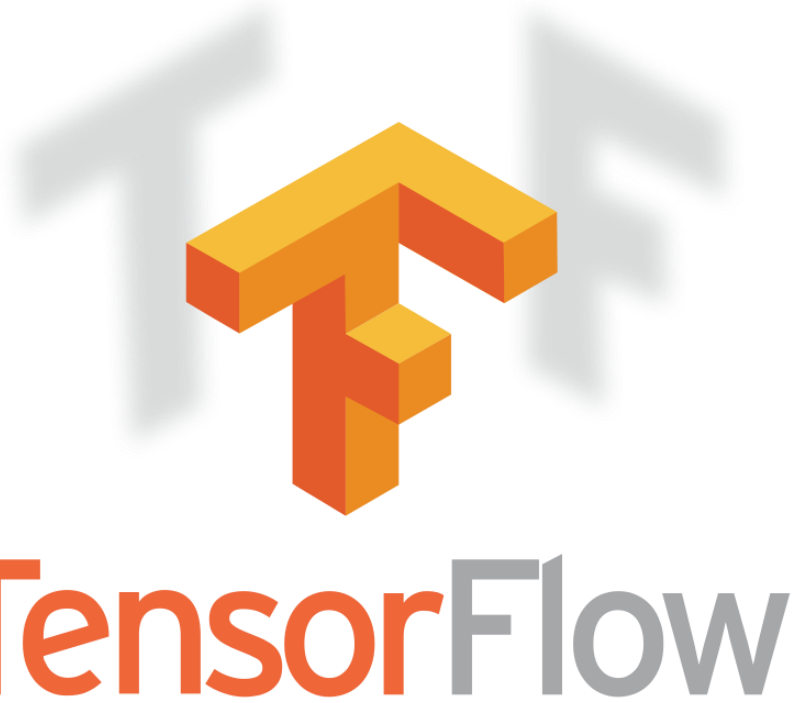
Tensorflow

MONA PIOTTER

Gliederung

- Allgemeines
- Maschinelles Lernen
  - Überwachtes maschinelles Lernen
    - Regression maschine learning systems
    - Classification maschine learning systems
  - Unüberwachtes maschinelles Lernen
- Programmieren mit Tensorflow
- Programmbeispiele



Allgemeines

- Freie Open-Source-Software Bibliothek
- Entwickelt vom Google Brain Team
- Für maschinelles Lernen verwendet
- Anwendung in Spracherkennung, Gmail, Google Foto, Google Suche und Google Maps
- Verwendet GPU und CPU
- Basiert auf C++ und Python
- Keras high level API für leichteren Einstieg

Maschinelles Lernen

- *“[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.” – Arthur Samuel*
- Für manche Fragestellungen zu hoher Programmieraufwand ohne maschinelles Lernen
- Möglich Probleme zu lösen, welche nicht mehr numerisch lösbar sind

- Zwei Kategorien:
  - Überwachtes maschinelles Lernen (Supervised Maschine Learning)

    → Mit vorhandene Trainingsbeispiele lernen um an neuen Daten Schlussfolgerungen zu ziehen

  - Unüberwachtes maschinelles Lernen (Unsupervised Maschine Learning)

    → In Daten Muster und Beziehungen erkennen und Vorhersagen für die Zukunft machen

