

An Introduction to Reinforcement Learning

Samuel Kurath



- Reinforcement Learning
 - Motivation
 - Einführung RL
 - Theoretische Grundlagen
 - Q-Learning
- (Deep) Neural Networks
 - Motivation
 - Einführung NN
 - Perceptron
 - Regression
 - Fully Connected NN
- Deep Q-Learning
- Quiz Auswertung



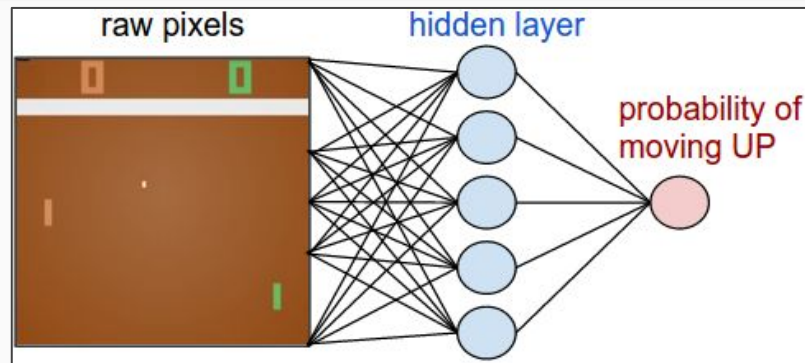
Reinforcement Learning - Motivation - Pong

Deep Reinforcement Learning - Pong

- Bild als Input (Pixels)
- Rechenintensive
- Anwendungsfelder
 - Verkehrsampel Steuerung
 - Ressourcenmanagement in Rechenzentren
 - Robotik

Video

- <https://www.youtube.com/watch?v=60pwnLB0DqY>



Beschreibung

- <http://karpathy.github.io/2016/05/31/rl/>

Reinforcement Learning


Rechnergestützter Ansatz, der versucht zu lernen, wie man sich in Bezug auf ein **Environment** (Umgebung) möglichst optimal verhält, um ein Ziel zu erreichen.


Dabei werden die **Actions** (Verhalten) eines **Agent** (Vertreter) optimiert, welcher vom Environment nur Informationen zum **State** (Zustand) und **Reward** (Belohnung) erhält.




Reinforcement Learning - Terminology (Begrifflichkeit)

Policy (π)  - Definiert das Verhalten / die Strategie eines Agents

Reward  - Beschreibung des Ziels; numerischer Wert, der vom Environment nach jeder Action zurückgegeben wird

Value function (v)  - Definiert die Güte des aktuellen Zustandes, dabei werden mögliche zukünftige Rewards auch in Betracht gezogen

Model  - Versucht das Environment zu modellieren, um Informationen über zukünftige mögliche Situationen zu erhalten

Markov Decision Process

Formelle Methode, um Reinforcement Learning Environments zu beschreiben. Dabei wird die **Markov Property** (der aktuelle Zustand enthält alle Informationen über die Vergangenheit, “memoryless”) erfüllt.

Markov Decision Process

- S - endliche Menge aller Zustände
- A - endliche Menge aller Actions (Übergänge zwischen Zuständen)
- R - Reward für jeden Zustandsübergang
- γ - Discount Factor, Wert um den Unterschied zwischen aktuellem und zukünftigen Reward zu quantifizieren

Markov Chain - Beispiel - Downhill

Tal (Ziel) möglichst "schnell" zu erreichen

Episode (Abfolge)

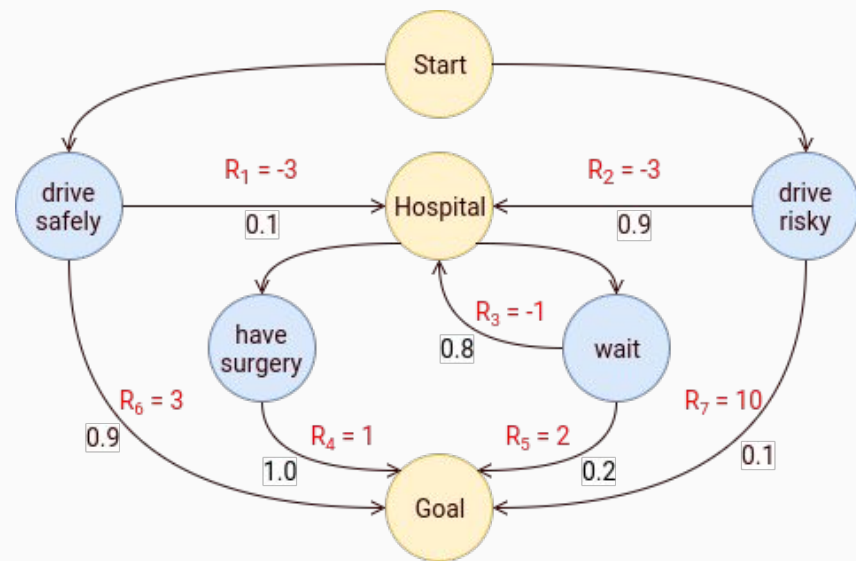
$S_{\text{Start}}, A_{\text{drive_risky}}, R_1, S_{\text{Hospital}}, A_{\text{have_surgery}}, R_4, S_{\text{Goal}}$

