Deep Reinforcement Learning

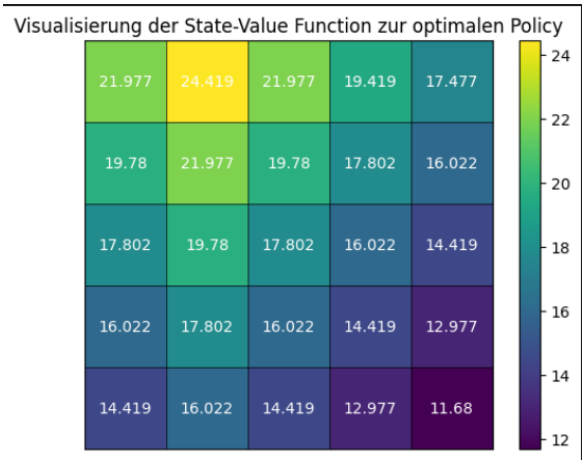
Sheet 03

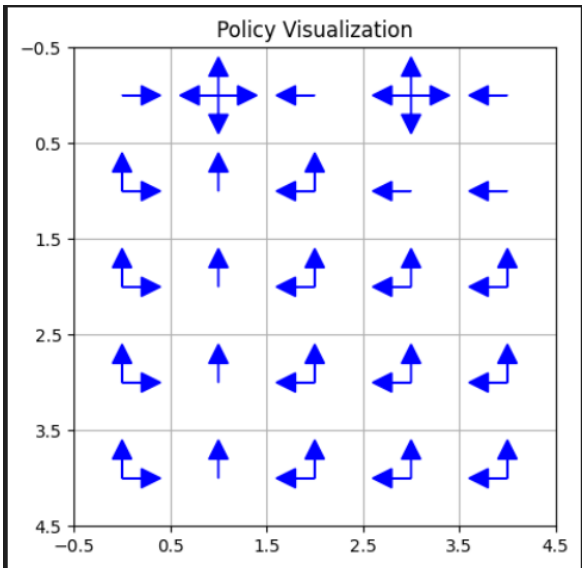
Sven Ullmann, Valentin Adam

Exercise 1

a) Visualisierung:

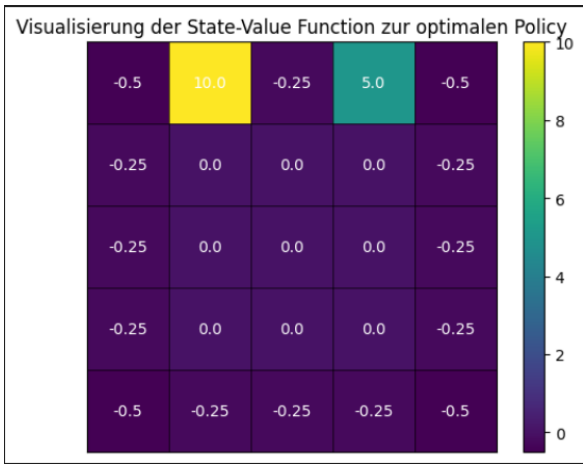
Im folgenden die optimal policy und die dazugehörige State-Value function für die GPI.