Überwachtes maschinelles Lernen

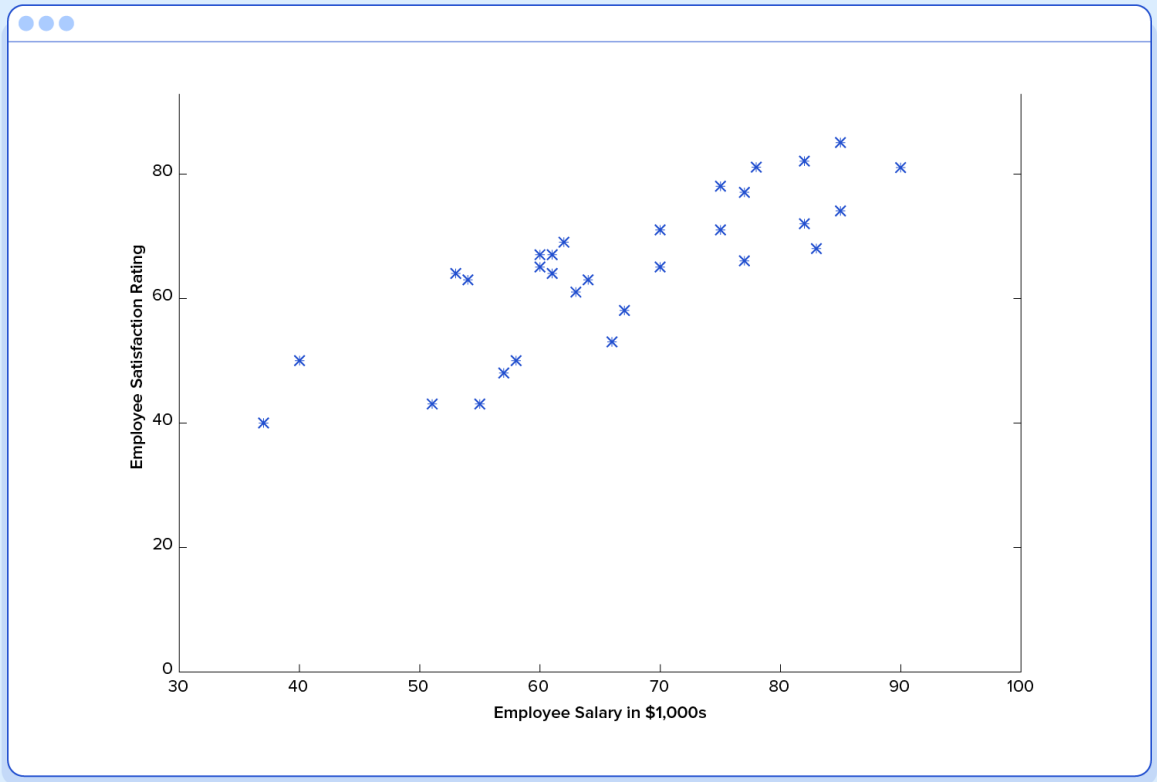
- Hypothesenfunktion h(x) erstellen
- Soll optimiert werden beim Lernen
- x normal multidimensionale Datenpunkte

- Beispiel: $h(x) = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_3^2 + \Theta_3 x_3\, x_4 + \Theta_4 x_1^3\, x_2^2 + \Theta_5 x_3^4\, x_4^2 x_2$

- Moderne Maschine Learning Programme benutzten Tausend bis Millionen an Dimensionen
- Klimaentwicklung oder Genexpression vorhersagen

