ETH zürich

Übungslektion 13 – Machine Learning II

Informatik II

20. / 21. Mai 2025

Willkommen!

Polybox

Passwort: jschul

Personal Website

`https://n.ethz.ch/~jschul`

Heutiges Programm

- Cross-Validation
- Grid Search für Polynomregression
- Neuronale Netze

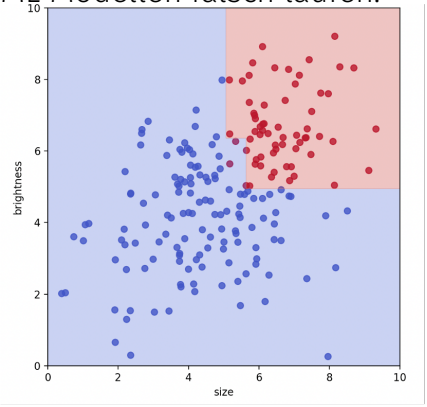
random_state

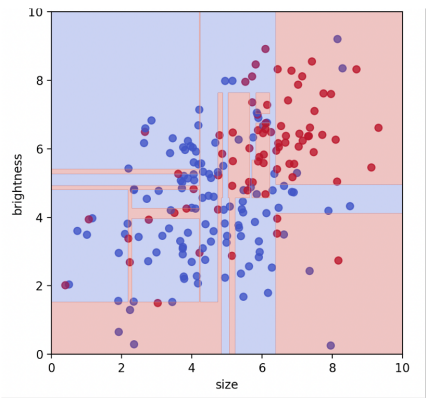
Basically: Wenn ihr einen random_state setzt, habt ihr einfach immer die **gleiche random Aufteilung** und erzielt dann jedes mal den **gleichen score**. **Euer Ergebnis wird reproduzierbar!**

1. Cross-Validation

Zusammenfassung

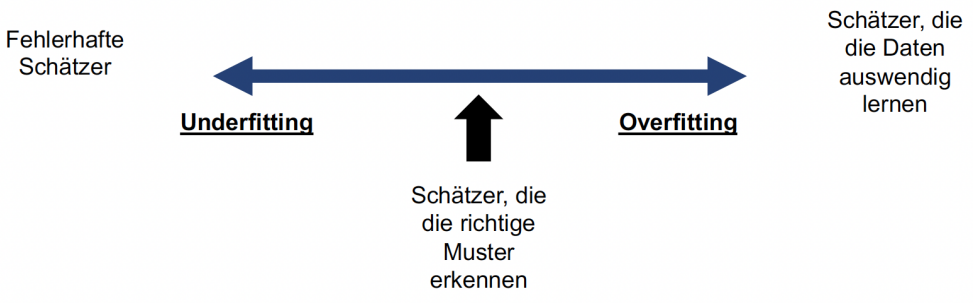
Wenn es Rauschen in dem Datensatz gibt, kann das Training von ML-Modellen falsch laufen.





Zusammenfassung

Um dies zu korrigieren, muss man die richtigen Hyperparameter für das Modell vor dem Training auswählen.



Grid-Search und K-Fold Cross-Validation

Grid-Search

Ziel ist es, die optimalen Parameter für ein Modell zu finden. Dafür übergeben wir Gridsearch verschiedene Werte die es testen soll, und es probiert dann alle möglich Kombinationen aus.
Z.B. "max_depth" : [1,2,3,4]

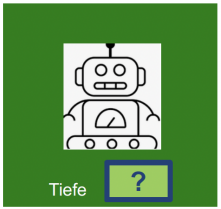
K-Fold Cross-Validation

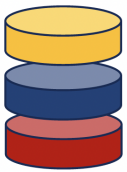
Anstatt unseren Datensatz einmal in Trainings- und Testdaten aufzuteilen, unterteilen wir ihn anfangs in K gleich grosse Teile und nehmen jedes mal einen davon als Test-Datensatz. In Kombination mit Grid-Search sehr stark!

Beide Methoden helfen uns, **optimale Parameter und Voraussetzungen** für unser Modell zu finden, und das meiste aus unseren Daten herauszuholen. Der **Nachteil** ist, dass das Training wesentlich länger dauert, da viel mehr Iterationen durchgeführt werden.