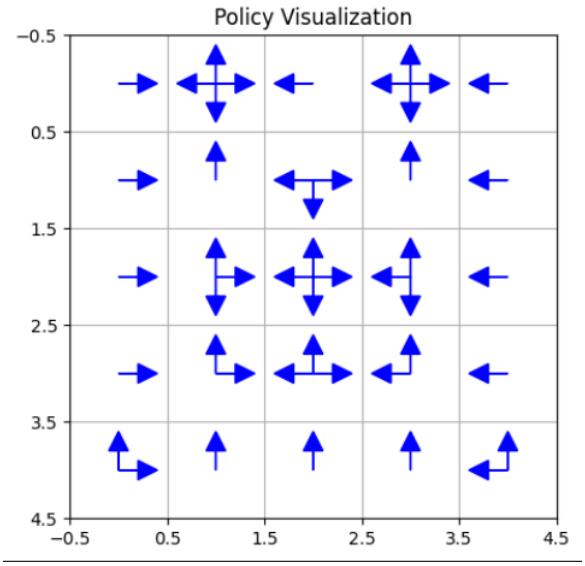


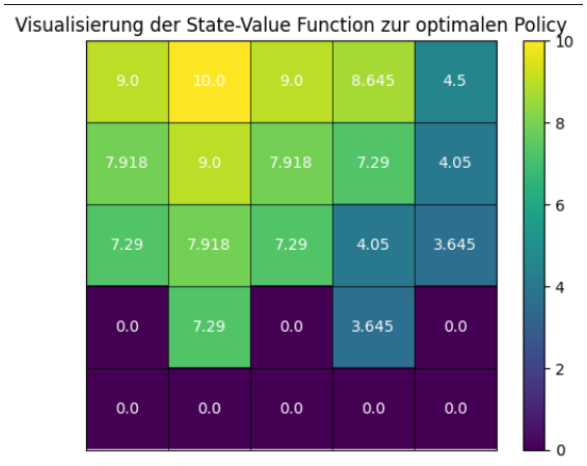


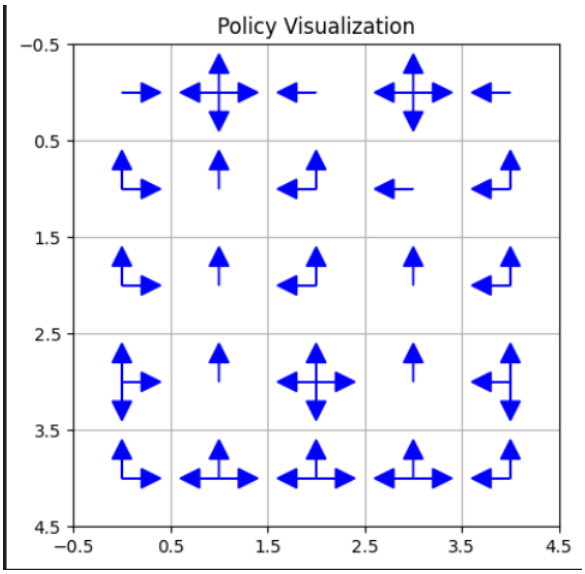
b) Visualisierung:

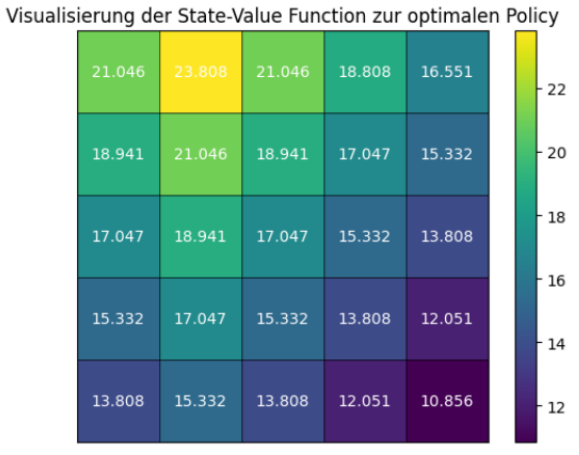
Wir haben uns dafür entschieden, in diese Dokument nicht alle plots (also nach jedem update) einzufügen, sondern wir haben uns auf ein paar wenige beschränkt.

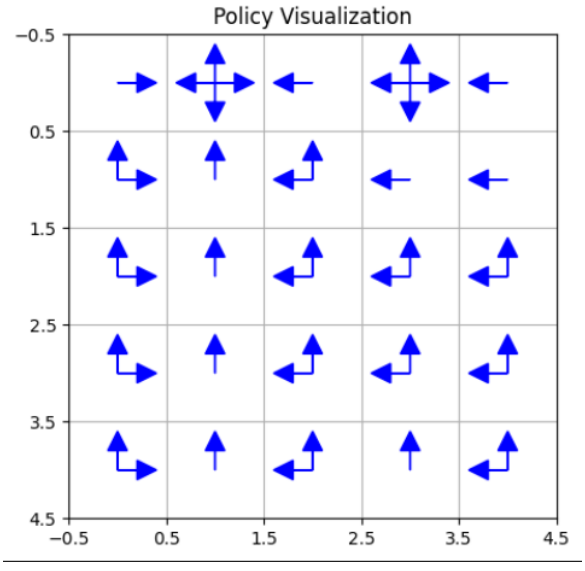

Visualisierung der State-Value Function zur optimalen Policy
-0.5 10.0 -0.25 5.0 -0.5
-0.25 0.0 0.0 0.0 -0.25
-0.25 0.0 0.0 0.0 -0.25
-0.25 0.0 0.0 0.0 -0.25
-0.5 -0.25 -0.25 -0.25 -0.5
10
8
6
4
2
0


Policy Visualization
−0.5
0.5
1.5
2.5
3.5
4.5
−0.5 0.5 1.5 2.5 3.5 4.5


Visualisierung der State-Value Function zur optimalen Policy
9.0 10.0 9.0 8.645 4.5
7.918 9.0 7.918 7.29 4.05
7.29 7.918 7.29 4.05 3.645
0.0 7.29 0.0 3.645 0.0
0.0 0.0 0.0 0.0 0.0
10
8
6
4
2
0