Beispiel: $h(x) = \Theta_0 + \Theta_1 x$

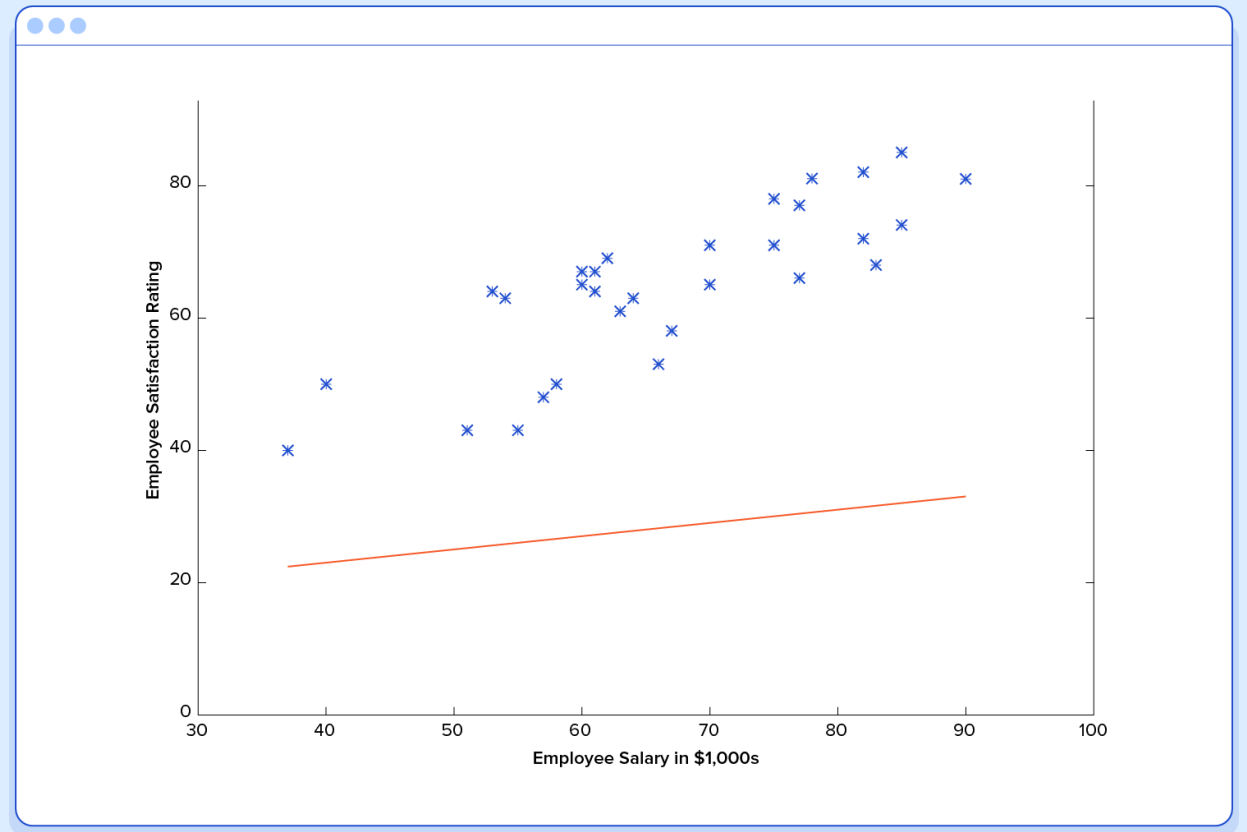
- Optimaler Wert für Θ gesucht
- Trainingsdaten für x mit bekanntem y Wert
- Korrektur von Θ an Hand von Differenz zwischen echten Wert (y) und unserer Vorhersage (h(x))
- Genug Beispiele nötig für echte gute Vorhersage
- Nie perfekte Vorhersage möglich, da auch in echter Welt keine eindeutigen Werte vorhanden
- Erstellen Vermutung, die gut genug ist um nützlich zu sein

