Informatik II Woche 13

Cross-Validation, Grid-Search, Neuronale Netze

Website: n.ethz.ch/~kvaratharaja/

Die Slides basieren auf den offiziellen Übungsslides der Kurswebsite: https://lec.inf.ethz.ch/mavt/informatik2/2025/

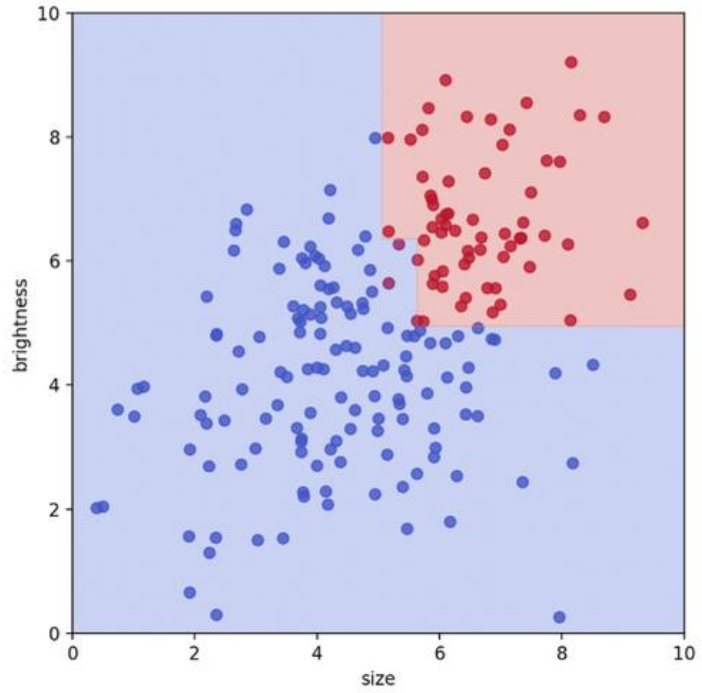
Heute

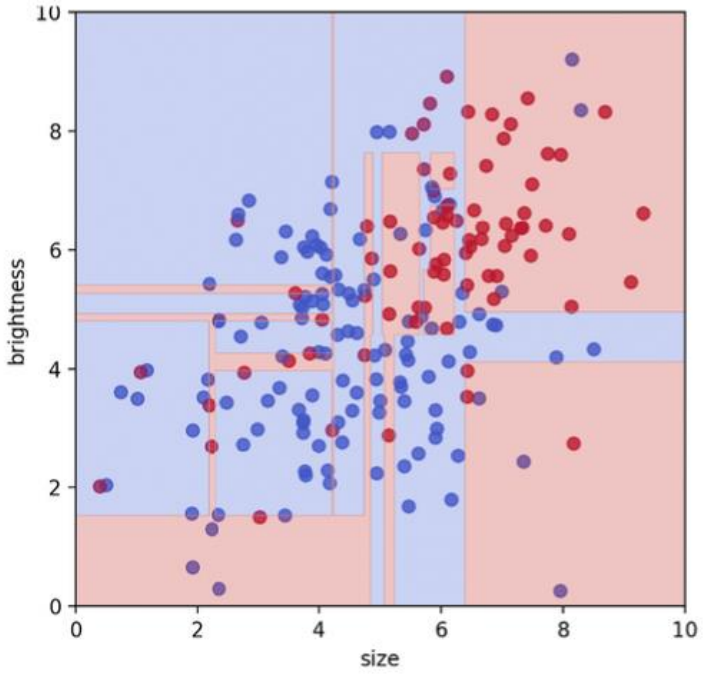
1. **Cross-Validation**
2. **Grid Search für Polynomregression**
3. **Neuronale Netze**
4. **Hausaufgaben**

1. Cross-Validation

Zusammenfassung

Wenn es Rauschen im 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.