Zusammenfassung

Grid-Search mit Cross-Validation ist eine beliebte Methode, um die richtigen Hyperparameter zu finden.

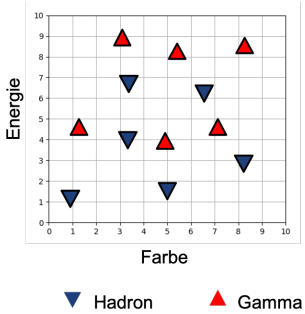




	Tiefe=2	Tiefe=20	Tiefe=200
% mistakes in	20	5	50
% mistakes in	15	3	80
% mistakes in	13	4	70

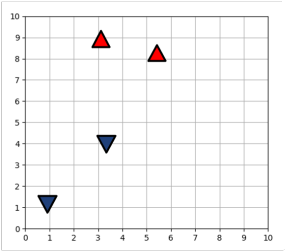
Übung

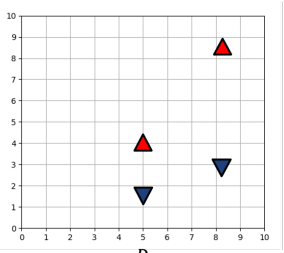
Betrachten Sie den folgenden Datensatz.

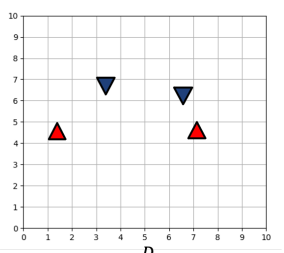


Übung

Der Datensatz ist wie folgt in drei Teile unterteilt.

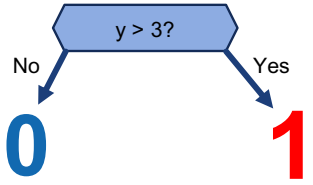






Übung

Das Modell besteht aus allen Bäumen, die nur eine Frage über die x- oder y-Koordinate stellen. Ein Schätzer in diesem Modell sieht so aus:

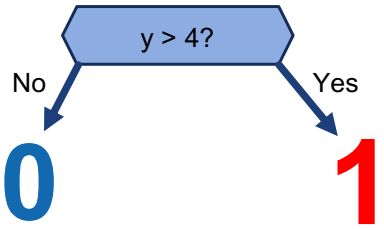


Übung

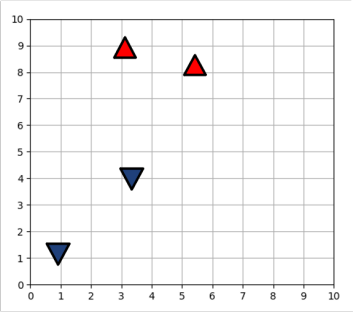
Berechnen Sie für jeden der drei Teile einen Schätzer für diesen Teil und bewerten Sie seine Genauigkeit anhand der anderen beiden Teile.

Lösung

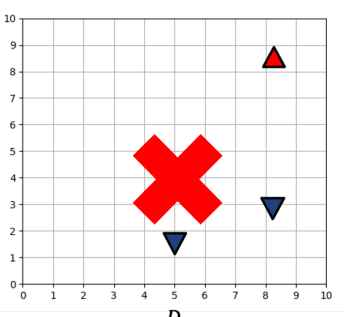
- Für D_1 betrachten Sie den folgenden Schätzer. Er prognostiziert 1 gdw $y > 4$.
- In D_2 macht der Schätzer 1 Fehler. In D_3 macht er 2 Fehler.