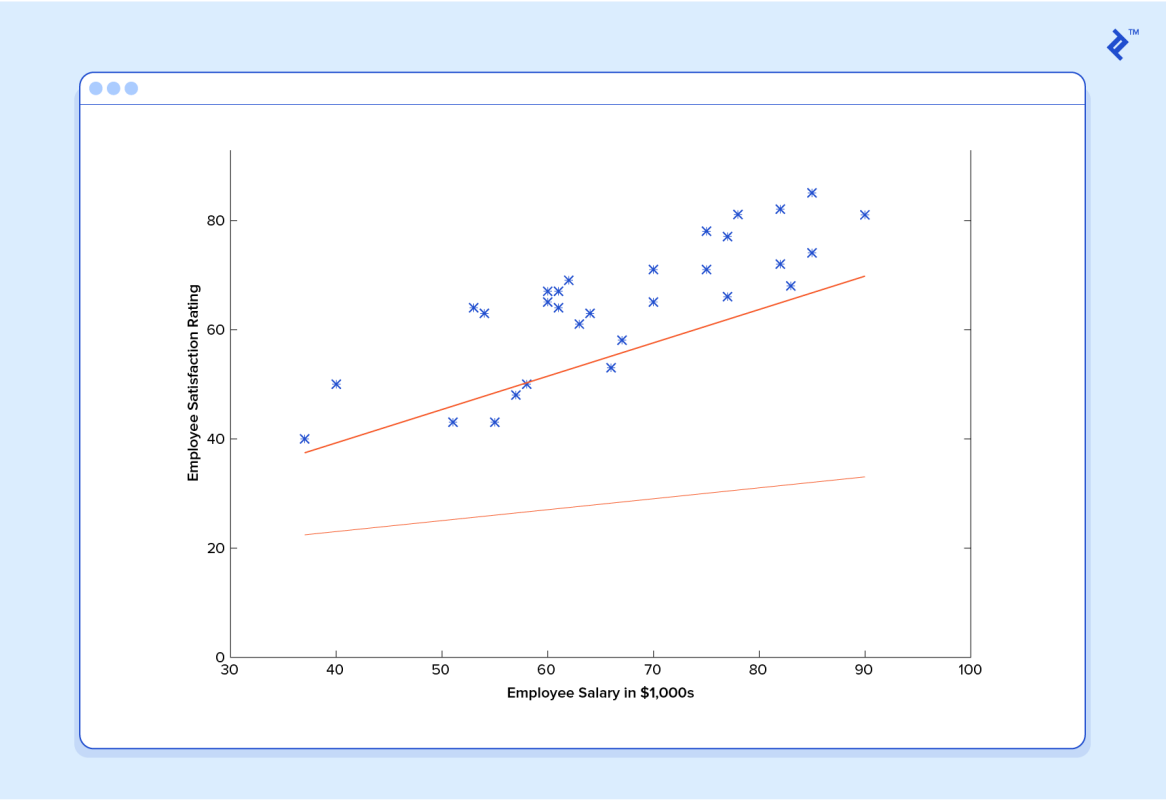


- Anfangswert setzten

  $h(x) = 12 + 0.2x$

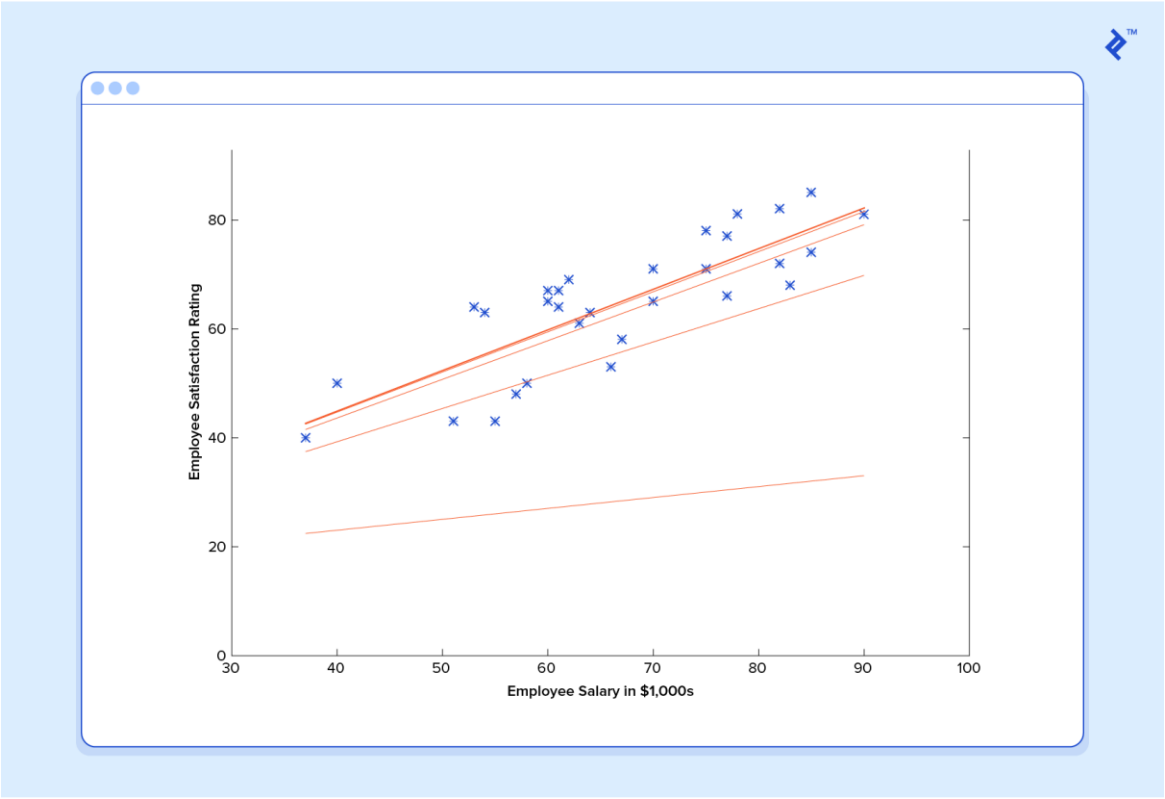
- Trainigsdaten dem Programm geben





Nach einem Durchgang durch die Trainigsdaten

$h(x) = 13.12 + 0.61x$



Nach 1500 Durchgängen durch die Trainigsdaten

$h(x) = 15.54 + 0{,}75x$

Abweichung berechnen

- Um Θ zu verbessern wird eine Kostenfunktion (loss function) erstellt
- $J(\Theta_0, \Theta_1)$ gibt an wie falsch h(x) mit aktuellen Werten von Θ ist
- Richtige Wahl der Kostenfunktion wichtig
- Für unser Problem: Methode der kleinsten Quadrate

