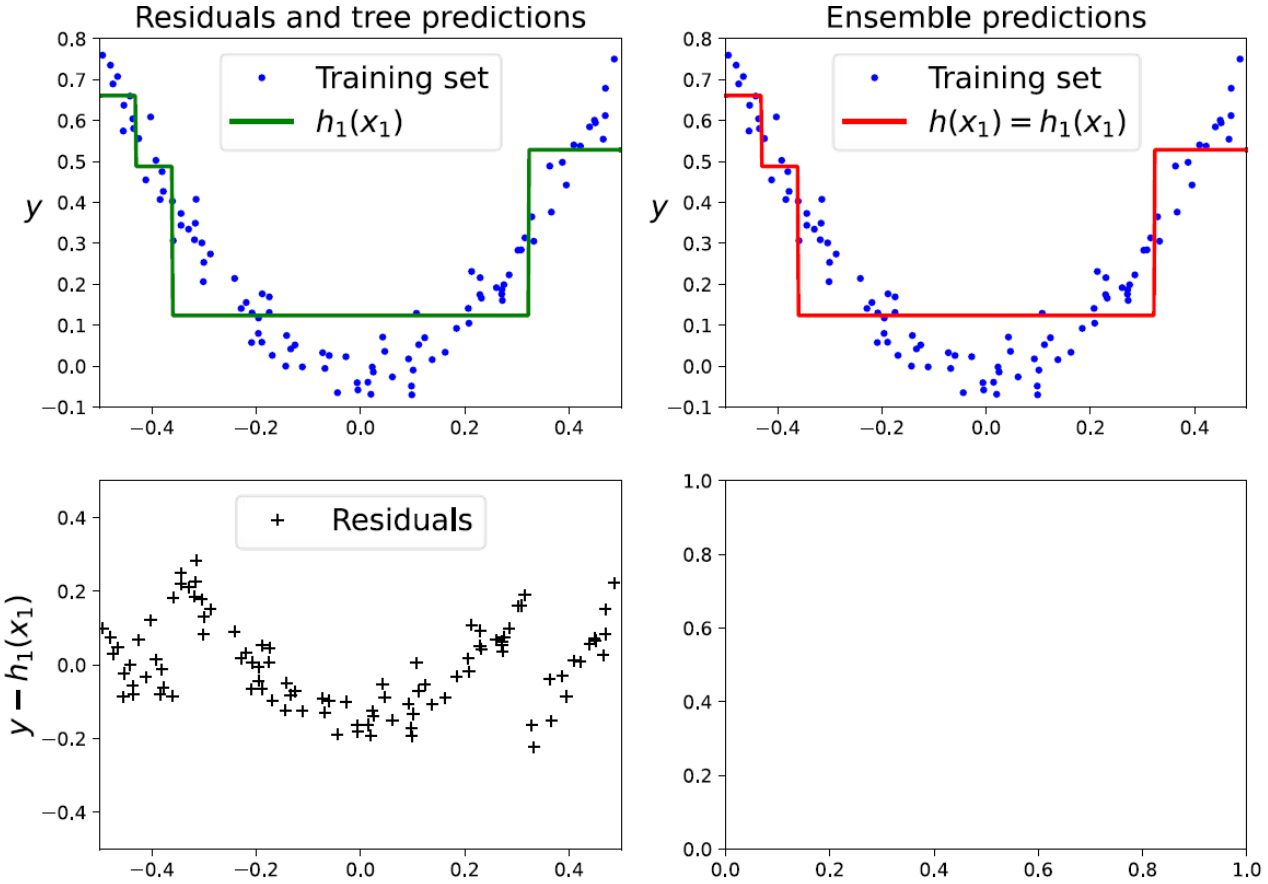
Ensemble-Lernen: Idee des Boosting



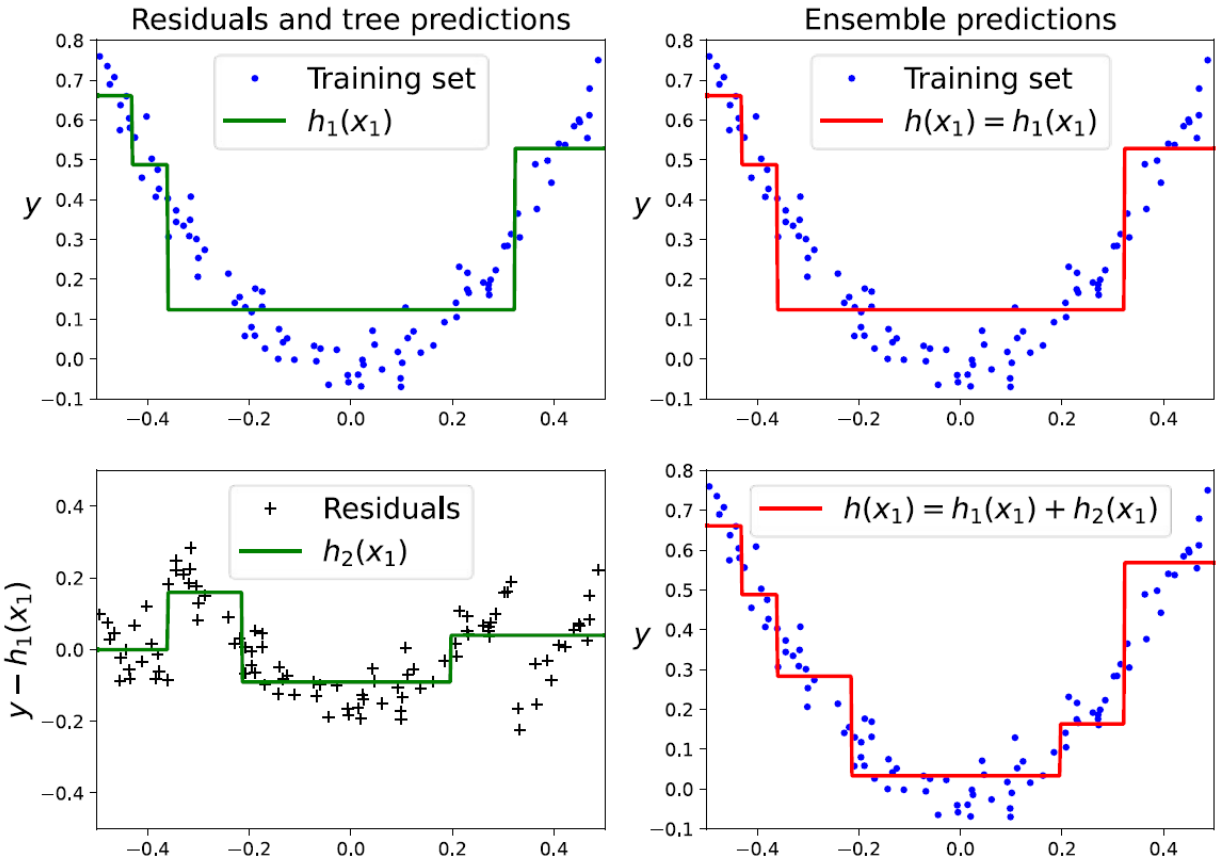
- Starte mit mehreren “schwachen Lernern”
  - Möglicherweise hoher Loss, aber besser als Raten
  - Z.B. lineare Regressionen, Entscheidungsbäume, …
- Kombiniere diese schwachen Lerner zu einem starken Lerner
- Intuition für Boosting: von den Fehlern der anderen lernen

