Würgeschlange 3 - Lesson 12

Tobias Maschek, Viktor Reusch
https://github.com/jemx/wise1920-python

mit Materialien von Felix Döring, Felix Wittwer https://github.com/fsr/python-lessons

28. Januar 2020

Python-Kurs

Gliederung

1. Vorbereitung
2. Künstlich neuronale Netze
3. Iris
4. MNIST
5. CNN

Vorbereitung

Das hier ist wichtig!

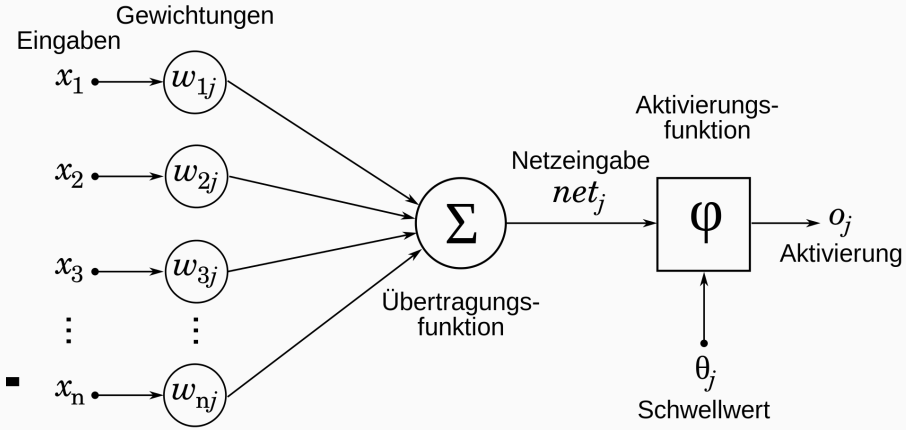
- *Python 3.4* bis *3.7*
- *venv*
  - erstellen *python3 -m venv /path/to/new/virtual/environment*
  - benutzen Windows: *.\env\Scripts\activate*; Linux: *source env/bin/activate*
  - Pakete installieren: *pip install PACKAGENAMEGOESHERE*
- PIP Pakete *tensorflow* und *tensorflow-datasets*

Künstlich neuronale Netze

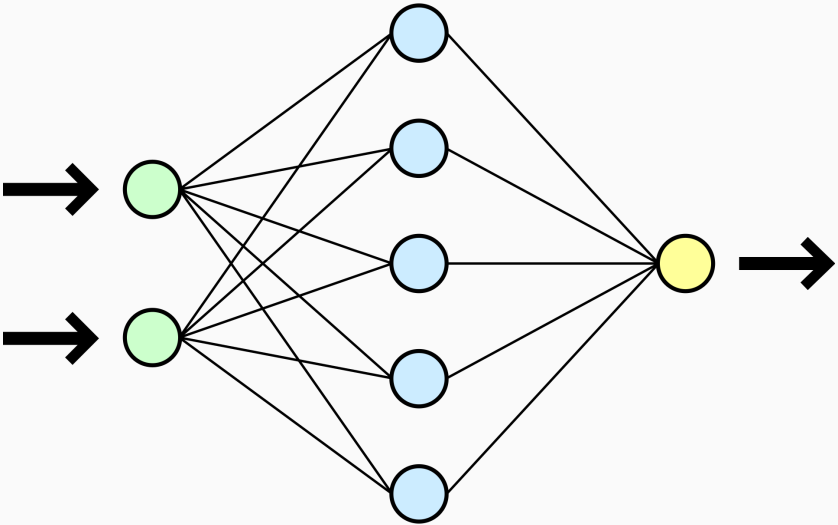
KNN - Theorie

- Neuronale Netze ermöglichen das selbständige Denken, Lernen und Optimieren von Prozessabläufen.
- Bestehen aus Neuronen in unterschiedlich vielen Layern

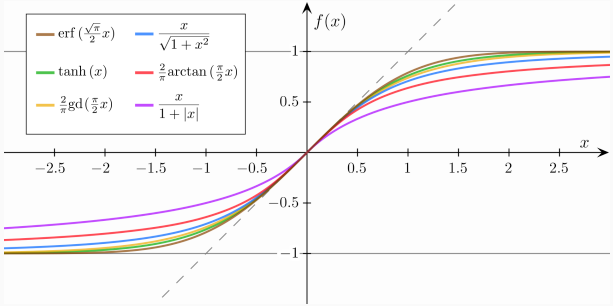
- Neuronen sind wesentlicher Bestandteil Neuronaler Netze



Neuronale Netzwerke



Aktivierungsfunktion



Some sigmoid functions compared von Georg-Johann unter CC BY-SA 3.0

- Sigmoid
- Tanh
- Step function
- ReLU/Leaky ReLU
- ...