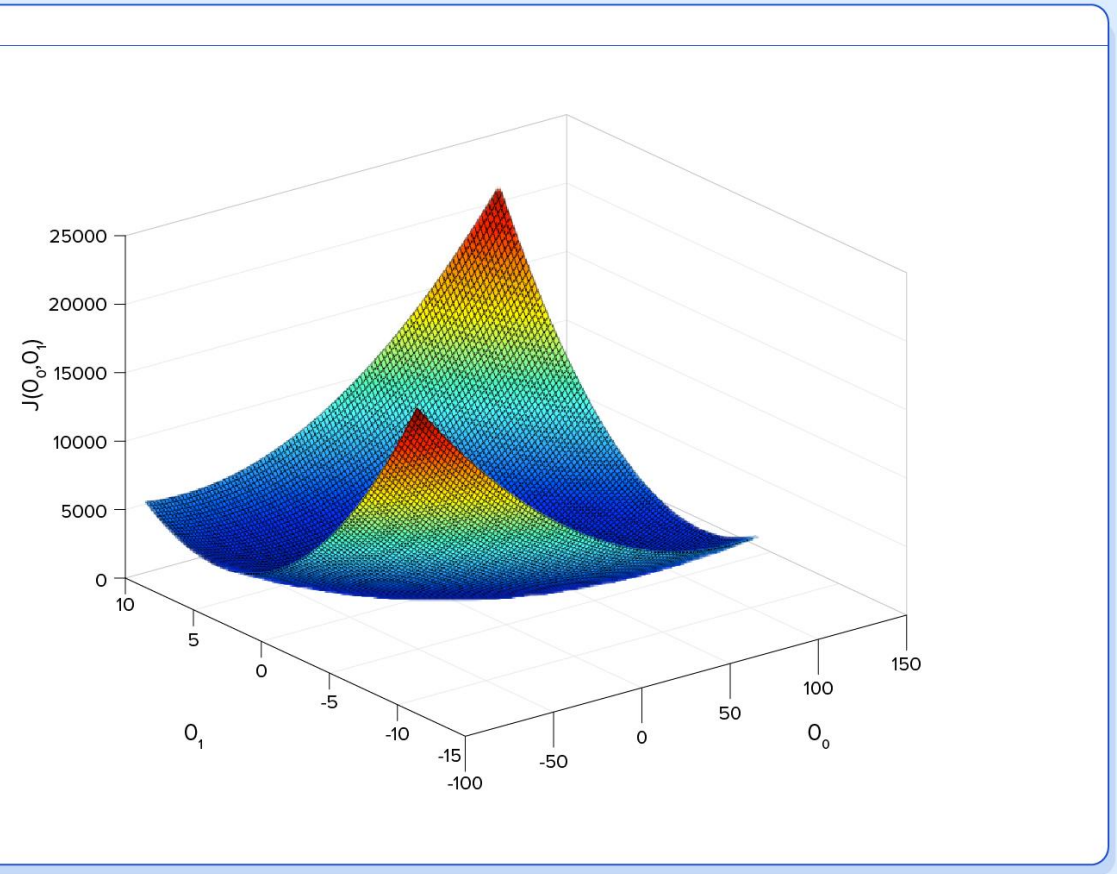
$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_{t,i}) - y)^2$$