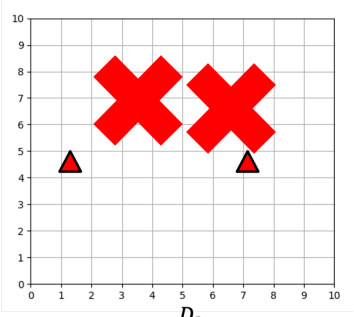
Lösung



D_1



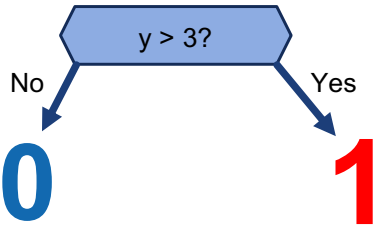
D_2



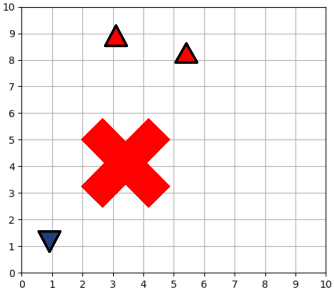
D_3

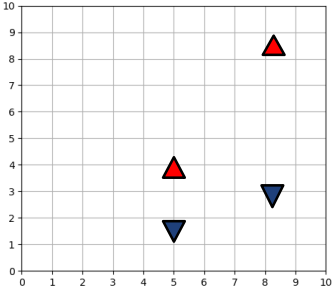
Lösung

- Für D_2 betrachten Sie den folgenden Schätzer. Er prognostiziert 1 gdw $y > 3$.
- In D_1 macht der Schätzer 1 Fehler. In D_3 macht er 2 Fehler.



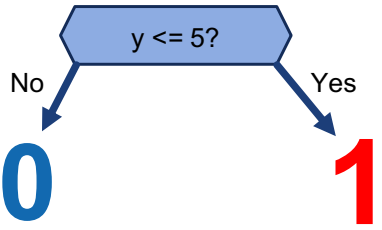
Lösung



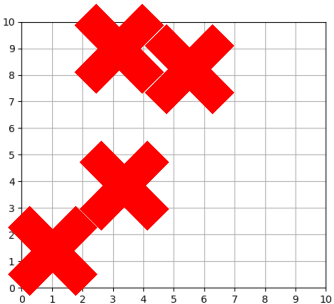


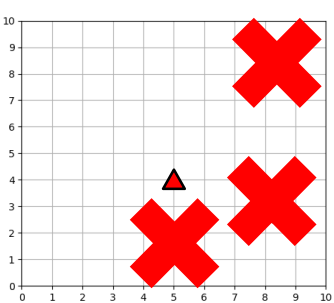
Lösung

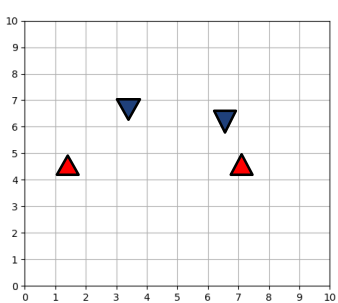
- Für D_3 betrachten Sie den folgenden Schätzer. Er prognostiziert 1 gdw $y \leq 5$.
- In D_1 macht der Schätzer 4 Fehler. In D_2 macht er 3 Fehler.



Lösung



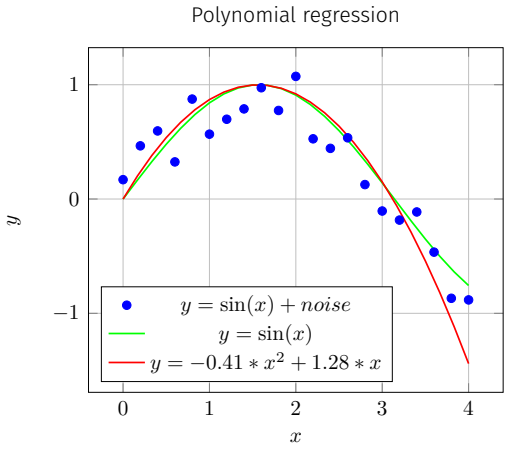




2. Grid-Search für Polynomregression

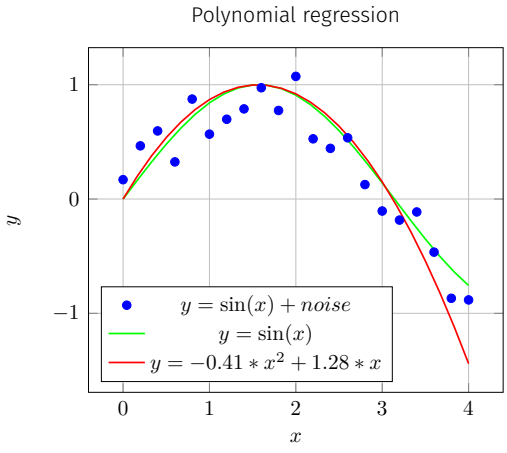
Übung

- Sie erhalten einen Datensatz $D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^2$ mit Punkten in $\mathbb{R}^2$.