Policy Visualization
−0.5
0.5
1.5
2.5
3.5
4.5
−0.5 0.5 1.5 2.5 3.5 4.5


Visualisierung der State-Value Function zur optimalen Policy
21.046 23.808 21.046 18.808 16.551
18.941 21.046 18.941 17.047 15.332
17.047 18.941 17.047 15.332 13.808
15.332 17.047 15.332 13.808 12.051
13.808 15.332 13.808 12.051 10.856
22
20
18
16
14
12


Policy Visualization
−0.5
0.5
1.5
2.5
3.5
4.5
−0.5 0.5 1.5 2.5 3.5 4.5

Analyse:

Beide Implementierungen (GPI und value function iteration) führen zum selben Ergebnis in der optimal policy. Die Werte in der State Value funktion variieren leicht. Der Grund dafür ist, dass bei a) (also GPI), die state Value function immer bis zur Konvergenz durchgerechnet wird, bei b) allerdings immer nur eine Iteration der state value function Berechnung gemacht wird, bevor man die policy updatet. Da unser abbruchkriterium eine sich nicht mehr änderende policy ist, sind diese leichte Unterschiede in der state value function erwartbar und nicht weiter problematisch.

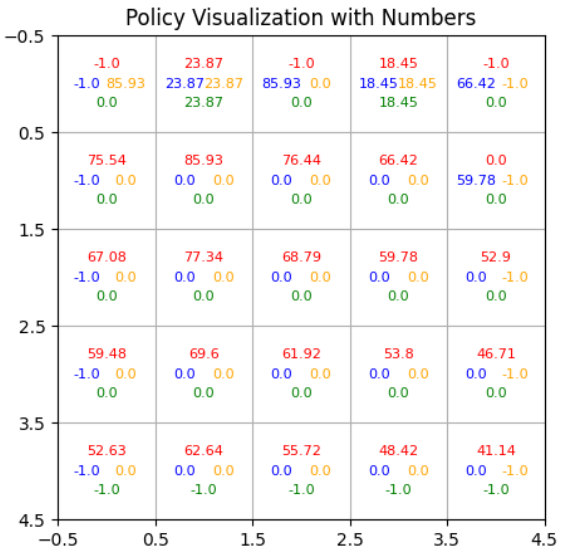
Weiterhin kann man sagen, dass die GPI viel länger braucht, bis sie die optimale policy erreicht. Bei GPI wird die state value function für jedes policy update komplett berechnet, was bei uns 315 Iterationen benötigt (die genau zahl ist abhängig vom gewählten abbruchsepsilon). Dann benötigen wir bei GPI 4 Policy updates, bis die optimale policy erreicht wird. Insgesamt macht das also mehr als 1200 Iterationen in der State value function Berechnung. Bei Value function iteration, also b), wird die policy immer nach einem Schritt upgedatet. Bis wird zur optimalen policy konvergiert sind, dauert es bei uns < 30 Iterationen, also auch < 30 Itertionen insgesamt bei der State value function Berechnung.

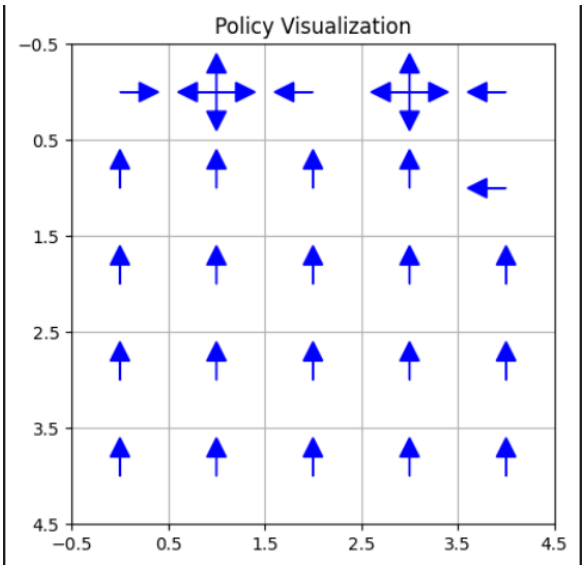
c) Reflexionsfrage:

Eigenschaft	**State-Value** $V(s)$	**Action-Value** $Q(s, a)$
Bewertet	Zustände	Zustände und Aktionen
Nutzung in Policy Improvement	Indirekt, erfordert Umwandlung zu $Q(s, a)$	Direkt, maximiert $Q(s, a)$
Rechenaufwand	Höher, da Aktionen erst evaluiert werden müssen	Geringer, da Aktionen direkt verglichen werden

Table 1: Vergleich zwischen State-Value- und Action-Value-Funktionen

Visualisierung:





Analyse:

Zur Bestimmung der optimalen policy haben wir immer für jeden State die action values der möglichen Aktionen verglichen und davon das maximum genommen. Wir sind uns nicht sicher, ob das hier gemeint