- Kostenfunktion berechnet durchschnittliche Strafe für alle Trainigsdaten

- Wollen kleinstes $J(\Theta_0, \Theta_1)$

→ Gradientenverfahren

- Θ so varrieren bis minimales $J(\Theta_0, \Theta_1)$ erreicht

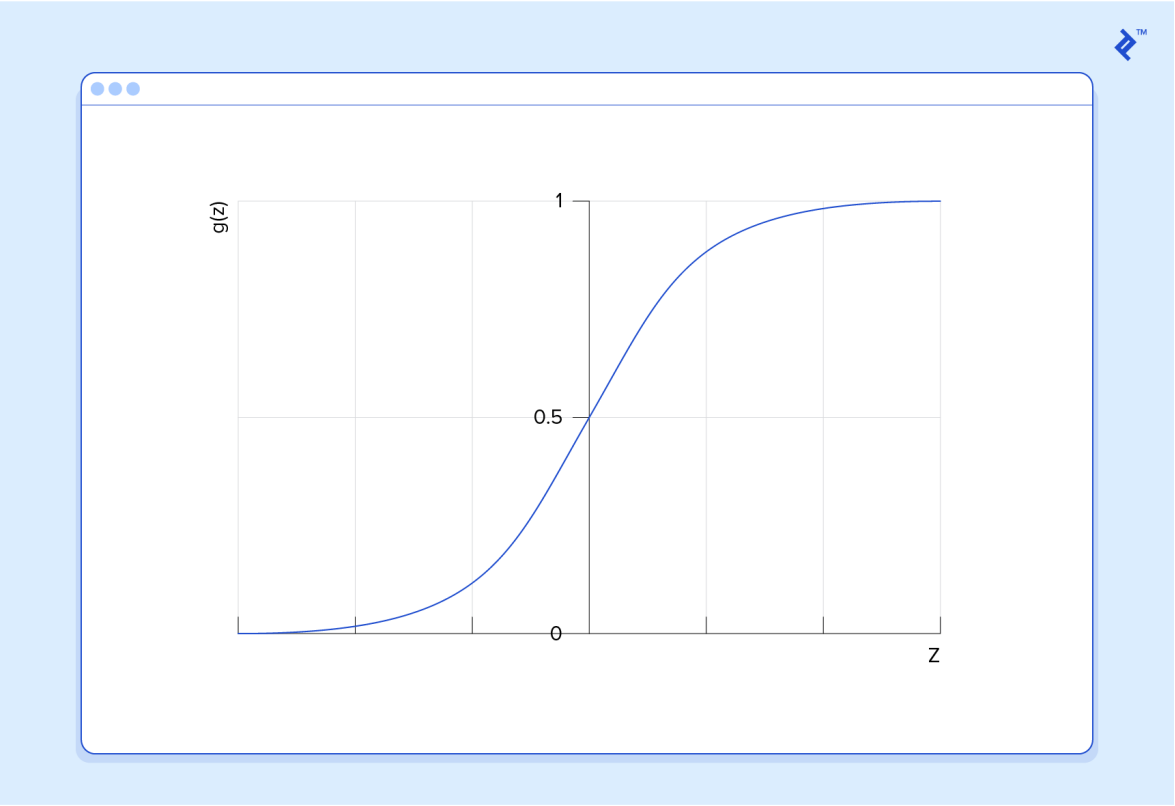


Weitere Unterteilung

- Beaufsichtigtes maschinelles Lernen kann in zwei Kategorien unterteilt werden
  - Regression machine learning systems → der zu bestimmende Wert fällt in ein Kontinuum
  - Classification machine learning systems → ja oder nein Fragen

- Prinzipielles Vorgehen gleich
- Vorhersage - und Kostenfunktion müssen angepasst werden
- Vorheriges Beispiel war ein Regressions Problem

- Beispiel wo 1 für gut steht und 0 für schlecht
- Wenn 0 vorhersagt wird statt 1 → großer Beitrag zur Kostenfunktion
- Wenn 0,6 vorhersagt wird statt 1 → geringerer Beitrag zur Kostenfunktion
- Sigmoid-Funktion bietet sich an
- Transformiert gegebenen Wert zwischen 0 und 1
- Noch Logarithmus der Sigmoid-Funktion verwenden
- Gradientenverfahren genau gleich