Übung

- Sie erhalten einen Datensatz $D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^2$ mit Punkten in $\mathbb{R}^2$.
- Wir möchten ein Polynom f finden, so dass $f(x_i)$ so nahe wie möglich an y_i liegt, für $i \leq n$.



Übung

Angenommen, als Modell verwenden wir die Menge aller Polynome und als Verlustfunktion verwenden wir den mittleren quadratischen Fehler (MSE). Wir erinnern daran, dass der MSE, für einen Datensatz D und eine Funktion $f : \mathbb{R} \to \mathbb{R}$, wie folgt definiert ist:

$$MSE(D, f) = \sum_{i \leq n} \|y_i - f(x_i)\|^2.$$

Übung

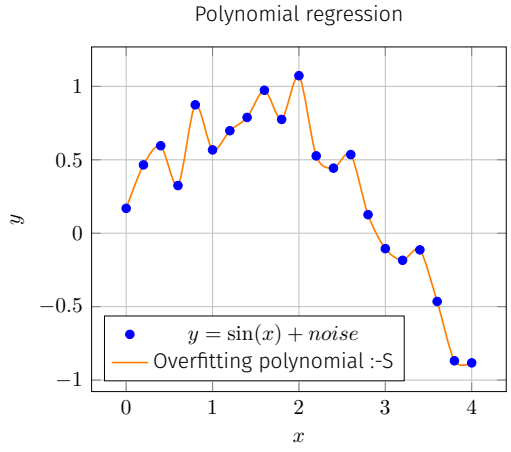
Beantworten Sie die folgenden Fragen:

- Wie kann es zu Overfitting bei diesem Datensatz kommen?

Übung

Beantworten Sie die folgenden Fragen:

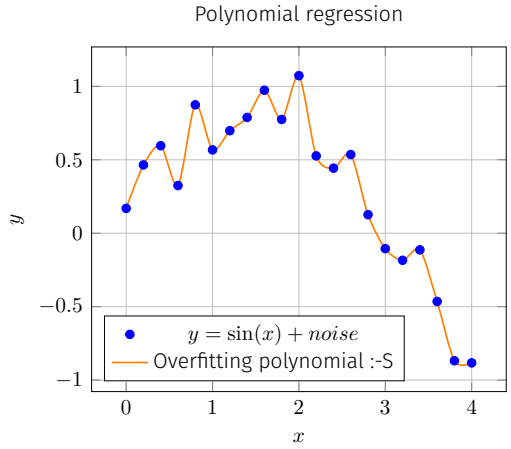
- Wie kann es zu Overfitting bei diesem Datensatz kommen? Ein Polynom kann durch alle Punkte hindurch verlaufen, ohne das korrekte zugrundeliegende Polynom zu lernen.



Übung

Beantworten Sie die folgenden Fragen:

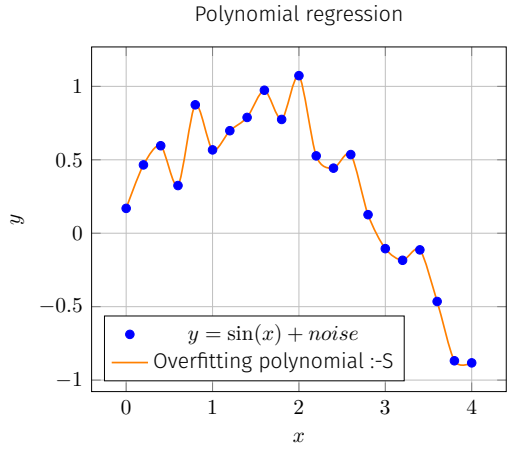
- Wie kann es zu Overfitting bei diesem Datensatz kommen? Ein Polynom kann durch alle Punkte hindurch verlaufen, ohne das korrekte zugrundeliegende Polynom zu lernen.
- Welchen Hyperparameter können wir steuern, um Overfitting zu vermeiden?



Übung

Beantworten Sie die folgenden Fragen:

- Wie kann es zu Overfitting bei diesem Datensatz kommen? Ein Polynom kann durch alle Punkte hindurch verlaufen, ohne das korrekte zugrundeliegende Polynom zu lernen.
- Welchen Hyperparameter können wir steuern, um Overfitting zu vermeiden? Den Grad des Polynoms.