Iris

Iris-Datensatz

- Klassiker der Datenanalyse
- Besteht aus Messdaten echter Pflanzen der Gattung *Schwertlilien* (*Iris*).
- Ziel ist es, anhand dieser Messdaten die einzelnen Arten unterscheiden zu können
- Maschinelles Lernen dafür nicht notwendig: Wir nutzen es dennoch.

sepal length	**sepal width**	**petal length**	**petal width**	**class**
5.1	3.5	1.4	0.2	Iris-setosa
6.9	3.2	5.7	2.3	Iris-virginica

Webseite

Iris — Beispiel

```
import tensorflow as tf
import tensorflow_datasets as tfds

iris = tfds.load("iris", split="train", as_supervised=True).shuffle(4096)
           .batch(10)

iris_validation = iris.take(5)
iris_train = iris.skip(5)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(3, activation="softmax")
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(iris_train, epochs=70, validation_data=iris_validation,
          validation_steps=5)
```

MNIST

MNIST-Datensatz

- Klassiker der Bildanalyse
- Große Sammlung Schwarz-Weiß-Bilder von handgeschriebenen Ziffern
- Bilder sind nur 28x28 Pixel groß
- Können durch Abflachen von simplen neuronalen Netzen gehändelt werden

Mnist Examples von Josef Steppan unter CC BY-SA 4.0

Webseite

MNIST — Beispiel

```
import tensorflow as tf
import tensorflow_datasets as tfds

mnist = tfds.load("mnist", as_supervised=True)
mnist_train = mnist["train"].take(4096).shuffle(4096).batch(128).repeat()
mnist_validation = mnist["test"].take(128).batch(32)

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(mnist_train, epochs=40, steps_per_epoch=32,
          validation_data=mnist_validation, validation_steps=4)
```

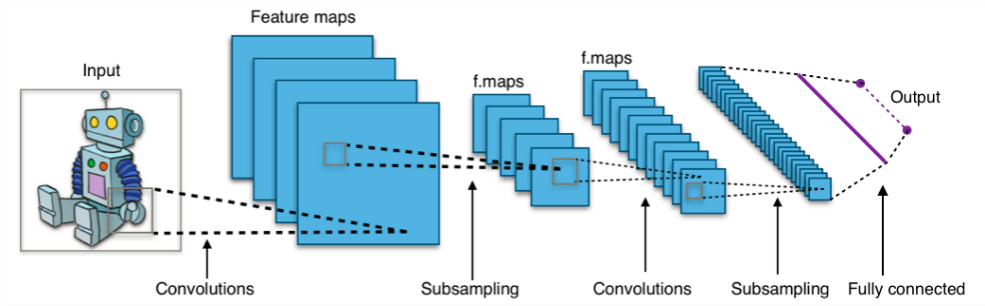
Aufgabe 12-1

- Versucht die Beispiele bei euch zum Laufen zu bekommen.
- Ändert Parameter wie die Anzahl der Neuronen im Layer
- Fügt mehr hidden layers hinzu und vergleicht das Ergebnis.
- EXPERTEN: Messt die Ausführungszeit des Skriptes. Wann dauert das Training länger?

CNN

Convolutional Neural Network

- CNNs dienen der Bildanalyse
- In den ersten Schichten werden keine Neuronen sondern Bildfilter trainiert
- Durch Überspringen und Skalieren wird Rechenaufwand verkleinert
- Letze Layer sind normales Netzwerk auf stark verkleinertem, gefiltertem Bild
- Es existieren vortrainierte CNN-Schichten zum schnellen Trainieren



CNN — Beispiel

```python
import tensorflow as tf
import tensorflow_datasets as tfds

def transform_images(image, label):
    image = tf.image.resize(image, (64, 64))
    image = image / 255
    return image, label

catsdogs = tfds.load("cats_vs_dogs", split="train", as_supervised=True).map(
    transform_images)
catsdogs_validation = catsdogs.take(64).batch(64)
catsdogs_train = catsdogs.skip(64).batch(64).repeat()
```

CNN — Beispiel - Auf eigene Faust

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(128, (5, 5), strides=(2, 2), padding="same",
       activation="relu", input_shape=(64, 64, 3)),
    tf.keras.layers.MaxPooling2D((2, 2), (2, 2), padding="same"),
    tf.keras.layers.Conv2D(128, (3, 3), padding="same", activation="relu"),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(2, activation="softmax"),
])
model.summary()

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

model.fit(catsdogs_train, epochs=5, steps_per_epoch=32,
          validation_data=catsdogs_validation,validation_steps=1)
```

CNN — Beispiel - Vortrainiert

```
mnet = tf.keras.applications.mobilenet_v2.MobileNetV2(
          input_shape=(160, 160, 3), include_top=False, weights='imagenet'
        )
mnet.trainable = False
mnet.summary()
model = tf.keras.models.Sequential([
        mnet,
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dropout(0.4),
        tf.keras.layers.Dense(2, activation="softmax"),
])
model.summary()

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

model.fit(catsdogs_train, epochs=5, steps_per_epoch=32, validation_data=
    catsdogs_validation,validation_steps=1)
```