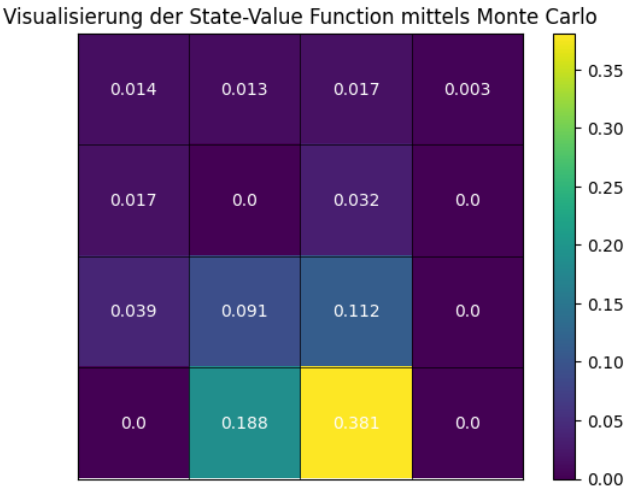
war, aber wir haben keine Formel dafür im Skript gefunden und halten die Ansatz für vertretbar. Damit konvergiert (i.e keine Änderungen mehr in der policy) es bereits nach einem policy update. Diese Methode ist wesentlich simpler, als das was in aufgabenteil a) und b) gemacht wurde, daher macht es für uns auch Sinn, dass das die damit berechnete optimal Policy etwas simpler aussieht.

Exercise 2

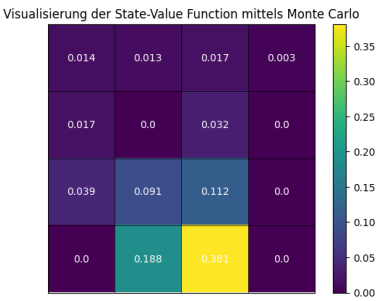
a) Reflexionsfrage:

Eine initiale Policy (z. B. zufällig) ist notwendig, um die Umgebung zu erkunden und eine Vorstellung davon zu bekommen, welche Aktionen zu guten Ergebnissen führen. Eine zufällige Policy ist oft suboptimal, da sie möglicherweise nicht zielgerichtet ist (z. B. kann der Agent in einem Loch landen). Allerdings hilft sie, die gesamte Umgebung zu erkunden. Trotzdem ist eines der größten Probleme bei Monte Carlo Simulation zur Schätzung der State-Values ist die potentiell sehr hohe Varianz. Wenn einige States nur sehr selten von der Simulation besucht werden, haben die geschätzen Werte dieser Varianten eine hohe Varianz.
Daher sollte eine initiale Policy zumindest sicherstellen, dass das Ziel erreichbar ist und realtiv häufig besucht wird (z. B. durch Priorisieren von Aktionen, die den Agenten näher an das Ziel bringen). Beispiel: Das Ziel G liegt bei (3,3) Die Policy könnte versuchen: Rechts, wenn das Ziel in der gleichen Reihe liegt und weiter rechts ist. Unten , wenn das Ziel unter dem aktuellen Zustand liegt. Dafür muss selbstverständlich die location des Ziels bekannt sein.

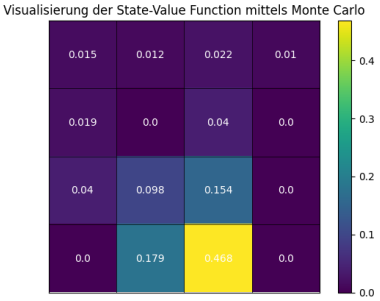
Visualisierung:



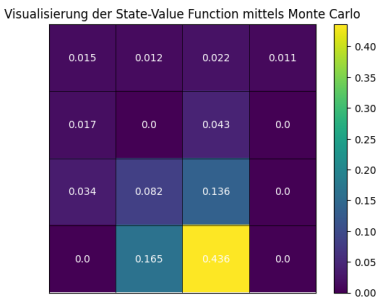
Analyse:
Die oben genannte hohe Varianz der zufälligen Policy ist genau das Problem was wir beobachten. Trotz 50 000 Monte Carlo Simulationen erhalten wir bei mehrmaliger Schätzung immer wieder verschiedene Schätzungen für die State Values. So erhalten wir bei 3 maliger Ausführug unseres Code drei signifikant unterschiedliche Ergebnisse. Dementsprechend benötigt man sehr sehr viele Episoden um die Varianz



(a) 1. Run



(b) 2. Run



(c) 3. Run

der Lösungen zu verringern. Selbst bei 1 000 000 Episoden bekommen wir immernoch unterschiedliche

Lösungen. Um eine optimale Policy zu finden muss zunächst eine bessere Policy gefunden werden, die den Zielzustand wahrscheinlicher macht.