Übung

Beschreiben Sie die Schritte zur Anpassung eines Polynoms an einen Datensatz mit Grid-Search und Cross-Validation.

Übung

Beschreiben Sie die Schritte zur Anpassung eines Polynoms an einen Datensatz mit Grid-Search und Cross-Validation.

- Wählen Sie eine Menge $C = \{d_1, \ldots, d_m\} \subseteq \mathbb{N}$ von Kandidaten für die Grade der Polynome.

Übung

Beschreiben Sie die Schritte zur Anpassung eines Polynoms an einen Datensatz mit Grid-Search und Cross-Validation.

- Wählen Sie eine Menge $C = \{d_1, \ldots, d_m\} \subseteq \mathbb{N}$ von Kandidaten für die Grade der Polynome.
- Teilen Sie den Datensatz in k Teile $D_1, \ldots, D_k$ (Falten) auf

Übung

Beschreiben Sie die Schritte zur Anpassung eines Polynoms an einen Datensatz mit Grid-Search und Cross-Validation.

- Wählen Sie eine Menge $C = \{d_1, \ldots, d_m\} \subseteq \mathbb{N}$ von Kandidaten für die Grade der Polynome.
- Teilen Sie den Datensatz in k Teile $D_1, \ldots, D_k$ (Falten) auf
  - Für jeden Teil $i = 1, ..., k$ und jedes $d \in C$, tun Sie folgendes.

Übung

Beschreiben Sie die Schritte zur Anpassung eines Polynoms an einen Datensatz mit Grid-Search und Cross-Validation.

- Wählen Sie eine Menge $C = \{d_1, \ldots, d_m\} \subseteq \mathbb{N}$ von Kandidaten für die Grade der Polynome.
- Teilen Sie den Datensatz in k Teile $D_1, \ldots, D_k$ (Falten) auf
  - Für jeden Teil $i = 1, ..., k$ und jedes $d \in C$, tun Sie folgendes.
  - Trainieren Sie ein Polynom $f_{i,d}$ des Grades d auf $\bar{D}_i$, wobei $\bar{D}_i = D \setminus D_i$.

Übung

Beschreiben Sie die Schritte zur Anpassung eines Polynoms an einen Datensatz mit Grid-Search und Cross-Validation.

- Wählen Sie eine Menge $C = \{d_1, \ldots, d_m\} \subseteq \mathbb{N}$ von Kandidaten für die Grade der Polynome.
- Teilen Sie den Datensatz in k Teile $D_1, \ldots, D_k$ (Falten) auf
  - Für jeden Teil $i = 1, ..., k$ und jedes $d \in C$, tun Sie folgendes.
  - Trainieren Sie ein Polynom $f_{i,d}$ des Grades d auf $\bar{D}_i$, wobei $\bar{D}_i = D \setminus D_i$.
  - Berechnen Sie den MSE $e_{i,d} = MSE(D_i, f_{i,d})$.

Übung

Beschreiben Sie die Schritte zur Anpassung eines Polynoms an einen Datensatz mit Grid-Search und Cross-Validation.

- Wählen Sie eine Menge $C = \{d_1, \ldots, d_m\} \subseteq \mathbb{N}$ von Kandidaten für die Grade der Polynome.
- Teilen Sie den Datensatz in k Teile $D_1, \ldots, D_k$ (Falten) auf
  - Für jeden Teil $i = 1, ..., k$ und jedes $d \in C$, tun Sie folgendes.
  - Trainieren Sie ein Polynom $f_{i,d}$ des Grades d auf $\bar{D}_i$, wobei $\bar{D}_i = D \setminus D_i$.
  - Berechnen Sie den MSE $e_{i,d} = \mathit{MSE}(D_i, f_{i,d})$.
- Geben Sie das Polynom f^* aus, das den kleinsten MSE über alle k Teile erreicht hat. Das heißt,

$$f^* = f_{d^*} \quad \text{wobei} \quad d^* = \arg\min_d \left(\sum_{i=1}^{k} e_{i,d} \right)$$

