Organic Computing 2

Lösungsvorschlag Blatt08

Lukas Huhn Qiang Chang Victor Gerling

7. Juli 2019

Universität Augsburg
Institut für Informatik
Lehrstuhl für Organic Computing

Gliederung

Aufgabe 02

Universität Augsburg University

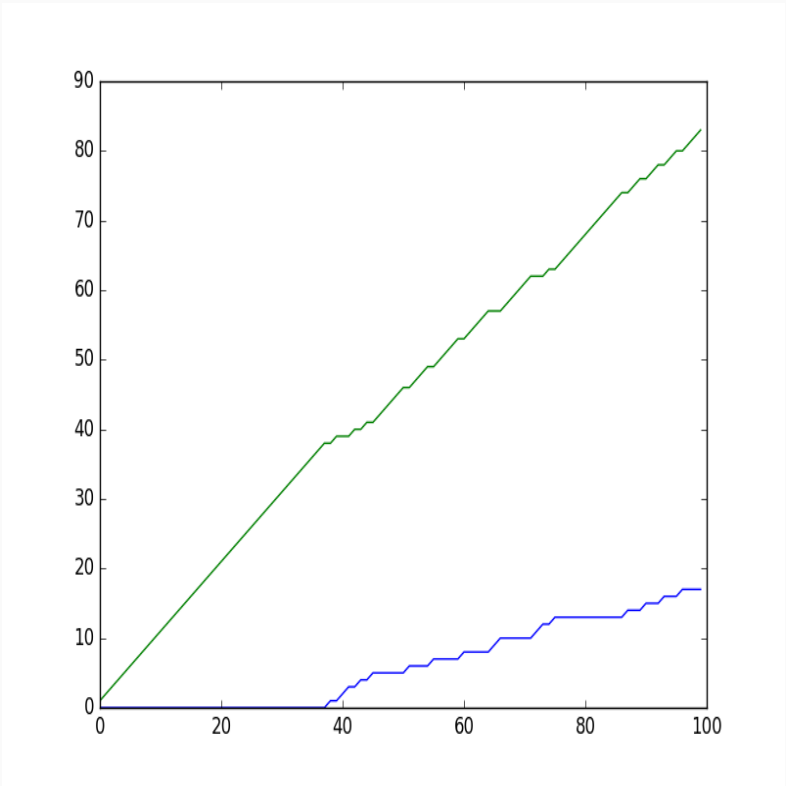
Implementieren Sie diesen und wenden Sie ihn auf das FrozenLake-v0-Szenario aus dem OpenAI- Gym an!

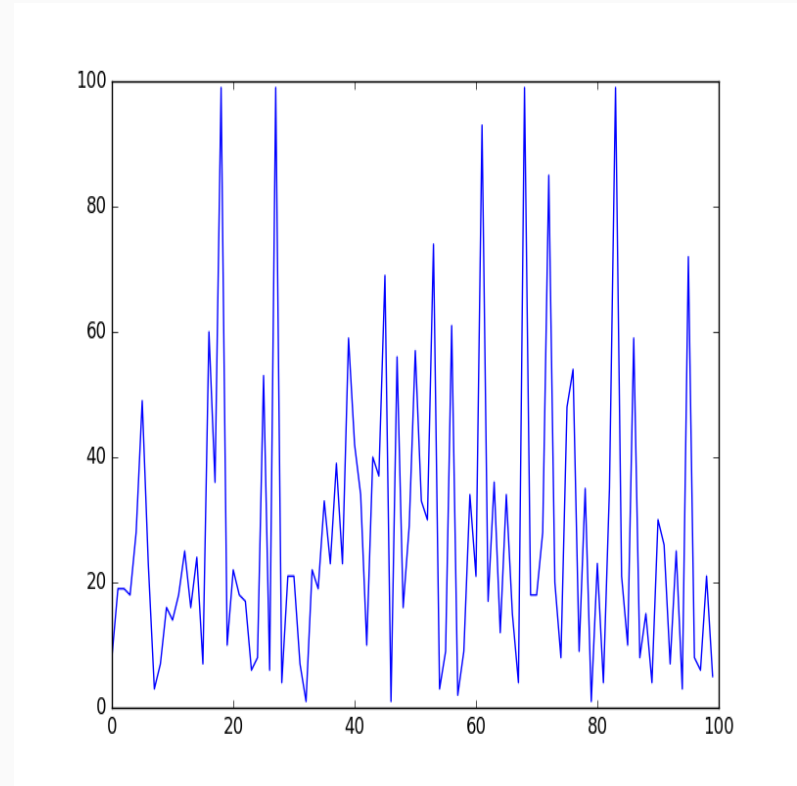
- siehe Code.