Fehlerhafte Schätzer

Schätzer, die die Daten auswendig lernen

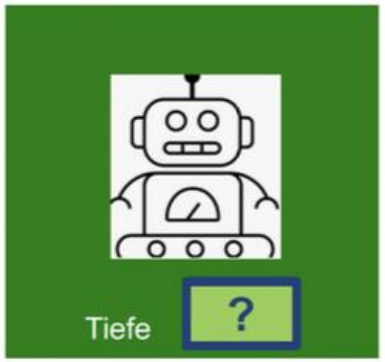
Underfitting

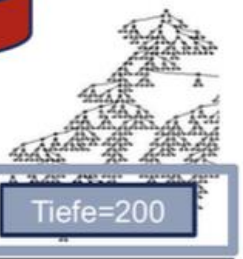
Overfitting

Schätzer, der das richtige Muster erkennt

Zusammenfassung

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



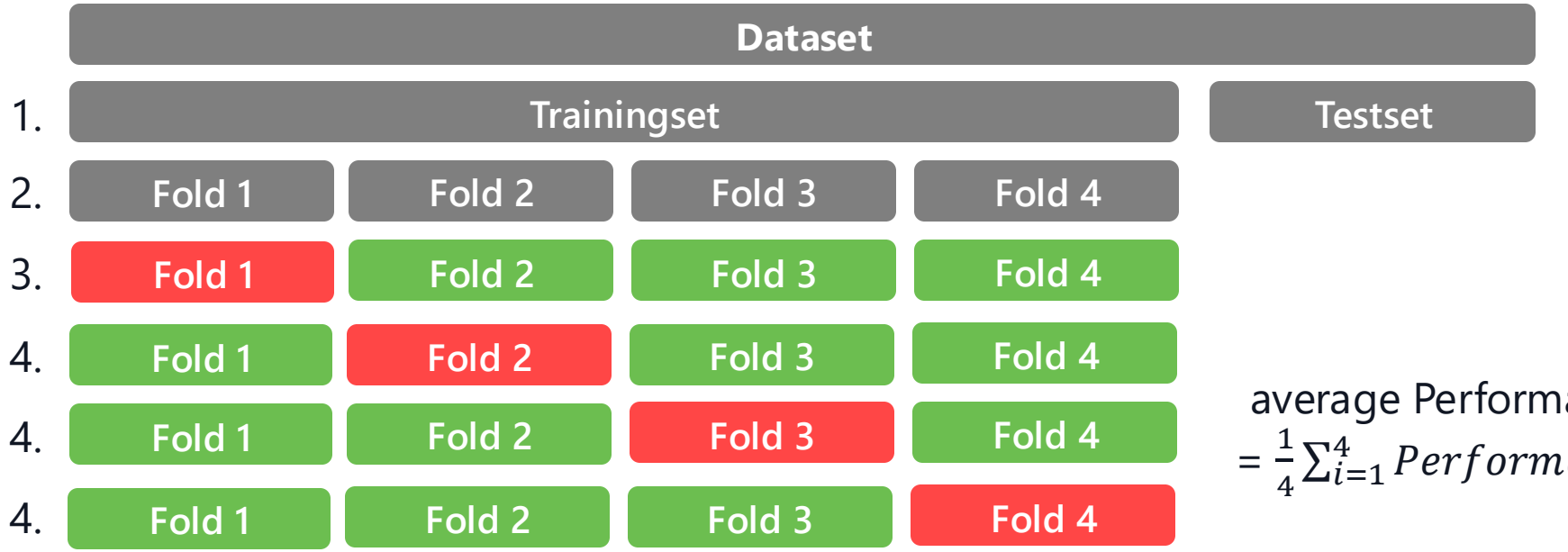


	Tiefe=2	Tiefe=20	Tiefe=200
% mistakes in	20	5	50
% mistakes in	15	3	80
% mistakes in ETH Zürich	13	4	70

K-fold Cross-Validation

Split Datensatz in **Trainings-** und **Testsatz**

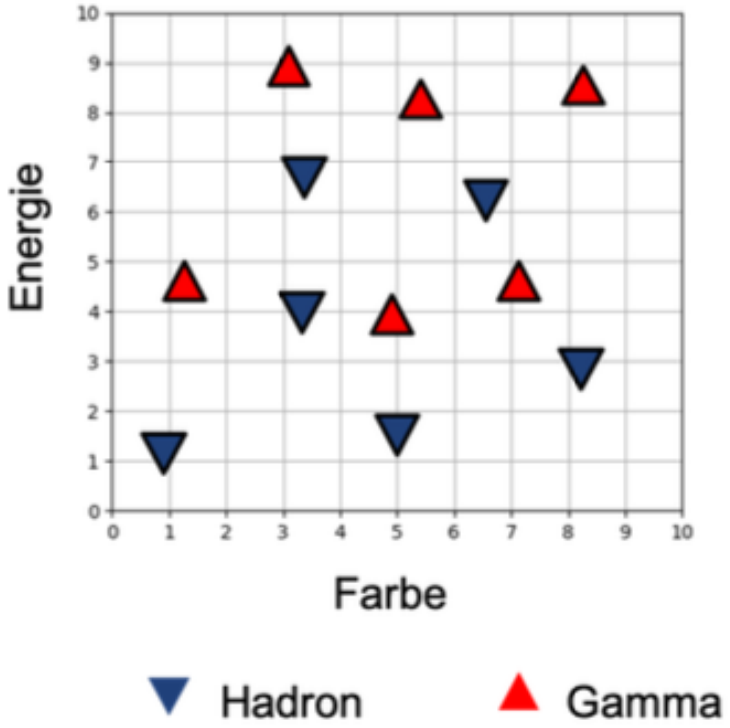
1. Split Trainingssatz in **k subsets (K-Folds)**
2. Benutze k-1 folds für **training** und 1 fold für **validation**
3. Wiederhole 3. Schritt mit verschiedenen training/validation fold Kombinationen und berechne die average performance (Loss)



average Performance $= \frac{1}{4}\sum_{i=1}^{4} Performance_i$

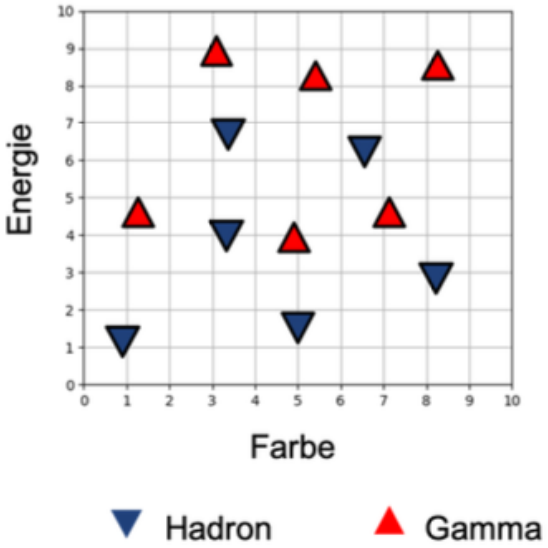
Übung

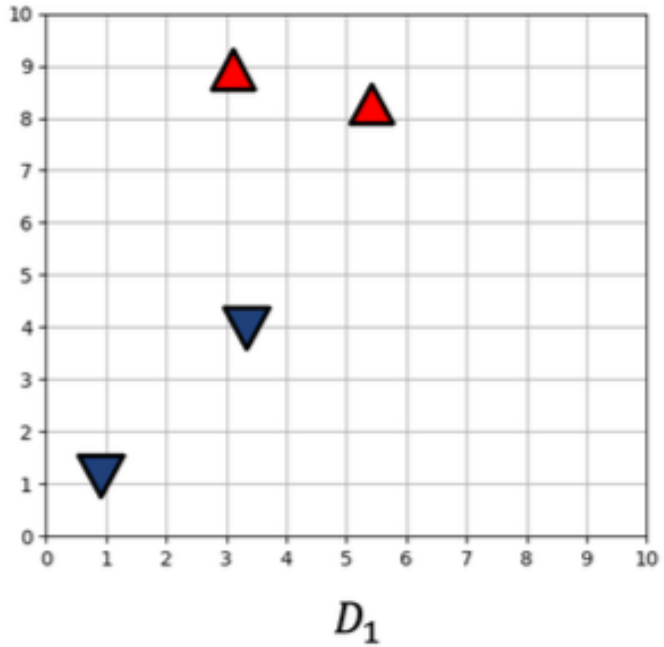
Betrachte folgenden Datensatz.