GridSearchCV

GridSearchCV Dokumentation

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
…
pam = {  # folgend zwei Beispiele für Bäume
   "max_depth" : range(1, 4),
   "criterion" : ["squared_error", "friedman_mse"]}
mdl = DecisionTreeRegressor()  # Modell wählen
grid = GridSearchCV(estimator=mdl, param_grid=pam, cv=3)
grid.fit(X, y)  # Cross-Validierung mit Grid-Search
```

Figure: Max Schaldach

Code Expert Übung 1

Implementieren Sie nun in Code Expert ein Python-Programm, das Grid-Search und Cross-Validation verwendet, um Polynome anzupassen.
Grid search for polynomials

3. Neuronale Netze

Wiederholung

- Neuronale Netze gehören zu den leistungsfähigsten Maschinenlernmodellen.

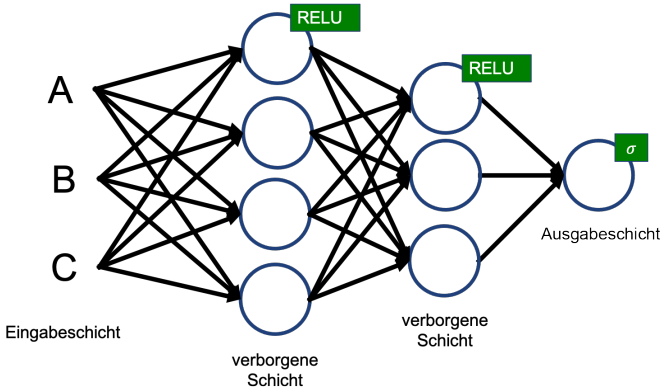
Wiederholung

- Neuronale Netze gehören zu den leistungsfähigsten Maschinenlernmodellen.
- Ihre Leistungsfähigkeit resultiert aus der Fähigkeit, automatisch Merkmale zu berechnen, die zur Lösung der Aufgabe relevant sind.

Wiederholung

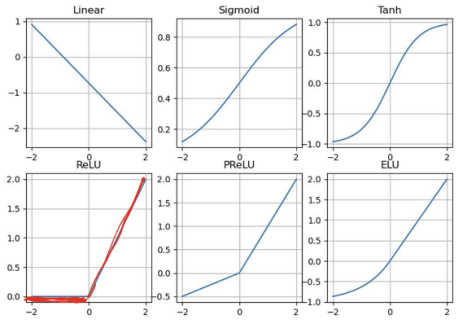
- Neuronale Netze gehören zu den leistungsfähigsten Maschinenlernmodellen.
- Ihre Leistungsfähigkeit resultiert aus der Fähigkeit, automatisch Merkmale zu berechnen, die zur Lösung der Aufgabe relevant sind.
- Ein neuronales Netz ist das Ergebnis von mehreren Schichten von Arrays logistischer Regressoren.

Wiederholung



Aktivierungsfunktionen

- Nichtlineare Aktivierungsfunktionen ermöglichen die **Modellierung komplexerer Abhängigkeiten**
- Häufigste Wahl: ReLU (und Varianten)



Name in PyTorch	Equation
`Linear` Layer* (Identity function)	$f(x) = x$
`Sigmoid`	$f(x) = \frac{1}{1+e^{-x}}$
`Tanh`	$f(x) = \tanh(x)$
`ReLU` (Rectified Linear Unit)	$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$
`PReLU` (Parametric ReLU)	$f(x) = \begin{cases} a \cdot x, & x < 0 \\ x, & x \geq 0 \end{cases}$
`ELU` (Exponential L.U.)	$f(x) = \begin{cases} a(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$

* Die Schicht als Ganzes berechnet $f(x) = a \cdot x + b$ (bzw. $f(x) = \mathbf{w} \cdot x$ im multidimensionalen Fall). Die Parameter a, b bzw. $\mathbf{w}$ gehören jedoch zu den Neuronen und nicht zur Aktivierungsfunktion.

Zufällige Initialisierung

Die zufällige Initialisierung von Gewichten ist ein wesentlicher Schritt beim Training neuronaler Netze.