Serrano

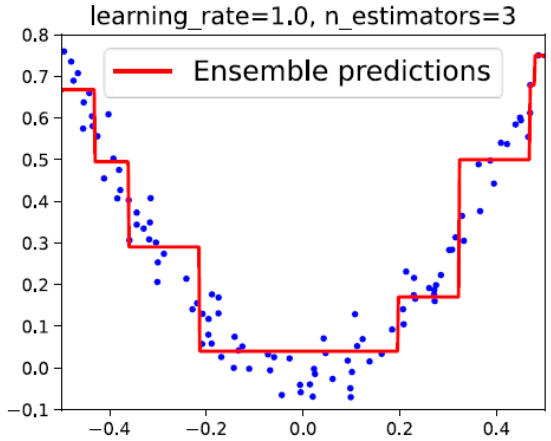
Ensemble-Lernen: Boosting-Beispiel

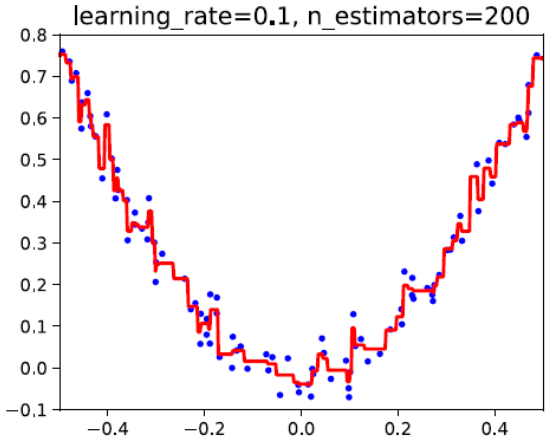


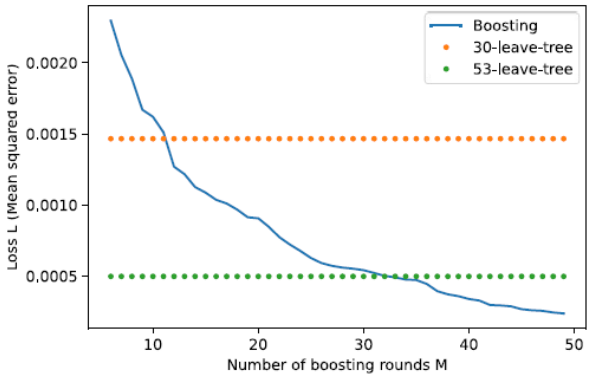
Ensemble-Lernen: Boosting-Beispiel



Ensemble-Lernen: Boosting-Beispiel







Erreichen die gleiche Genauigkeit wie große Bäume

Ensemble-Lernen: Boosting-Mathematik

- Sei S eine Menge von möglichen schwachen Lernern (z.B. Bäume)
- Gegeben ein schwacher Lerner $h_i \in S$ und eine Lernrate α, so ist die Ensemble-Funktion gegeben als $H(x) = \sum_{m=0}^{M} \alpha h_m$
- Wie bestimmen wir nun den nächsten Schritt?
  Wie finden wir den nächsten Lerner?
- $M \rightarrow M + 1$ und daher $H \rightarrow H + \alpha h$, aber welches h wählen wir?
- Idee: Den Fehler (loss) der Ensemble-Funktion minimieren
- Ansatz: Taylorreihe

$$L(H + \alpha H) : h = argmin_{h \in S}(L(H + \alpha h))$$

Ensemble-Lernen: Boosting-Mathematik

- Ziel: Minimiere $L(H+\alpha h): h = argmin_{h \in S}(L(H+\alpha h))$
- Taylor: $L(H+\alpha h) \approx L(H) + \frac{\partial L}{\partial H} \cdot \alpha h$

- *L(H)* hängt nicht vom Argument *h* ab und wird daher weggelassen

$$argmin_{h \in S}(L(H+\alpha h)) \approx argmin_{h \in S}(\frac{\partial L}{\partial H} \cdot \alpha h)$$

- Im $\mathbb{R}^n$ ist das innere Produkt das Skalarprodukt und daher vereinfacht sich dies zu

$$argmin_{h \in S}(\sum_{i=1}^{N}(\frac{\partial L}{\partial H(x_i)} \cdot \alpha h(x_i)))$$

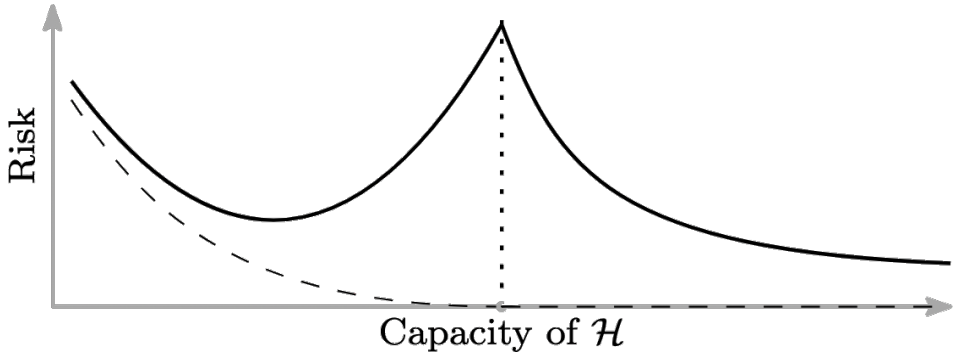
- Wenn wir nun den MSE minimieren, bedeutet dies

$$L(H,Y) = \frac{1}{2}\sum_{i=1}^{N}(y_i - H(x_i))^2 \rightarrow \frac{\partial L}{\partial H(x_i)} = y_i - H(x_i)$$

