Methoden des Maschinellen Lernen

Die Grundidee des Maschinellen Lernen besteht darin einem Computer nicht wie üblich Schritt für Schritt zu sagen, was er tun soll, sondern viele Daten zusammenstellen und dem Computer selbst Muster und Regeln erkennen lässt. Der Computer bekommt so viele Beispiele. Zum Beispiel, wenn der Computer lernen soll, Katzen auf Bildern zu erkennen, zeigen wir ihm Tausende von Bildern, auf denen Katzen zu sehen sind, und auch Bilder ohne Katzen. Der Computer analysiert diese Bilder und sucht nach gemeinsamen Merkmalen, die typisch für Katzen sind, wie Formen, Farben und Texturen. Maschinelles Lernen bedeutet, dass Computer aus vielen Beispielen lernen, um Aufgaben besser zu erledigen, anstatt dass wir ihnen jede einzelne Regel vorschreiben.

Das Grundprinzip

ki_overview_v2.png

Supervised Learning (Überwachtes Lernen)

Idee	Der Computer lernt aus Beispielen, bei denen die richtigen Antworten bereits bekannt sind.
Funktion	Wir geben dem Computer viele Beispiele mit den richtigen Antworten. Zum Beispiel: Bilder von Hunden und Katzen, bei denen jedes Bild beschriftet ist (Hund oder Katze). Der Computer lernt, diese Beispiele zu analysieren und Regeln zu entwickeln, um neue, unbekannte Bilder richtig zu erkennen.
Beispiel	Wenn du dem Computer 100 Bilder von Hunden und Katzen zeigst und ihm sagst, welche Bilder Hunde und welche Katzen sind, kann er lernen, neue Bilder zu identifizieren.
Zusammenfassung	*Supervised Learning*: Lernen mit bekannten Antworten.

```
import tensorflow as tf
from tensorflow.keras import layers


# Daten laden
```

```
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Daten normalisieren
x_train, x_test = x_train / 255.0, x_test / 255.0

# Modell aufbauen
model = tf.keras.models.Sequential([
  layers.Flatten(input_shape=(28, 28)),  # Eingabeschicht, die die Bilder flach drückt
  layers.Dense(128, activation='relu'),  # Dicht verbundene Schicht mit 128 Neuronen
  layers.Dropout(0.2),                   # Dropout-Schicht zur Reduzierung von Overfitting
  layers.Dense(10, activation='softmax') # Ausgabeschicht mit 10 Neuronen für 10 Klassen
])

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

# Modell trainieren
model.fit(x_train, y_train, epochs=5, validation_split=0.1)

# Modell evaluieren
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test Accuracy: {test_acc:.4f}')
```

Unsupervised Learning (Unüberwachtes Lernen)

Idee	Der Computer bekommt Daten ohne die richtigen Antworten und muss selbst Muster oder Gruppen finden.
Funktion	Der Computer analysiert die Daten und sucht nach Gemeinsamkeiten oder Gruppen, ohne dass wir ihm sagen, was er suchen soll. Er findet beispielsweise heraus, dass bestimmte Bilder ähnlich sind und bildet daraus Gruppen.
Beispiel	Wenn du dem Computer viele Bilder gibst, ohne zu sagen, ob es Hunde oder Katzen sind, wird er versuchen, die Bilder in Gruppen zu sortieren, die sich ähneln, z.B. eine Gruppe mit Hunden und eine Gruppe mit Katzen.
Zusammenfassung	*Unsupervised Learning*: Finden von Mustern ohne bekannte Antworten.

```
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Daten laden
data = load_iris()
X = data.data

# Daten standardisieren
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# KMeans Modell erstellen
kmeans = KMeans(n_clusters=3, random_state=0)  # 3 Cluster, entspricht der Anzahl der Iris-Arten

# Modell auf Daten anwenden
kmeans.fit(X_scaled)

# Cluster-Zentren und Labels
centers = kmeans.cluster_centers_
labels = kmeans.labels_

# Ergebnisse visualisieren
plt.figure(figsize=(10, 6))
colors = ['red', 'green', 'blue']
for i in range(3):
    plt.scatter(X_scaled[labels == i, 0], X_scaled[labels == i, 1], color=colors[i], label=f'Cluster {i+1}')
plt.scatter(centers[:, 0], centers[:, 1], s=100, color='yellow', label='Centroids', edgecolors='black')
plt.title('KMeans Clustering of Iris Dataset')
plt.xlabel('Sepal Length (standardized)')
plt.ylabel('Sepal Width (standardized)')
plt.legend()
plt.show()
```

Reinforcement Learning (Bestärkendes Lernen)

Idee	Der Computer lernt durch Ausprobieren und Belohnung oder Bestrafung.
Funktion	Der Computer führt Aktionen aus und erhält Feedback, ob diese gut oder schlecht waren. Er versucht, Belohnungen zu maximieren und Bestrafungen zu minimieren, indem er aus seinen Erfahrungen lernt.
Beispiel	Stell dir ein Computerspiel vor. Der Computer versucht, das Spiel zu spielen und erhält Punkte (Belohnung) für gute Züge und verliert Punkte (Bestrafung) für schlechte Züge. Mit der Zeit lernt er, wie man das Spiel am besten spielt.
Zusammenfassung	*Reinforcement Learning*: Lernen durch Ausprobieren und Feedback.

```
import gym
import numpy as np

# Umgebung erstellen
env = gym.make('CartPole-v1')

# Q-Tabelle initialisieren
Q = np.zeros([env.observation_space.shape[0], env.action_space.n])

# Lernparameter
learning_rate = 0.8
discount_factor = 0.95
episodes = 1000

# Funktion zur Wahl der Aktion
def choose_action(state):
    action = np.argmax(Q[state, :] + np.random.randn(1, env.action_space.n) * (1. / (i + 1)))
    return action

# Training des Modells
for i in range(episodes):
    state = env.reset()
    done = False
    while not done:
        action = choose_action(state)
        new_state, reward, done, info = env.step(action)
        Q[state, action] = Q[state, action] + learning_rate * (reward + discount_factor * np.max(Q[new_state, :]) -
Q[state, action])
```

```
        state = new_state

# Modell evaluieren
rewards = []
for _ in range(100):
    state = env.reset()
    total_reward = 0
    for _ in range(200):
        action = np.argmax(Q[state, :])
        state, reward, done, _ = env.step(action)
        total_reward += reward
        if done:
            break
    rewards.append(total_reward)

print(f'Durchschnittliche Belohnung nach dem Training: {np.mean(rewards)}')

# Umgebung schließen
env.close()
```

Revision #4
Created 27 May 2024 22:07:57 by Shelly Pröhl
Updated 28 May 2024 12:30:46 by Shelly Pröhl