Reward

$$R_{\text{total}} = R_1 + R_4 = -3 + 1 = -2$$

Discounted Reward ($\gamma = 0.9$)

$$R_{\text{discounted}} = \gamma^0 R_1 + \gamma^1 R_4 = 0.9^0 \cdot (-3) + 0.9^1 \cdot 1 = -2.1$$



Gelb - Zustände (S)
Blau - Actions (A)
Rot - Reward (R)
Schwarz - W'keiten

Value Functions

Reinforcement Learning Algorithmen versuchen Value Functions abzuschätzen um möglichst passend Action durchzuführen und den Reward zu maximieren.

State-Value-Function

Beschreibt den erwarteten Wert zu einem Zustand (S) unter der Policy (π).

“Wie gut ist es, sich in einem bestimmten Zustand (S) zu befinden?”

Action-Value-Function

Gibt einen Wert, der beschreibt, wie der Übergang vom Zustand (S) mit der Action (A) zu Werten ist.

“Wie gut ist es, eine gewisse Action (A) im Zustand (S) durchzuführen?”

State-Value-Function

Formel
$$V^\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right]$$

E_π - Erwartungswert unter Policy (π)
 S_t^π - Zustand zu gewissem Zeitpunkt (t)

Downhill ($\gamma = 0.9$)

Policy “Vorsicht”

$$E_\pi(S_{Start}) = 0.1 \cdot (-3) + 0.9 \cdot 3 = 2.4$$

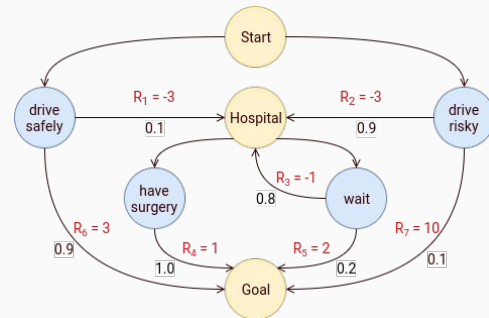
$$E_\pi(S_{Hospital}) = 1.0 \cdot 1.0 = 1.0$$

$E_\pi(S_{Goal}) = 0$, Wert des Terminal Zustandes ist immer 0

$$\begin{aligned} V^\pi(S_{Start}) &= \gamma^0 \cdot E_\pi(S_{Start}) + \gamma^1 \cdot E_\pi(S_{Hospital}) + \gamma^2 \cdot E_\pi(S_{Goal}) \\ &= 0.9^0 \cdot 2.4 + 0.9^1 \cdot 1.0 + 0.9^2 \cdot 0 = 3.3 \end{aligned}$$

$$\begin{aligned} V^\pi(S_{Hospital}) &= \gamma^0 \cdot E_\pi(S_{Hospital}) + \gamma^1 \cdot E_\pi(S_{Goal}) \\ &= 0.9^0 \cdot 1 + 0.9^1 \cdot 0 = 1.0 \end{aligned}$$

$$\pi : \begin{cases} S_{Start} \rightarrow A_{drive_safely} \\ S_{Hospital} \rightarrow A_{have_surgery} \end{cases}$$



Action-Value-Function

Formel

$$Q^{\pi}(s, a) = E_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right], \text{ for all } s \in S$$

E_{π} - Erwartungswert unter Policy (π)

S_t - Zustand zu gewissem Zeitpunkt (t)

Downhill ($\gamma = 0.9$)

Policy "Risiko"