- In Worten: Wir minimieren die „Residuale“, also die Fehler des bisherigen Ensembles

Fragen?

Teil 5 (Optional)



1) Generalisierung

2) Hyperparameter

3) Regularisierung

4) Ensemble Methoden

5) Double Descent

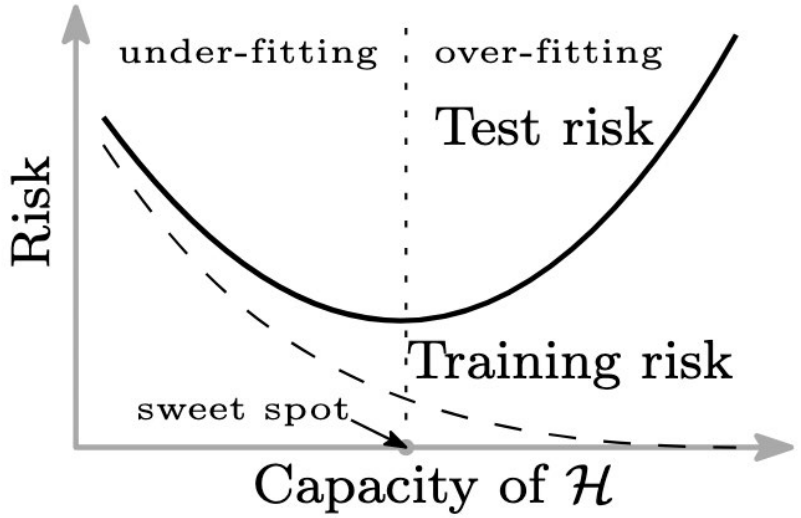
Bisher: Overfitting/Überanpassung

Theorie

- Zu kleine Modellkapazität
  → Underfitting/Unteranpassung
- Zu große Modellkapazität
  → Overfitting/Überanpassung

Praxis

- Sehen wir dieses Verhalten wirklich?