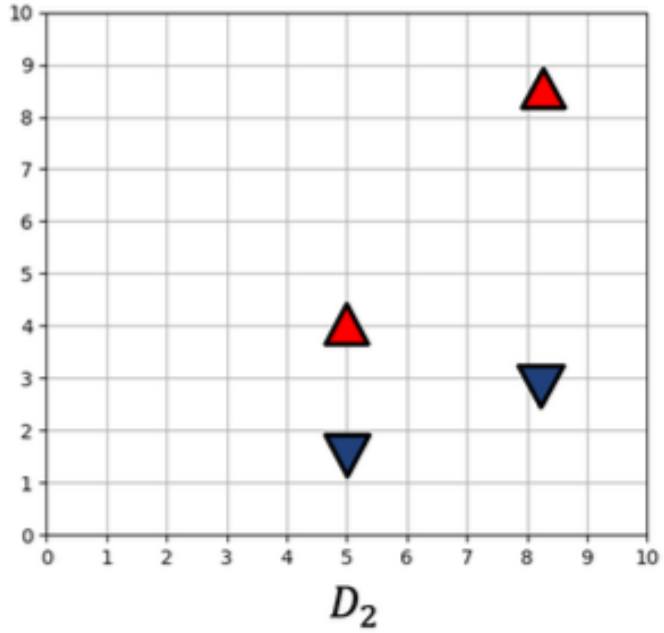


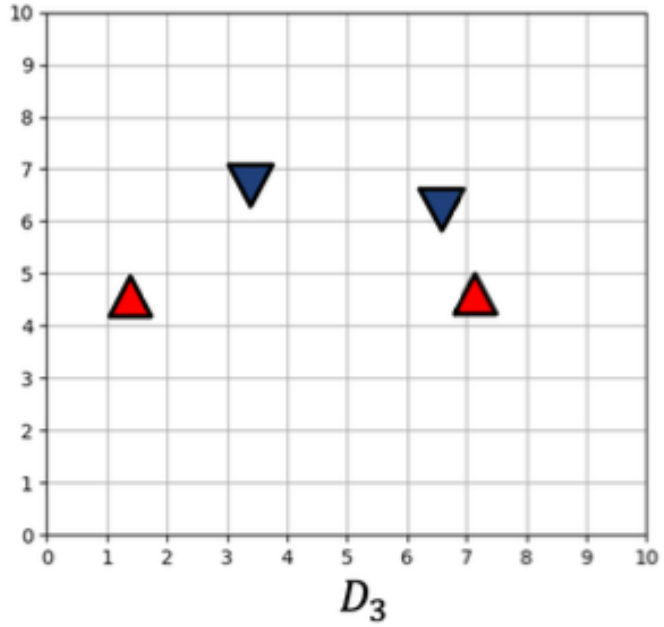
Übung

Der Datensatz wird in folgende drei Teile unterteilt:

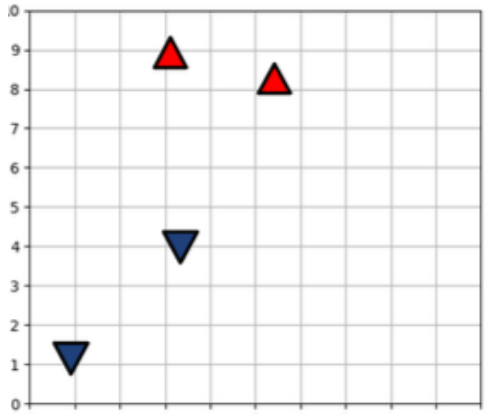


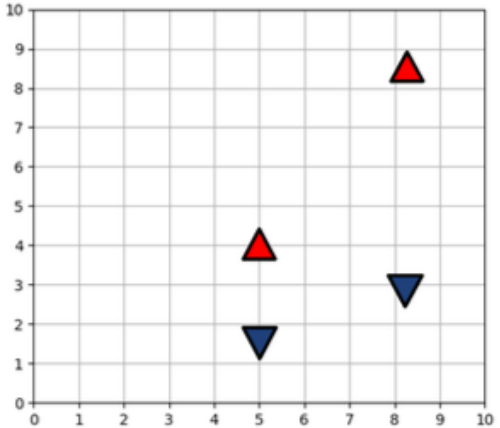


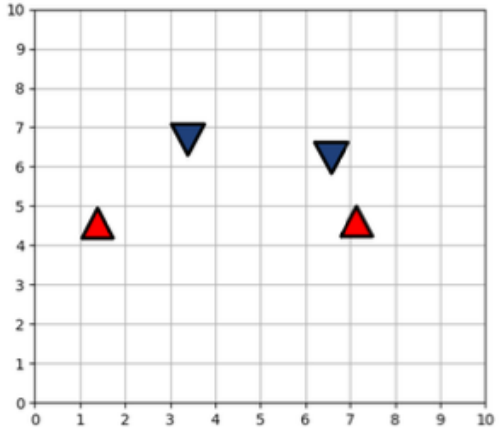




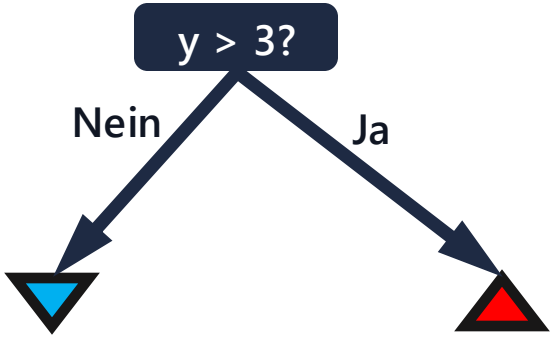
Übung







Das Modell besteht aus allen Bäumen, die nur eine Frage über die y-Koordinate stellen. Z.B.:



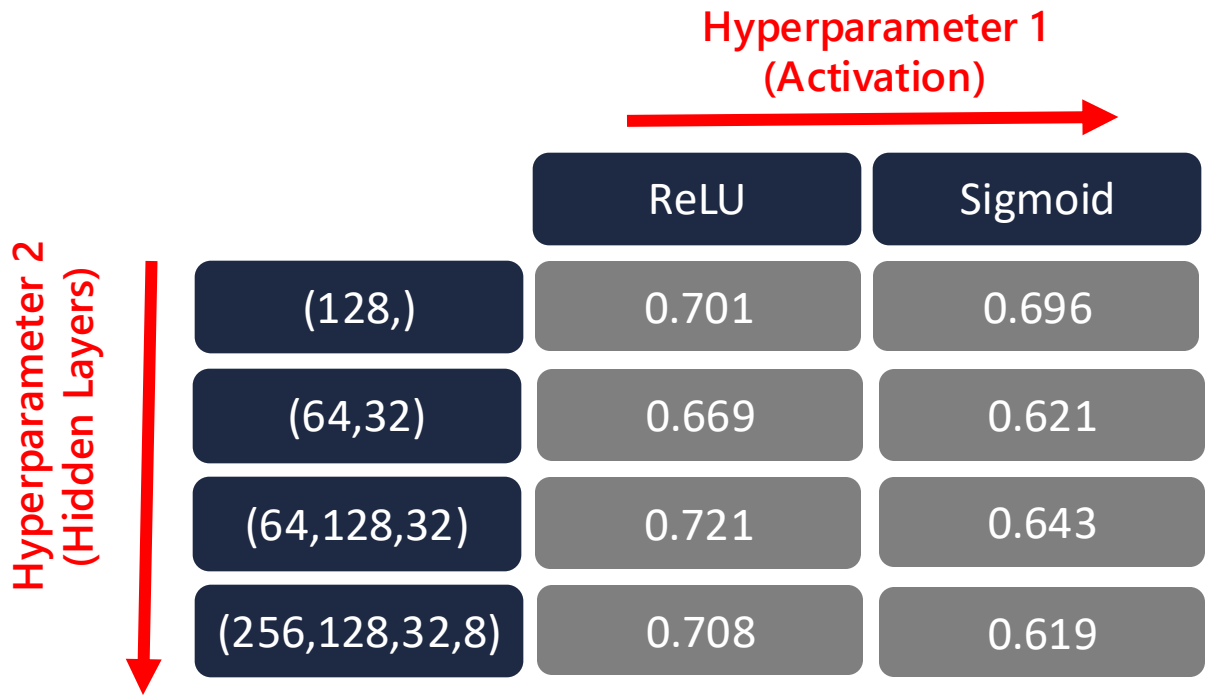
Berechne nun für jeden der drei Teile einen Schätzer für diesen Teil und bewerte seine Genauigkeit anhand der beiden Teile. (Cross-Validation)

2. Grid Search für Polynomregression

Grid Search

Vergleiche verschiedene Hyperparameter zur Feinabstimmung des Modells

- erstelle ein Raster verschiedener Hyperparameter
- trainiere und bewerte jede Kombination von Hyperparameterwerten im Raster



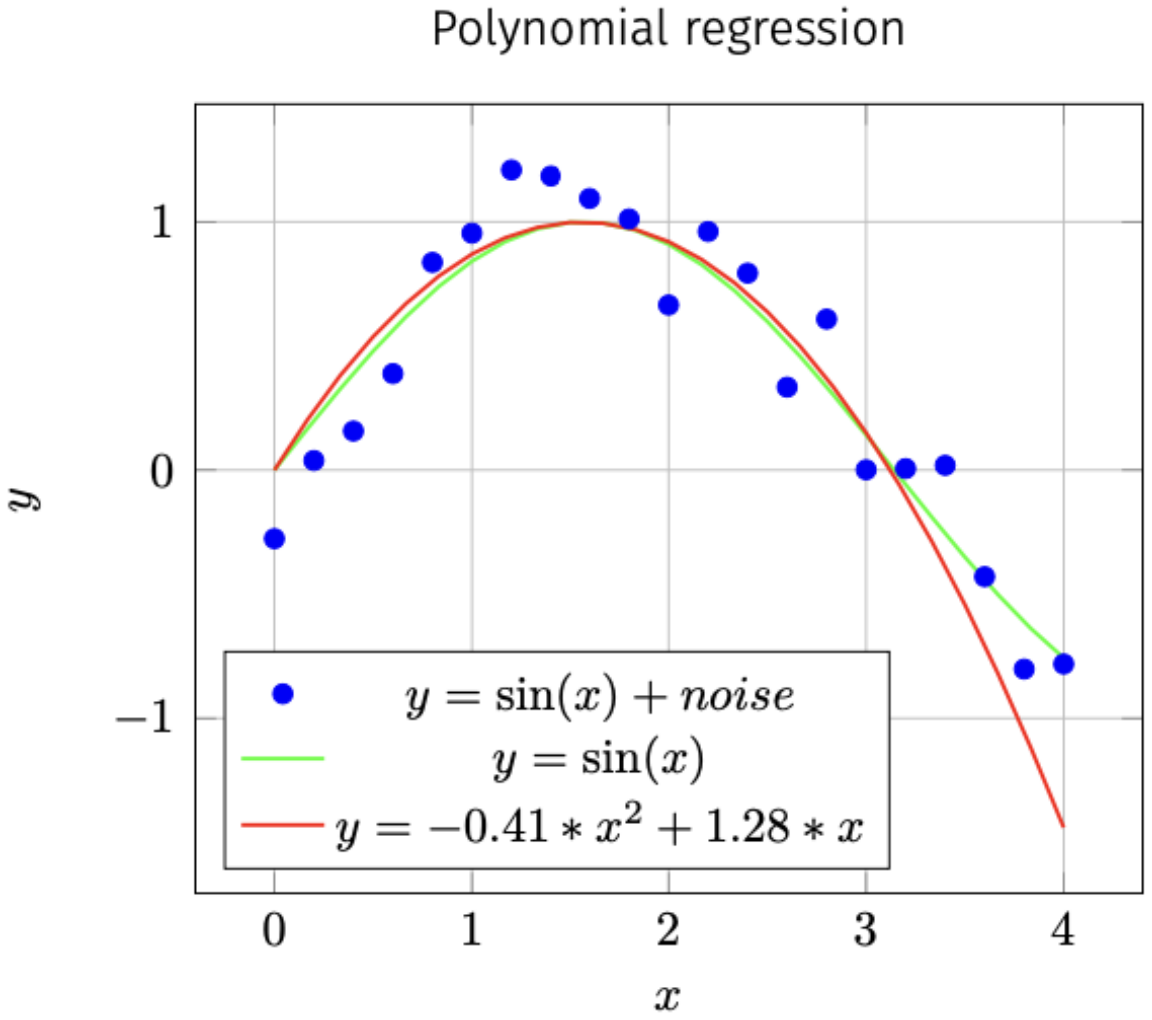
Grid Search & CV in Python:

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neural_network import MLPRegressor
#split data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
#create grid
param_grid = {'hidden_layer_sizes': [(128,),(64,32),(64,128,32)],
              'activation': ['relu','logistic']}
model = MLPRegressor() #select model
grid_search = GridSearchCV(model, param_grid, cv = 5) #define GridSearchCV
model

grid_search.fit(X_train,y_train) #perform grid search with cross-validation

y_pred = grid_search.predict(X_test) #predict using best parameters
```

Übung

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

- Wir möchten ein Polynom f finden, so dass $f(xi)$ 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: 1. Quiz

- Wie kann es zu Overfitting bei diesem Datensatz kommen?

A. Ein Polynom kann durch alle Punkte hindurch verlaufen, ohne das korrekte zugrundeliegende Polynom zu lernen.

B. Weil das Modell überhaupt keine nicht-linearen Terme besitzt und somit jeden Ausreisser exakt ausnutzt.

C. Polynom hat Parameter Overload und wird nun hyperaktiv.

Übung: 2. Quiz

- Welchen Hyperparameter können wir steuern, um Overfitting zu vermeiden?

A. Grösse vom Datensatz

B. Grad des Polynoms

C. Liniendicke des Plots

Übung

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

Übung

Implementiere nun in Code Expert ein Python-Programm, das Grid-Search und Cross-Validation verwendet, um Polynome anzupassen.

3. Neurale Netze

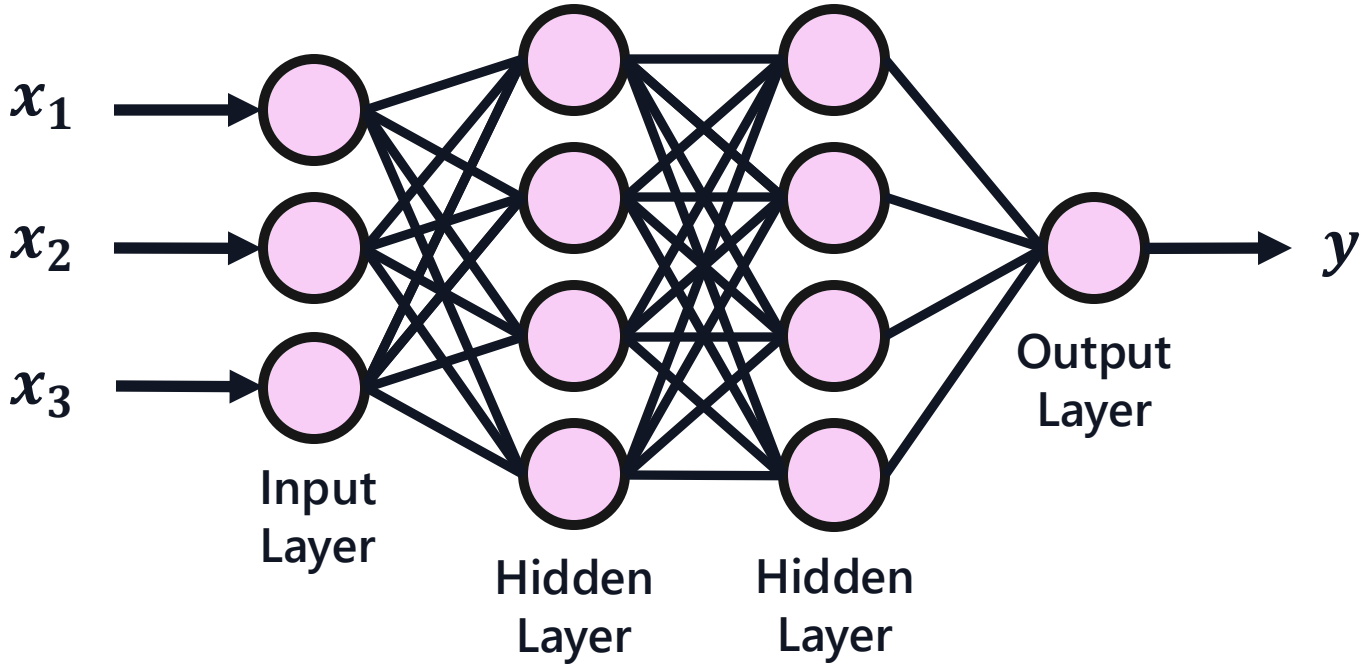
The Ultimate ML Algorithm

Neuronale Netze

- 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 Neurales Netz ist:

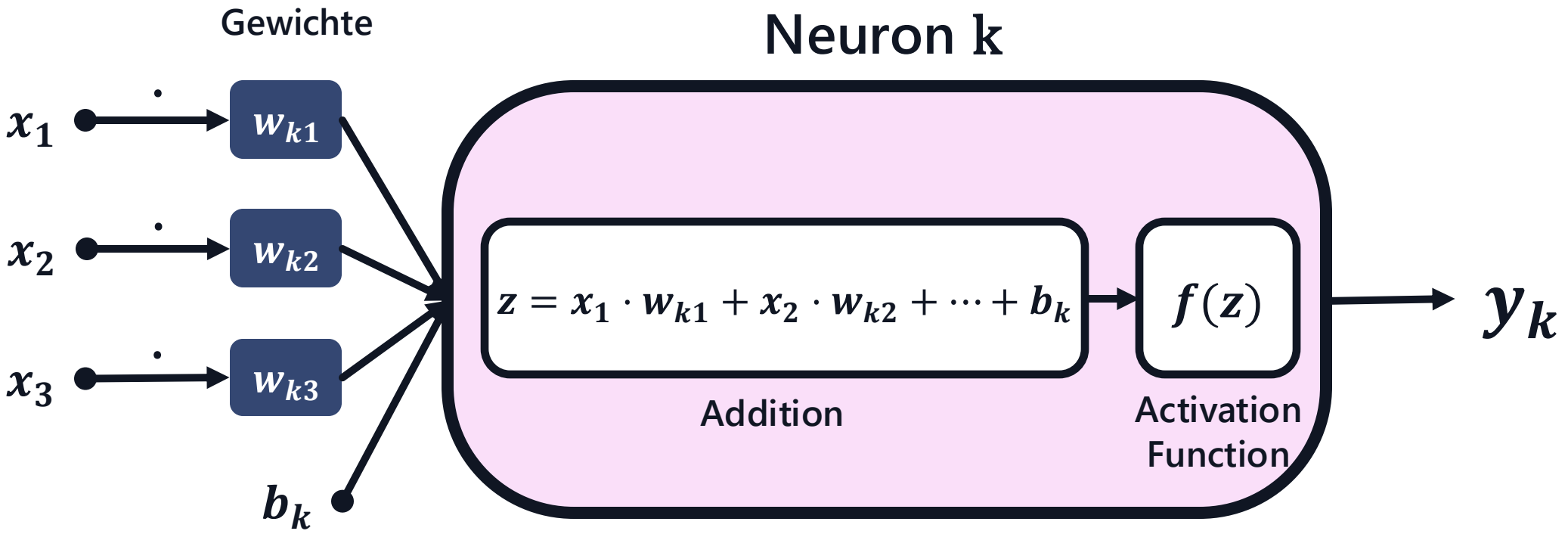
- Schichten verbundener Neuronen, bei denen die Ausgabe des Neurons durch eine Aktivierungsfunktion „nicht-linear“ gemacht wird



Ein Neuron ist:

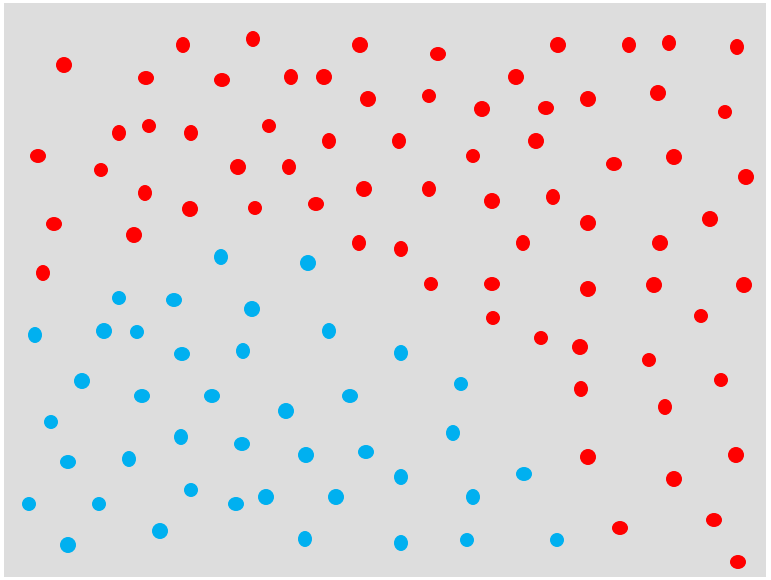
Mathematische Funktion, die einen Input nimmt und einen Output berechnet

$$y_k = f(x_1 \cdot w_{k1} + x_1 \cdot w_{k2} + \cdots + b_k)$$

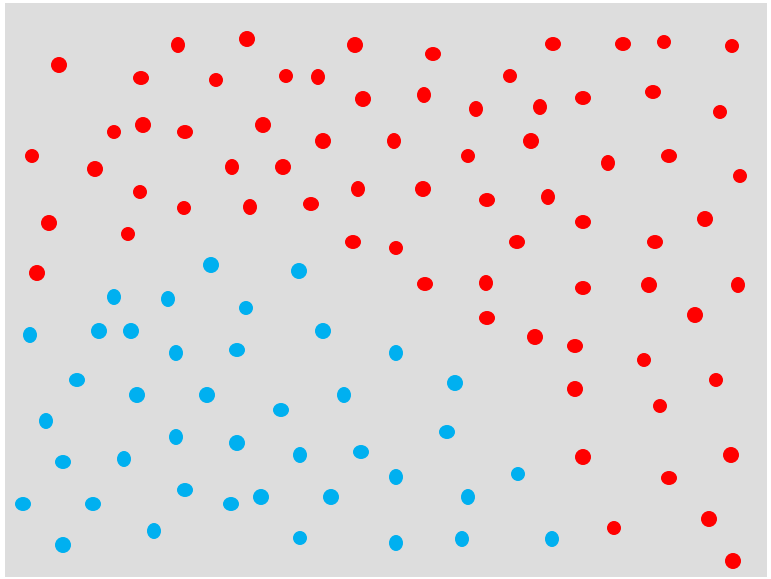


Eine Activation Function ist:

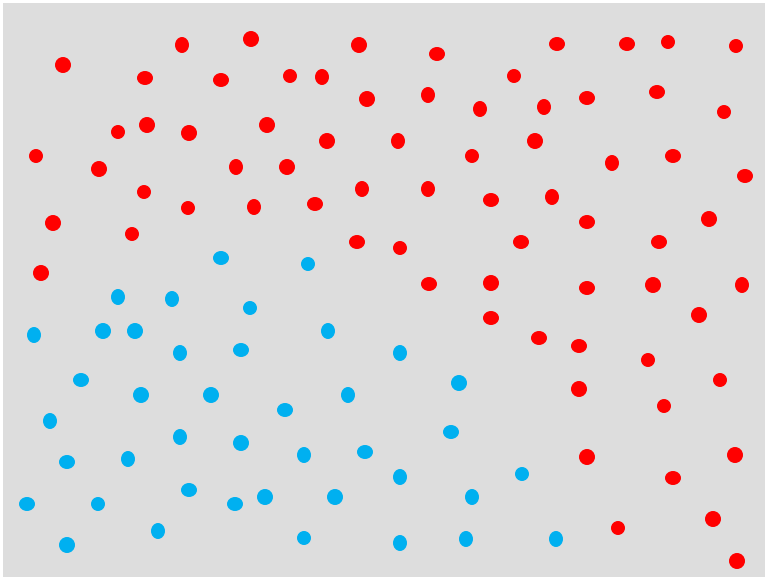
Mathematische Funktion, die Nichtlinearität einführt:



no activation function
(linear)



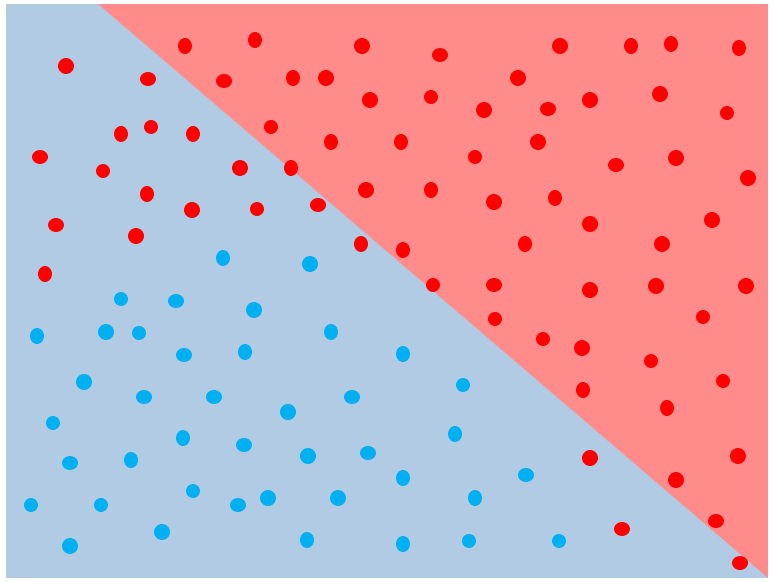
step activation function
(non-linear)



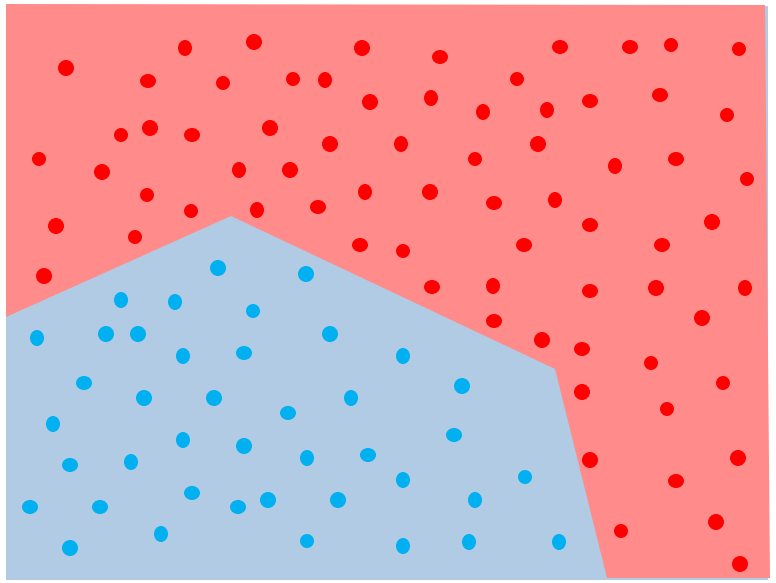
sigmoid/logistic
activation function
(non-linear)

Eine Activation Function ist:

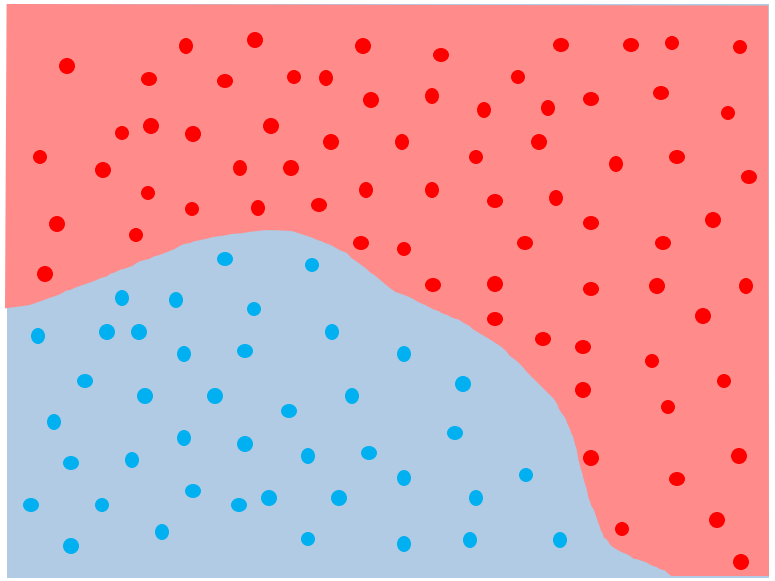
Mathematische Funktion, die Nichtlinearität einführt:



no activation function
(linear)



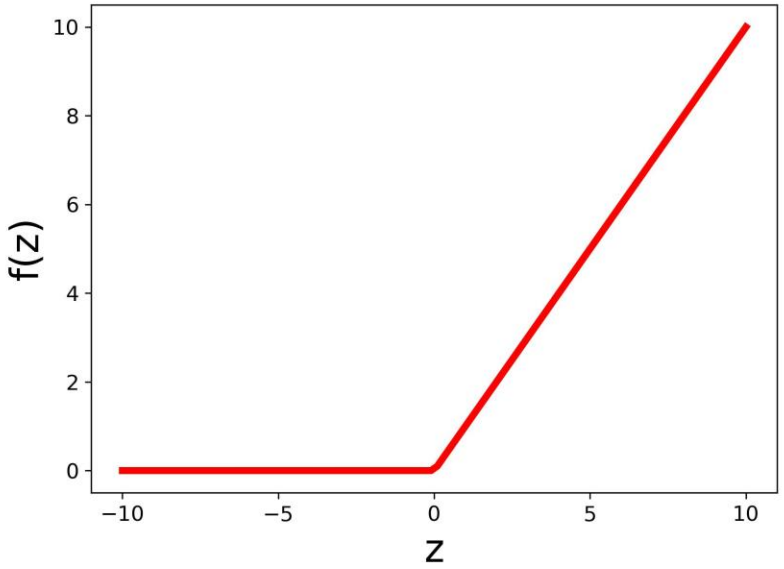
step activation function
(non-linear)



sigmoid/logistic
activation function
(non-linear)

ReLU vs Sigmoid

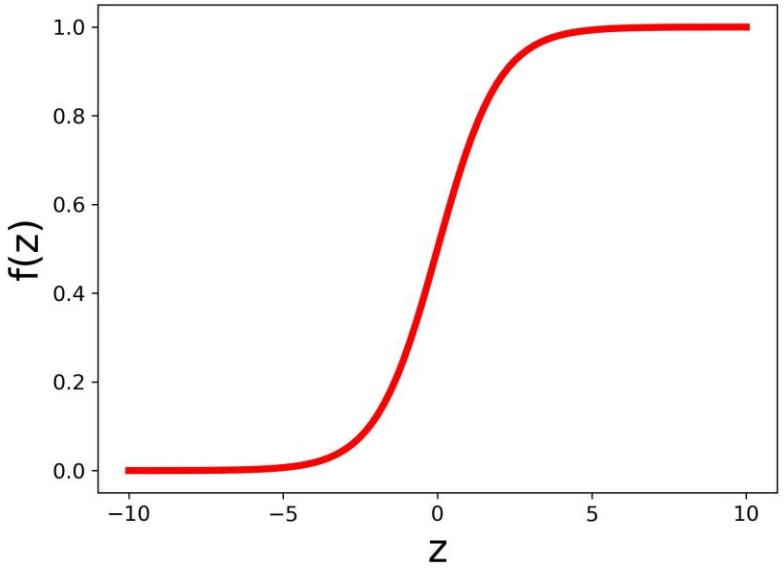
ReLU



$$f(z) = max(0, z)$$

- Gut für deep learning (alle layers)
- recheneffizient

Sigmoid (Logistic)

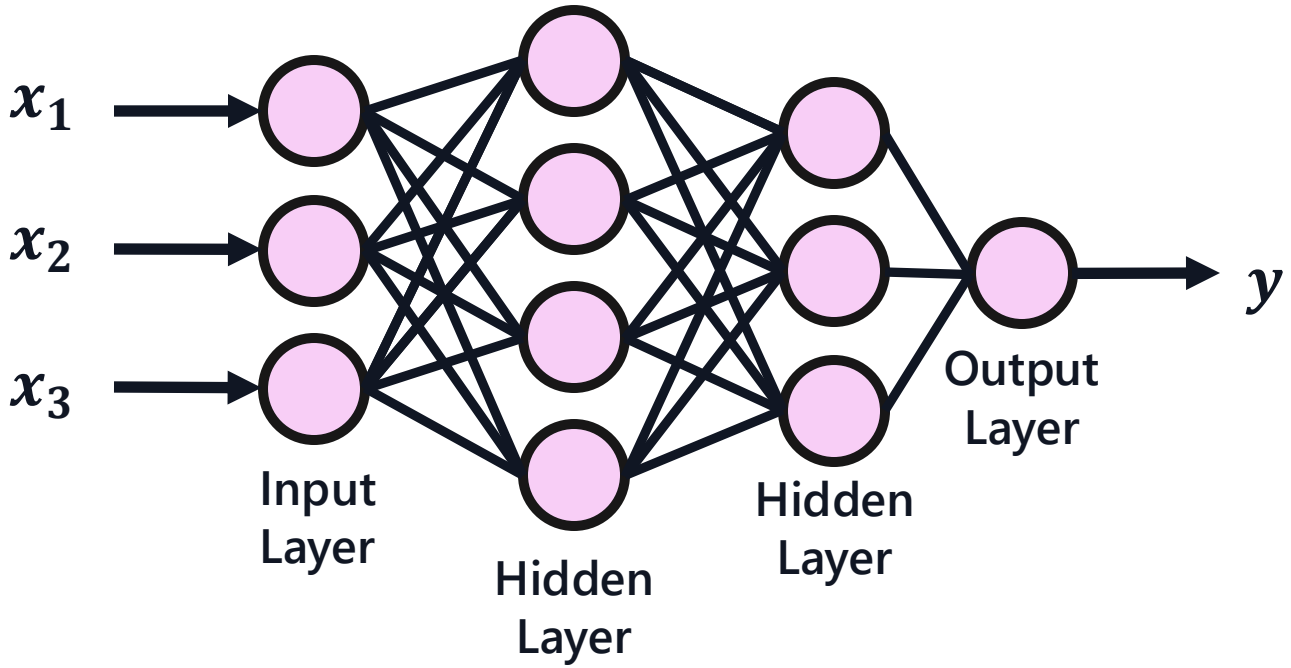


$$f(z) = \frac{1}{1 + e^{-z}}$$

- Gut für Klassifikation (letzter layer)

Hidden Layers

- Mehr layers, mehr Komplexität-> kann komplexere Muster besser fitten
- Risk von overfitting



In Python:

```
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier

#split data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state = 42)

#select model and perform training
mlp = MLPClassifier(hidden_layer_sizes=(32,32,16,16,8), max_iter=5000,
random_state = 42)
mlp.fit(X_train, y_train)

y_pred = mlp.predict(X_test) #predict using best parameters
```

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).

Übung: 1.Quiz

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

A. Weil sie ähnlich wie das menschliche Gehirn funktionieren

B. Neuronale Netze können komplexere Flächen im euklidischen Raum beschreiben.

C. Weil sie nicht linear sind

Übung: 2. Quiz

- Warum übertreffen tiefe neuronale Netze (Netze mit mehreren Schichten) flache neuronale Netze?

A. Weil tiefe neuronale Netze weniger Parameter haben und dadurch besser generalisieren.

B. Jede Schicht lernt Muster zu erkennen, die auf den vorherigen Schichten erkannten Muster basieren. → Tiefere Schichten lernen kompliziertere Muster als erste die Schichten

C. Neuronale Netze sind Scam und wurden von NVIDIA erfunden, um Geld zu machen.

Übung: 3. Quiz

- Was ist der Unterschied zwischen ReLU und Sigmoid?

A. ReLU: Linear, Sigmoid: Nicht Linear

B. ReLU: Klassifikation, Sigmoid: Regression

C. ReLU: Regression, Sigmoid: Klassifikation

D. Es gibt keine Unterschiede

Übung 4. Quiz

- Welche Python-Klassen repräsentieren Modelle für neuronale Netze in Scikit-learn?

A. from sklearn.neural_network import MLPClassifier

B. from sklearn.neural_network import CNNClassifier

C. from sklearn.tree import RandomForestClassifier

D. from sklearn.neural_network import MLPRegressor

Übung

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

4. Hausaufgaben

Exercise 11: Intro ML II

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

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

Abgabedatum: Montag 26.05.2025, 20:00 MEZ

NO HARDCODING

Fragen?

Feedback?

Zu schnell? Zu langsam? Weniger Theorie, mehr Aufgaben?
Dankbar für Feedback am besten mir direkt sagen oder Mail schreiben

Credits

Die Slide(-templates) stammen ursprünglich von Julian Lotzer und Daniel Steinhauser, vielen Dank!

→Checkt ihre Websites ab für zusätzliches Material in Informatik I, Informatik II und Stochastik & Machine Learning.

- https://n.ethz.ch/~jlotzer/
- https://n.ethz.ch/~dsteinhauser/