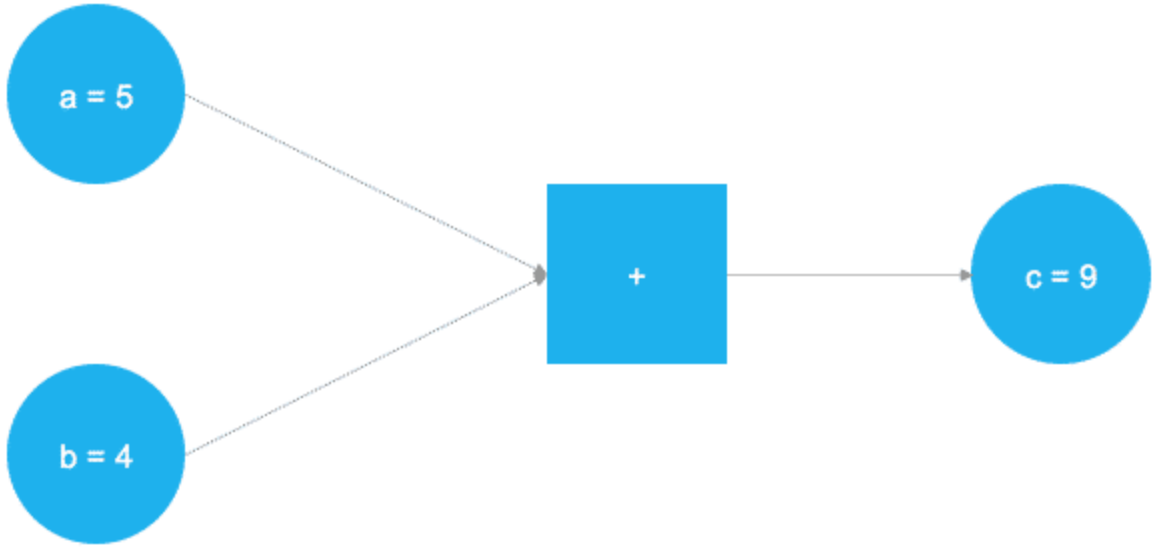


Unsupervised maschine learning

- Um Beziehungen in Daten zu finden
- Keine Trainingsdaten gegeben
- Verwendet z.B. zum Finden von engen Freundesgruppen in sozialen Netzwerken
- Andere Algorithmen werden verwendet

Programmieren mit Tensorflow

- Tensoren beschreibt Vektor oder Matrix
- Tensoren „fließen“ im Graph
- Operationen werden in Diagrammen dargestellt
- Knoten repräsentieren Tensorflowdaten und mathematische Operationen



- Zu Beginn Tensorflow importieren

  Import Tensorflow as tf

- Um Berechnungen im Graphen durchzuführen muss eine Session gestartet werden

  sess = tf.Session()

- Beim Definieren der Variablen muss Datentyp bekannt sein (C++ basiert)

  x = tf.constant(3 , dtype = tf.int8)

  Weitere Typen: tf.int16, tf.int32, tf.float16, tf.float32, tf.float64, tf.string, tf.bool, …

- Befehle aus Python funktionieren gleich mit Tensorflow (matmul, add, linalg, …)

- Bestimmung des Ergebniss durch ausführen des Graphen an Stelle z

  sess.run(z)

- Ausführen auf der CPU und GPU möglich
  - Werden als string dargestellt („/cpu:0“, „/device:GPU:0“, /device:GPU:1“, …)
  - tf.Session(config=tf.ConfigProto(log_device_placement = True)) → gibt an wo welcher Prozess stattfindet
  - with tf.device(„ “) → festlegen wo welcher stafffinden soll

Beispiel zum selber programmieren

- Addieren Sie zwei Zahlen miteinander mit Hilfe von Tensorflow
- Berechnen Sie mit Hilfe von Tensorflow eine Matrixmultiplikation der Matrizen

$$M_1 = \begin{pmatrix} 5 & 7 & 9 & 1 \\ 3 & 1 & 6 & 4 \\ 8 & 4 & 2 & 7 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 5 & 2 & 1 \\ 9 & 1 & 3 \\ 1 & 7 & 4 \\ 8 & 1 & 2 \end{pmatrix}$$

- Lassen Sie sich ausgeben ob auf der CPU oder GPU gearbeitet wird
- Legen Sie fest, dass auf der CPU die Matrizen M_1 und M_2 berechnet werden und vergleichen Sie den zeitlichen Unterschied (Hinweis: import time ist möglich für die Zeiterfassung)

Neuronales Netzwerk programmieren

- Benutzen Keras, da leichterer Einstieg
- Modele werden durch Layers gebildet

  Model = tf.keras.Sequential() → Stapel an Schichten

  Model.add(layers.Dense(64, activation = tf.nn.relu) → besser als Sigmoid

  Model.add(layers.Dense(64, activation = tf.sigmoid)

  Model.add(layers.Dense(64, activation = 'softmax') → gibt einen Array aus mit Wahrscheinlichkeiten

- Model kompiliren
  - Kostenfunktion definieren (z.B. mse (Mean Square Error), categorical_crossentropy, …)
  - Metrik - überwacht das Training (z.B. mae (mean absolut error, accuracy, …)
  - Optimizer – wie Model durch die Kostenfunktion verbessert wird (z.B. AdamOptimizer, GradientDescentOptimzier, …)

```
Model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

- Model trainieren:
  - Trainigsdaten übergeben
  - Epochen: Anzahl der Iterationen über alle Daten
  - batch_size: Daten werden in kleinere Teile gespalten

  ```
  model.fit(data, labels, epochs=5, batch_size = 32)
  ```

- Mit Testdaten unser trainiertes Model überprüfen:

  ```
  test_loss, test_acc = model.evaluate(testdaten, testlabels)
  ```

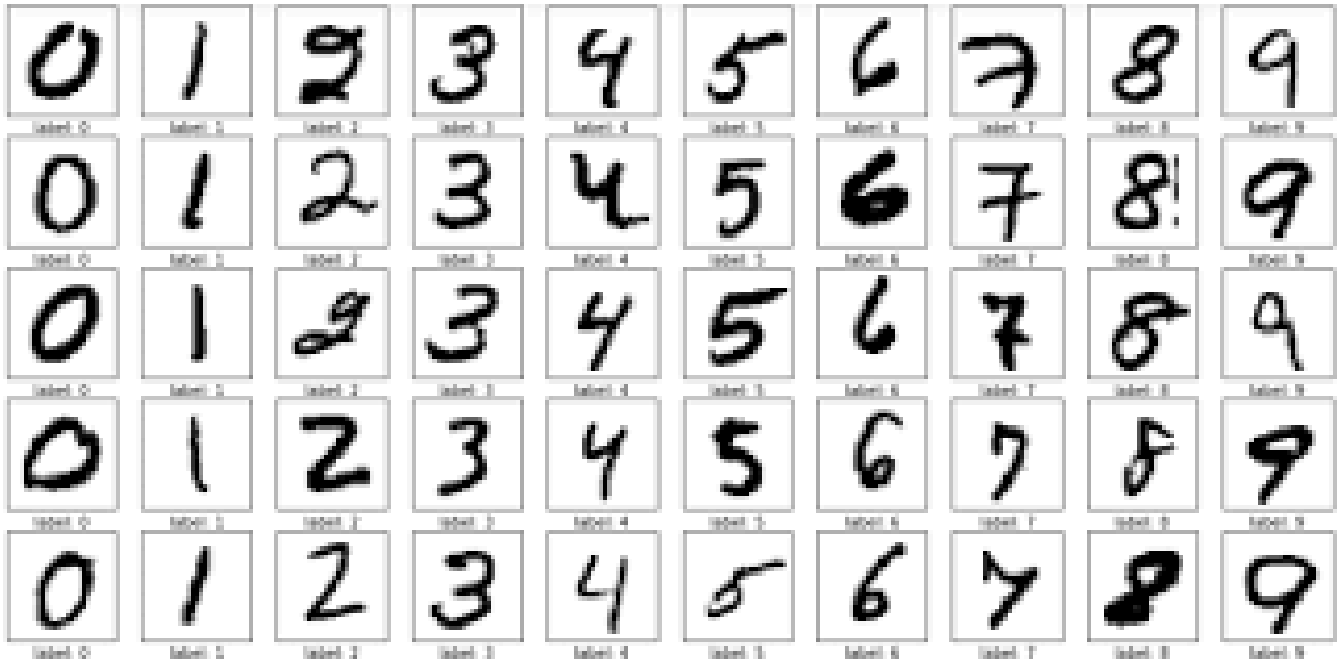
  Sollte die Accuracy der Testdaten niedriger sein als die der Trainigsdaten → Overfitting (Model läuft schlechter auf neuen Daten, zu sehr an Trainigsdaten angepasst)

- Vorhersagen machen:

  ```
  Predictions = model.predict(testdaten)
  Predictions[0]
  ```

Beispiel zum selber programmieren 2

Nutzen sie die gegebene Vorlage um selber ein Model zu erstellen. Verwenden Sie dazu entweder das Beispiel der Kleidererkennung oder Zahlenerkennung. Beide funktionieren nach dem gleichen Prinzip.



Zusätzliche Infos zur Aufgabe

- Daten von MNIST
- 70 000 Bilder mit je 10 Kategorien (Kleidung oder Zahlen)
- 60 000 Trainigsdaten, 10 000 Testdaten
- 28 x 28 pixel große Auflösung
- Kleiderdaten müssen präprozessiert werden

Quellen

- https://www.toptal.com/machine-learning/machine-learning-theory-an-introductory-primer
- https://de.wikipedia.org/wiki/TensorFlow
- https://www.tensorflow.org/guide/using_gpu
- https://www.tensorflow.org/tutorials/eager/eager_basics
- https://www.statworx.com/de/blog/data-science/einfuehrung-tensorflow/
- http://www.ashukumar27.io/mnist-cnn-keras/
- https://www.youtube.com/watch?v=2FmcHiLCwTU
- https://www.tensorflow.org/guide/keras