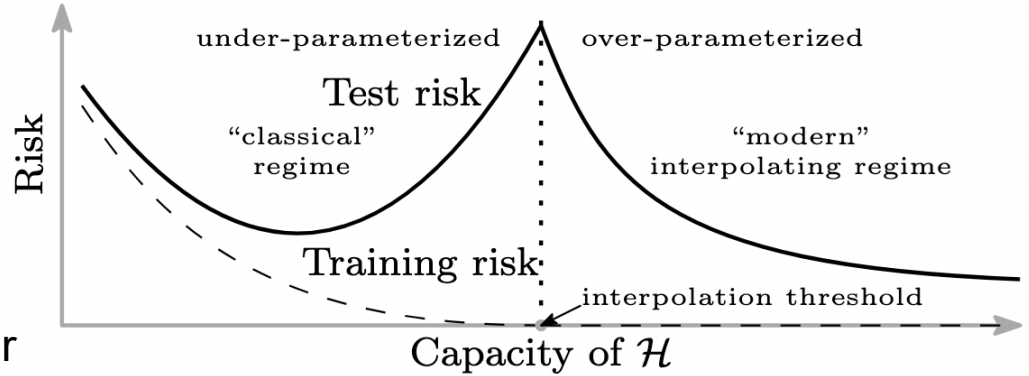
Double descent

Beobachtungen

- Das initiale Verhalten ist wie vorhergesagt
  - U-Form des Testfehlers

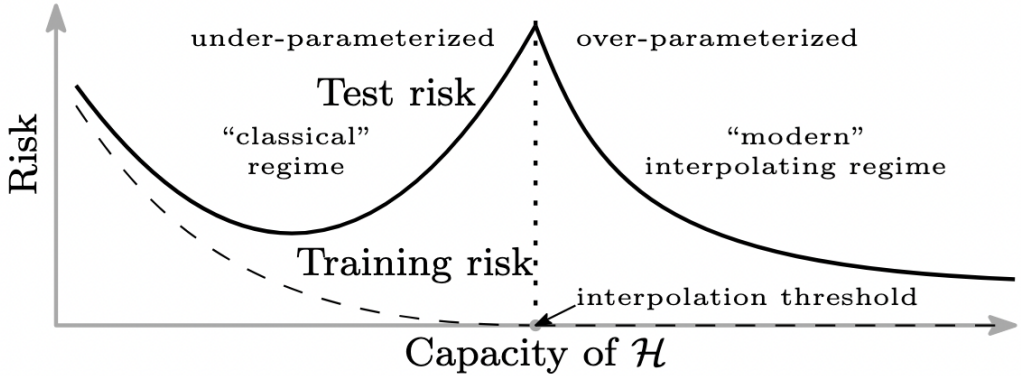
Aber

- Ab einem bestimmten Punkt reduziert sich der Testfehler wieder



Dieser Effekt wird „double descent“ genannt (relativ neuer Begriff von 2019)

Double descent



Classical Regime

- Bias-Variance Trade-Off

Interpolation Threshold

- Kapazität ist gerade ausreichend, um den Trainingsloss auf 0 zu reduzieren
- Zahl der Parameter ~ Zahl der Samples

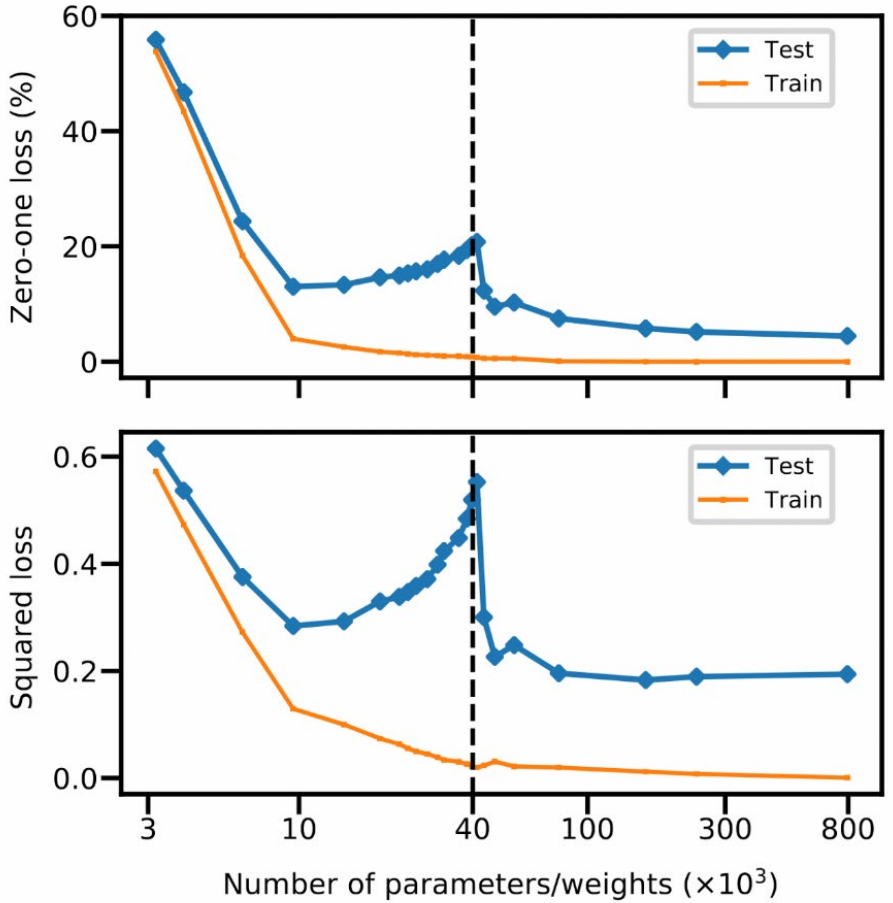
„Modern“ Interpolating Regime

- Trainingsloss ist 0
- Zahl der Parameter > Zahl der Samples
- Trotzdem gute Generalisierung

