

# Reinforcement Learning - Projektvorstellung

Thema: Q-Learning für das Rucksack-Problem

Patrizia Schalk und Manuel Richter

February 5, 2018

- 1 Spiele / OpenAI Gym
- 2 Das Rucksack-Problem
- 3 Lösung durch Q-Learning
- 4 Anwendung: Ein Terminkalender
- 5 Ausblick

# Ursprüngliches Projekt

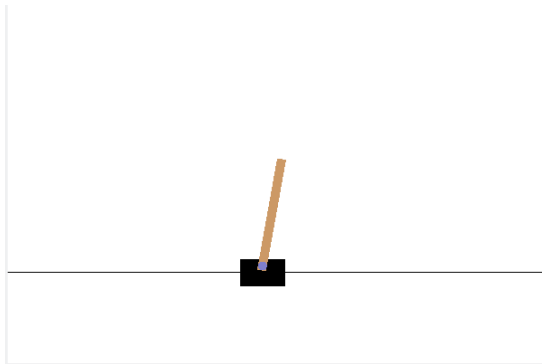
## Installation und Verwendung des OpenAI Gyms

### ■ Installation des OpenAI Gyms

# Ursprüngliches Projekt

## Installation und Verwendung des OpenAI Gyms

- Installation des OpenAI Gyms
- Implementieren von Q-Learning für das Cartpole-Problem



# Konzept

## Inspiration

Gegeben:

# Konzept

## Inspiration

Gegeben:

- Ein Rucksack mit vorgegebenem Maximalgewicht

# Konzept

## Inspiration

Gegeben:

- Ein Rucksack mit vorgegebenem Maximalgewicht
- Eine Menge von Gegenständen, denen jeweils Gewicht und Wert zugeordnet wird

# Konzept

## Inspiration

Gegeben:

- Ein Rucksack mit vorgegebenem Maximalgewicht
- Eine Menge von Gegenständen, denen jeweils Gewicht und Wert zugeordnet wird

Gesucht:



# Konzept

## Inspiration

Gegeben:

- Ein Rucksack mit vorgegebenem Maximalgewicht
- Eine Menge von Gegenständen, denen jeweils Gewicht und Wert zugeordnet wird

Gesucht:

- Welche Gegenstände können im Rucksack verstaut werden, ohne die Gewichtsgrenze zu überschreiten?

# Konzept

## Inspiration

Gegeben:

- Ein Rucksack mit vorgegebenem Maximalgewicht
- Eine Menge von Gegenständen, denen jeweils Gewicht und Wert zugeordnet wird

Gesucht:

- Welche Gegenstände können im Rucksack verstaut werden, ohne die Gewichtsgrenze zu überschreiten?
- Welche Gegenstände sollten ausgewählt werden, um den Gesamtwert zu maximieren?

# Konzept

## Abwandlung

Wie kann dieses Problem auf eine GridWorld übertragen werden?

# Konzept

## Abwandlung

Wie kann dieses Problem auf eine GridWorld übertragen werden?

- Anstelle einer Gewichtsgrenze definieren wir eine Platz-Grenze

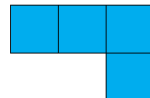
# Konzept

## Abwandlung

Wie kann dieses Problem auf eine GridWorld übertragen werden?

- Anstelle einer Gewichtsgrenze definieren wir eine Platz-Grenze

	0	1	2	3
0				
1				
2				
3				



# Definition der Parameter für Q-Learning

## Zustandsraum

Der optimale Spielzug unterscheidet sich je nachdem, wo Spielsteine gesetzt wurden.

# Definition der Parameter für Q-Learning

## Zustandsraum

Der optimale Spielzug unterscheidet sich je nachdem, wo Spielsteine gesetzt wurden.

⇒ Für jeden möglichen Status des Spielfeldes muss ein Zustand generiert werden.

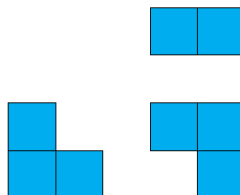
# Definition der Parameter für Q-Learning

## Zustandsraum

Der optimale Spielzug unterscheidet sich je nachdem, wo Spielsteine gesetzt wurden.

⇒ Für jeden möglichen Status des Spielfeldes muss ein Zustand generiert werden.