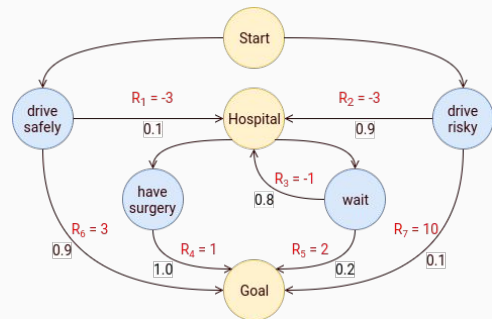
$$\pi : \begin{cases} S_{Start} \rightarrow A_{drive_risky} \\ S_{Hospital} \rightarrow A_{have_surgery} \end{cases}$$

$$E_{\pi}(S_{Start}, A_{drive_risky}) = 0.9 \cdot (-3) + 0.1 \cdot 10 = -1.7$$

$$E_{\pi}(S_{Hospital}, A_{have_surgery}) = 1.0 \cdot 1.0 = 1.0$$

$$E_{\pi}(S_{Goal}) = 0, \text{ Wert des Terminal Zustandes ist immer } 0$$

$$\begin{aligned} Q^{\pi}(S_{Start}, A_{drive_risky}) &= \gamma^0 \cdot E_{\pi}(S_{Start}, A_{drive_risky}) + \gamma^1 \cdot E_{\pi}(S_{Hospital}, A_{have_surgery}) + \gamma^2 \cdot E_{\pi}(S_{Goal}) \\ &= 0.9^0 \cdot -1.7 + 0.9^1 \cdot 1.0 + 0.9^2 \cdot 0 = -0.8 \end{aligned}$$



Q-Learning

Q-Learning ist eine modellfreie Reinforcement Learning Technik. Sprich, die Action-Value-Function (Q) ist unbekannt und soll selbständig und möglichst optimal erlernt werden.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

update

a - alle mögliche nächste Zustände

α - Explorations-Rate (0, 1]

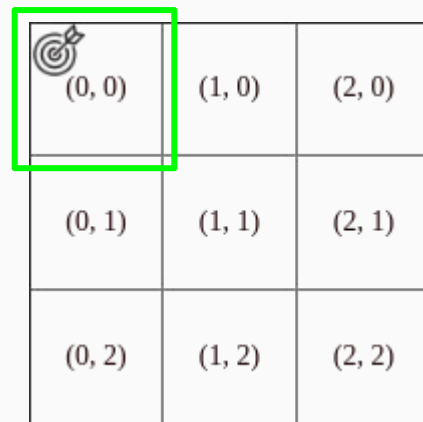
Q-Learning - Beispiel - Grid World


Grid World

Ziel, mit so wenig Actions wie möglich Zielfeld gelangen.

Rahmenbedingungen:

- 3 x 3 Welt (Grid)
- (0, 0) ist das Zielfeld
- Startfeld arbiträr
- Erlaubte Actions
 - auf, ab, link, rechts
- Nicht erlaubte Actions
 - diagonal, die Welt verlassen




 (0, 0)	(1, 0)	(2, 0)
(0, 1)	(1, 1)	(2, 1)
(0, 2)	(1, 2)	(2, 2)

Deep Learning meets Q-Learning → Deep Q-Learning

- **Motivation**, oft ist es nicht möglich, alle möglichen Zustände eines Environments abzubilden / abzuarbeiten / zu erlernen
- Neuronale Netze, die die **Action-Value-Function** approximieren

Vorgehen

1. Neuronales Netz zufällig initialisieren
2. Spiele simulieren und merken
3. Basieren auf den Simulationen Neuronales Netzwerk trainieren
4. Repetieren, bis man mit dem Resultat zufrieden ist

 (0, 0)	(1, 0)	(2, 0)
(0, 1)	(1, 1)	(2, 1)
(0, 2)	(1, 2)	(2, 2)

- Reinforcement Learning: An Introduction
 - Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.
- The Paths Perspective on Value Learning
 - <https://distill.pub/2019/paths-perspective-on-value-learning/>

Über mich



Name - Samuel Kurath

Bachelor - Informatik HSR (2013 - 2016)

Master - Informatik HSR (2016 - 2019)

Arbeit - HSR Institut für Software (IFS)
(2016 -2020)

Arbeiten

Bachelor

- SA - [Crosswalk Detection](#)
- BA - StrongMan (UI for StrongSwan)

Master

- PA1 - Rank (App to reduce downhill risks)
- PA2 - [Reinforcement Learning Schieber Jass Bot](#)
- MA - [DeepSquat](#)