Double descent

Beispiel

- MNIST Datensatz
- Zuerst Überanpassung
- Dann bessere Generalisierung im „interpolation regime“ von sehr großen Modellen



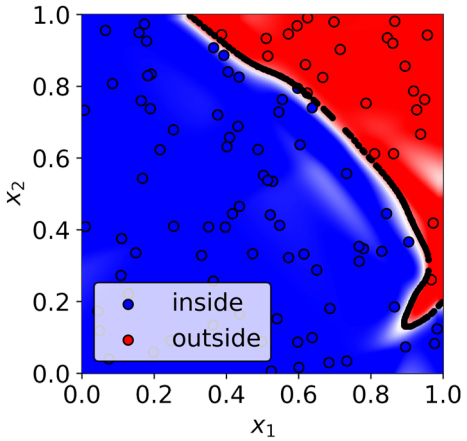
Mehr Daten, kleinerer Fehler?

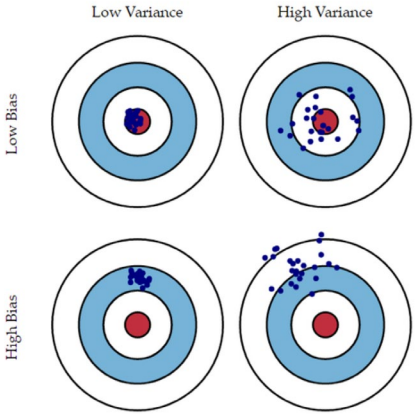
Interpolation threshold

- Je mehr Daten, desto höher
  → Daher können **mehr Daten** bei gleicher Modellgröße zu **schlechteren Ergebnissen** führen.

- Widerspruch zu vielen Grundannahmen im maschinellen Lernen!
- Nicht gut verstanden.

Zusammenfassung



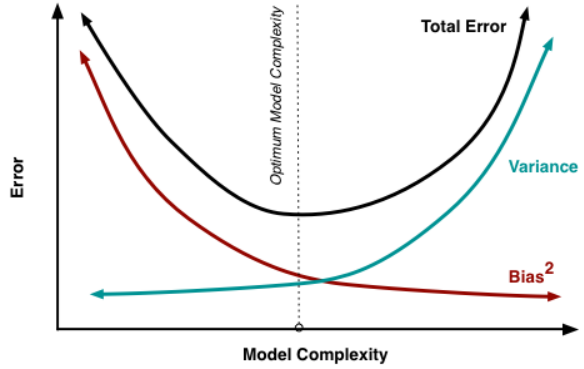


Parameter und Hyperparameter
- Hyperparametersuche

Regularisierung
- L1 und L2
- Early Stopping
- …

Ensembles

$$\mathrm{Var}[\hat{f}_{\mathrm{ens}}] = \frac{1}{m}(1-\rho)\sigma^2 + \rho\sigma^2$$



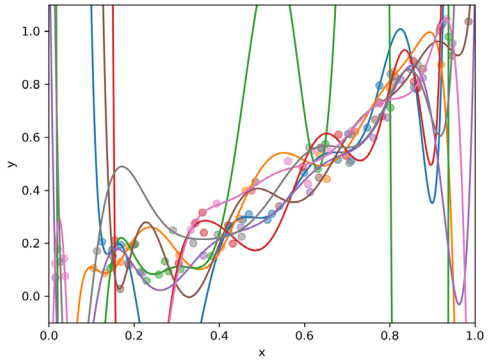
1) Generalisierung

2) Hyperparameter

3) Regularisierung

4) Ensemble Methoden

5) Double Descent



Fragen?

Choose your question

+ New question

GKI_25_26_VL6 [HIPFAX]

Go to app.wooclap.com to edit this event

Join this Wooclap event

1. Wie können wir aktiv Bias und Varianz in neuronalen Netzwerken beeinflussen? (bitte alle richtigen Antworten auswählen)

2. Für das Polynom 7ten Grades, haben wir...?

wooclap

Quick tip!

Note: we strongly recommend only adding questions from the same event.

Peer Nowack
peer.nowack@kit.edu