	0	1	2	3
0	1	1	1	1
1	1	0	0	1
2	1	1	1	0
3	0	1	1	0

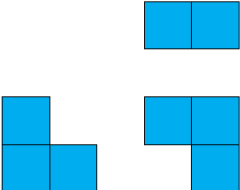




# Definition der Parameter für Q-Learning

## Zustandsraum

	0	1	2	3
0	1	1	1	1
1	1	0	0	1
2	1	1	1	0
3	0	1	1	0

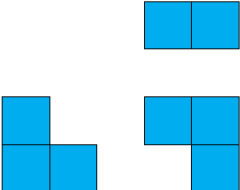


Zur Darstellung des Spielbrettes verwenden wir  $n^2$  Bits, indem wir das Spielfeld als binäre Darstellung eines Integers auffassen.

# Definition der Parameter für Q-Learning

## Zustandsraum

	0	1	2	3
0	1	1	1	1
1	1	0	0	1
2	1	1	1	0
3	0	1	1	0



Zur Darstellung des Spielbrettes verwenden wir  $n^2$  Bits, indem wir das Spielfeld als binäre Darstellung eines Integers auffassen.

In diesem Beispiel:  $(1111100111100110)_2 = 63.974$



# Definition der Parameter für Q-Learning

## Aktionen

Jeder Spielstein kann an eine beliebige Position des Spielfeldes gesetzt werden.

# Definition der Parameter für Q-Learning

## Aktionen

Jeder Spielstein kann an eine beliebige Position des Spielfeldes gesetzt werden.

Beispiel einer Aktion:  $((0, 0), 12)$ :

Setze Spielstein  an Position  $(0, 0)$

# Definition der Parameter für Q-Learning

## Ende einer Episode

Die Episode endet, wenn der Agent versucht, einen illegalen Spielzug auszuführen:

- Der gesetzte Spielstein liegt (zum Teil) auf einem bereits gesetzten Spielstein
- Der Agent versucht, einen Spielstein so zu setzen, dass er das Spielfeld überlappt

# Definition der Parameter für Q-Learning

## Reward

Für jeden erfolgreich gesetzten Spielstein erhält der Agent einen Reward von:

$$\frac{\text{Anzahl der neu verdeckten Felder}}{\text{Anzahl der gesamten Felder}}$$

# Definition der Parameter für Q-Learning

## Reward

Für jeden erfolgreich gesetzten Spielstein erhält der Agent einen Reward von:

$$\frac{\text{Anzahl der neu verdeckten Felder}}{\text{Anzahl der gesamten Felder}}$$

Besitzen die Spielsteine jeweils Geldwerte, so modifizieren wir den Reward wie folgt:

$$\frac{\text{Anzahl der neu verdeckten Felder}}{\text{Anzahl der gesamten Felder}} \cdot \frac{\text{Wert des gesetzten Spielsteins}}{\text{Maximaler auftretender Wert}}$$



# Modifikationen

## Spielfeldgröße

Für einen Terminkalender, der Termine für nur einen Tag organisiert, benötigen wir kein Spielfeld quadratischer Größe.

# Modifikationen

## Spielfeldgröße

Für einen Terminkalender, der Termine für nur einen Tag organisiert, benötigen wir kein Spielfeld quadratischer Größe.  
⇒ Die Breite des Spielfelds wird stets 1 sein

# Modifikationen

## Aktionen

- Unsere Spielsteine stehen für Termine, die in den Planer eingereiht werden sollen.

# Modifikationen

## Aktionen

- Unsere Spielsteine stehen für Termine, die in den Planer eingereiht werden sollen.
- Der Wert eines Spielsteins entspricht der Priorität einer Aufgabe.

# Modifikationen

## Aktionen

- Unsere Spielsteine stehen für Termine, die in den Planer eingereiht werden sollen.
- Der Wert eines Spielsteins entspricht der Priorität einer Aufgabe.
- Der Benutzer interessiert sich nur für das optimale Ergebnis und nicht für den Prozess dorthin.

# Weitere mögliche Anwendungen

- Basierend auf dem Terminkalender ließe sich ein Scheduler mittels Q-Learning implementieren

# Weitere mögliche Anwendungen

- Basierend auf dem Terminkalender ließe sich ein Scheduler mittels Q-Learning implementieren
- Ermöglicht man dem Agenten, die Spielsteine zu rotieren, ließe sich eine einfache Form von Tetris implementieren

# Vielen Dank für die Aufmerksamkeit.

## Noch Fragen?