

# TT2 Problemstellung 3 : Ausarbeitung

Pascal Jäger, Stefan Münchow, Armin Steudte,  
Milena Rötting, Sven-Andjes Pahl, Carsten Noetzel, Oliver Steenbuck

01.06.2012

## Inhaltsverzeichnis

<b>1 Fragestellung 1: V-Werte / Q-Werte</b>	<b>1</b>
<b>2 Fragestellung 2: SARSA / Q-Learning bei kleinem <math>\epsilon</math></b>	<b>1</b>
2.1 On-Policy: SARSA . . . . .	1
2.2 Off-Policy: Q-Learning . . . . .	2
2.3 Verhalten bei kleinem $\epsilon$ . . . . .	3
<b>3 Fragestellung 3: kontinuierliche Zustandsvariablen</b>	<b>3</b>
<b>4 Fragestellung 4: Kniffel</b>	<b>3</b>

## Abbildungsverzeichnis

1 Sarsa: On Policy TD Control Algorithmus . . . . .	2
2 Q Learning: Off Policy TD Control Algorithmus . . . . .	2

## 1 Fragestellung 1: V-Werte / Q-Werte

## 2 Fragestellung 2: SARSA / Q-Learning bei kleinem $\epsilon$

### 2.1 On-Policy: SARSA

Basis für den SARSA-Algorithmus sind die Q-Werte, also Zustands-Aktions-Paare  $Q(s, a)$ . Die Bewertung eines solchen Zustands-Aktions-Paares erfolgt durch die unten genannte Gleichung welche auch in Abbildung 1 zu finden ist.

Der erwartete Wert  $Q(s, a)$  für eine Aktion  $a$  im Zustand  $s$  wird dabei aktualisiert durch, den bereits ermittelten, erwarteten Wert  $Q(s, a)$  und dem Schrittfaktor  $\alpha$ , der mit der Summe aus Reward für den Folgezustand  $r_{t+1}$  und der Differenz aus dem discounted Reward für das Folge-Zustands-Aktions-Paar  $\gamma Q(s_{t+1}, a_{t+1})$  und dem erwarteten Wert  $Q(s, a)$  addiert wird.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (1)$$

In Abbildung 1 ist zu erkennen, dass die Aktionen  $a'$  im Zustand  $s'$  in Abhängigkeit von der Policy gewählt werden, die zum Beispiel  $\epsilon$ -Greedy sein kann. Das heißt es wird die meiste Zeit diejenige Aktion  $a'$  ausgewählt die den größten erwarteten Gewinn verspricht und mit einer Wahrscheinlichkeit  $\epsilon$  eine zufällige Aktionen, um die Umgebung weiter zu explorieren und nicht deterministisch zu sein.

```

Initialize  $Q(s, a)$  arbitrarily
Repeat (for each episode):
  Initialize  $s$ 
  Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
  Repeat (for each step of episode):
    Take action  $a$ , observe  $r, s'$ 
    Choose  $a'$  from  $s'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
     $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'; a \leftarrow a';$ 
  until  $s$  is terminal

```

Abbildung 1: Sarsa: On Policy TD Control Algorithmus

## 2.2 Off-Policy: Q-Learning

Auch das Q-Learning Verfahren setzt auf den Q-Werten auf und berechnet diese über die unten genannte Formel, die Teil des Algorithmus in Abbildung 2 ist.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (2)$$

Der Unterschied zum Sarsa-Algorithmus ist hierbei der Teil  $\gamma \max_a Q(s_{t+1}, a)$ . Anstelle den erwarteten Return  $Q(s_{t+1}, a_{t+1})$  für die gewählte Aktion  $a'$  im Zustand  $s'$  mit dem Discount-Faktor  $\gamma$  zu multiplizieren wie es beim Sarsa-Algorithmus gemacht wird, wird hier diejenige Aktion bestimmt, die den größten Reward verspricht und deren Reward  $\max_a Q(s_{t+1}, a)$  mit dem Discount-Faktor multipliziert. **Damit wird der Q-Wert  $Q(s, a)$  unabhängig von der Strategie, allein auf Basis des Returns und der Q-Werte benachbarter Zustände aktualisiert!** Unabhängig davon, welche Aktion die  $\epsilon$ -Greedy Policy wählt, wird immer der Q-Wert zur Berechnung genommen, die den höchsten Reward verspricht und nicht der, des durch die Policy gewählten Nachbarn.

```

Initialize  $Q(s, a)$  arbitrarily
Repeat (for each episode):
  Initialize  $s$ 
  Repeat (for each step of episode):
    Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
    Take action  $a$ , observe  $r, s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s';$ 
  until  $s$  is terminal

```

Abbildung 2: Q Learning: Off Policy TD Control Algorithmus

## 2.3 Verhalten bei kleinem $\epsilon$

Das Verhalten vom Sarsa-Algorithmus nähert sich dem vom Q-Learning an, da mit kleiner werdendem  $\epsilon$  die Wahrscheinlichkeit sinkt eine zufällige Aktion auszuwählen. Bei sehr kleinem  $\epsilon$  wird vermehrt die Aktion  $a'$  ausgewählt, die den größten Reward verspricht, ähnlich wie es beim Q-Learning mit  $\max_a Q(s_{t+1}, a)$  getan wird. Damit nähert sich das Verhalten von Sarsa dem des Q-Learnings an.

## 3 Fragestellung 3: kontinuierliche Zustandsvariablen

Partitionierung/Decomposing  
states verwerfen

machine  
learning  
state space  
too big de-  
composition

## 4 Fragestellung 4: Kniffel

### Todo list

■ machine learning state space too big decomposition . . . . . 3