- Wenn alle Neuronen mit dem gleichen Anfangsgewicht beginnen, lernen sie die gleichen Merkmale und das Netzwerk kann keine komplexen Muster lernen.
- In Code Expert fixieren wir die Startwerte des Zufallszahlengenerators mit einem *random seed*, um stets die dieselben Ergebnisse zu erhalten (Reproduzierbarkeit).

Playground

- **Coole Visualisierung:**
  http://playground.tensorflow.org
- **YouTube Explainers:**
  3Blue1Brown

Übung

Beantworten Sie die folgenden Fragen:

- Warum übertreffen neuronale Netze andere Maschinenlernmodelle wie Entscheidungsbäume oder logistische Regression?

Übung

Beantworten Sie die folgenden Fragen:

- Warum übertreffen neuronale Netze andere Maschinenlernmodelle wie Entscheidungsbäume oder logistische Regression? Neuronale Netze können komplexere Flächen im euklidischen Raum beschreiben.

Übung

Beantworten Sie die folgenden Fragen:

- Warum übertreffen neuronale Netze andere Maschinenlernmodelle wie Entscheidungsbäume oder logistische Regression? Neuronale Netze können komplexere Flächen im euklidischen Raum beschreiben.
- Warum übertreffen tiefe neuronale Netze (Netze mit mehreren Schichten) flache neuronale Netze?

Übung

Beantworten Sie die folgenden Fragen:

- Warum übertreffen neuronale Netze andere Maschinenlernmodelle wie Entscheidungsbäume oder logistische Regression? Neuronale Netze können komplexere Flächen im euklidischen Raum beschreiben.
- Warum übertreffen tiefe neuronale Netze (Netze mit mehreren Schichten) flache neuronale Netze? Jede Schicht in einem neuronalen Netz lernt, Muster zu erkennen, die auf den von vorherigen Schichten erkannten Mustern basieren. Daher lernen tiefere Schichten, kompliziertere Muster zu erkennen als die ersten Schichten.

Übung

Beantworten Sie die folgende Fragen:

- Was ist der Unterschied zwischen ReLU und Sigmoid?

Übung

Beantworten Sie die folgende Fragen:

- Was ist der Unterschied zwischen ReLU und Sigmoid? Eine Sigmoidfunktion gibt einen Wert zwischen 0 und 1 aus und wird für Klassifizierungszwecke verwendet; das heißt, wenn wir zwischen zwei Klassen von Objekten unterscheiden wollen. Sigmoidfunktionen werden in der Regel für die letzte Schicht verwendet. ReLU gibt einen Wert zwischen 0 und ∞ aus und wird normalerweise für alle anderen Schichten verwendet.

MLPClassifier