Finden Sie durch Ausprobieren oder eine kleine Parameterstudie geeignete Werte für die drei Parameter ε, γ (Diskontierungsfaktor) und α (Lernrate)!

- alpha = 0.1
- gamma = 0.5
- epsilon = 0.09

Zeichnen Sie ein Episode-Return-Diagramm sowie ein Episode-Returndurchschnitt-Diagramm, wobei der Durchschnitt in Fenstern jeweils über die 100 Episoden vor der jeweiligen Episode gebildet werden soll!


90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100


100
80
60
40
20
0
0
20
40
60
80
100

Nach wie viel Episoden ist der Lernprozess zufriedenstellend abgeschlossen? Begründen Sie!

- Ein Versuch benötigte 452 Episoden um einen Gewinndurchschnitt von 0.51 zu erzielen.

Ist Q-Learning gut geeignet für dieses Szenario?

- Ja, da es einen diskreten Zustand- und Aktionsraum gibt, mit nur 16 möglichen Zuständen und 4 möglichen Aktionen. Somit benötigen wir nur 16*4 = 64 q-Werte.

Aufgabe 03

Kann Ihre Q-Learning-Implementierung auch das CartPole-v1-Problem lösen? Wenn nicht: Warum nicht?

Problem bei CarPole reicht der observation space von [-4.8000002e+00 -3.4028235e+38 -4.1887903e-01 -3.4028235e+38] bis [4.8000002e+00 3.4028235e+38 4.1887903e-01 3.4028235e+38]

1. Problem: Ist vier-dimensional. Lässt sich prinzipiell auf eine Array-Dimension reduzieren: z.B. bei 8 Werten pro Richtung ergeben sich $8^4 states$

2. Problem Werte sind nicht quantisiert. Es liegen zwischen -4 und 4 unendlich viele reale Werte. Lässt sich in Stufen aufteilen

def quantize(val, min, max, levels): val = math.floor((val - min) * levels / (max - min)) return val

3. Problem zweite und vierte Spalte reichen von -unendlich bis +unendlich, während beim Testen nur Werte nah an der null beobachtet werden konnten. Damit werden beide bei der Quantisierung immer auf 0 abgebildetet also nicht vom Agenten wahrgenommen.

Wie könnte man Ihre Implementierung anpassen, um doch eine Lösung zu finden?

Für die Anpassung muss die Größe der q-Tabelle die Größe des quantisierten state-spaces haben, in unserem Fall $(8^4) * 2$, wobei 2 die Größe des Actionspaces bei Cartpole ist (0, 1).

Setzen Sie Ihre Idee um. Funktioniert sie? Weshalb (nicht)? Wie lange dauert der Lernprozess?

Es wurde mit 4096 * 4 * 10 Iterationen Gelernte, sodass jedes gültige State-Action-Paar durchschnittlich 10 mal besucht werden konnte. Das Ergebnis ist ernüchternt, der Agent hält das Gleichgewicht keine Sekunde.

Die liegt vermutlich an den nicht-wahrgenommen Zustands-Attributen.

Vielleicht kann wäre auch eine andere Form der Generalisierung für die reelwertigen Attribute günstiger.