```
from sklearn.neural_network import MLPClassifier
…
mlp = MLPClassifier(hidden_layer_sizes=(100,50),
max_iter=200, random_state=42)  # beliebige Werte
mlp.fit(X, y)
…
```

32,32,16,16,8

theroetisch für params: {'hidden_layer_sizes': [(128,),(64,32),(64,128,32)]}

wobei der **Hyperparameter** `hidden_layer_sizes` die Anzahl der neuronalen Schichten und ihrer Knoten spezifiziert, `max_iter` die Anzahl an Iterationen des Algorithmus angibt, und `random_state` sicherstellt, dass das Resultat reproduzierbar ist

Figure: Max Schaldach

Code Expert Übung 2

Programmieraufgabe: Implementieren Sie ein Python-Skript, das ein neuronales Netz trainiert. Es muss lernen, gelbe von lila Punkten zu unterscheiden, wie in der Abbildung gezeigt.

DecisionTrees

(4.a.2)

Entscheidungsbäumen sind immer balanciert. / *Decision trees are always be balanced.*

- ◯ Wahr / *True*
- ☒ Falsch / *False*

CrossValidation

(4.a.1)

In der K-Fold-Cross-Validation teilen wir den Datensatz in K Teile. Dann teilen wir jeden Teil in einen Trainings- und einen Testsatz. Anschliessend berechnen wir für jeden Teil einen Schätzer mit dem Trainingssatz und berichten dann über die Leistung auf dem entsprechenden Testsatz. / *In K-fold cross-validation, we split the data into K parts. We then split each part into training and testing set. Then, for each part, we compute an estimator using the training set, and then report the performance on its testing set.*

◯ Wahr / *True*

☒ Falsch / *False*

Over- Underfitting

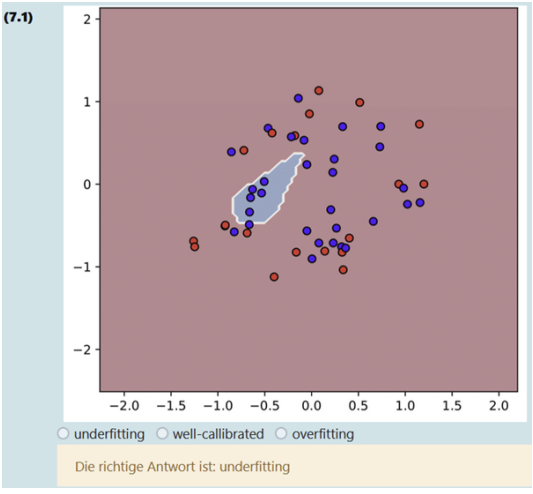
(4.a.3)

Underfitting tritt auf, wenn der Schätzer eines Modells auf den Trainingsdaten sehr gut, aber auf den Testsatz viel schlechter abschneidet. / *Underfitting occurs when the estimator of a model performs very well on the training data, but much worse on the testing data.*

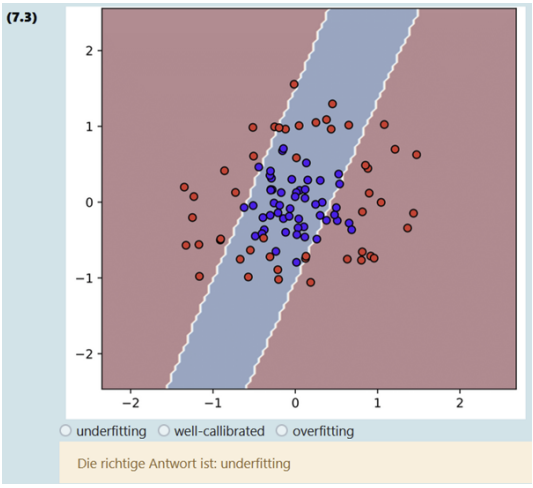
◯ Wahr / *True*

☒ Falsch / *False*

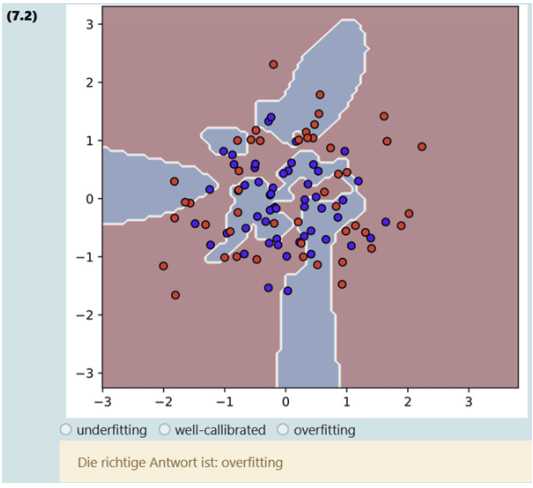
Over- Underfitting



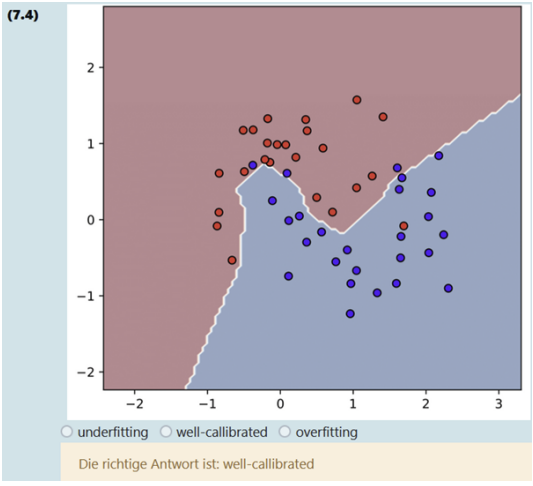
Over- Underfitting



Over- Underfitting



Over- Underfitting



5. Hausaufgaben

Exercise 11: Intro ML II

Auf https://expert.ethz.ch/enrolled/SS25/mavt2/exercises

- Spirals
- Diabetes with regression trees and grid search
- Confusion Matrix
- Sine Regression

Abgabedatum: Montag 26.05.2025, 20:00 MEZ

NO HARDCODING

Feedback

https://n.ethz.ch/~jschul/Feedback