

Universität Augsburg Institut für Informatik Lehrstuhl für Organic Computing Prof. Dr. Jörg Hähner Ansprechpartner

Wenzel Pilar von Pilchau, M. Sc. wenzel.pilar-von-pilchau@informatik.uni-augsburg.de Eichleitnerstr. 30, Raum 507

Wintersemester 2019/2020

Praktikum Selbstlernende Systeme

Aufgabenblatt 1

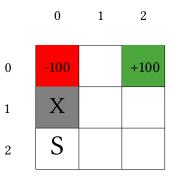
Schicken Sie Ihre Lösung in der Form die in der ersten Veranstaltung festgelegt wurde bis zum **Sonntag, den 27.10.2019 um 24:00 Uhr** an obenstehende E-Mail-Adresse.

1 StarCraft II Framework in Python

- 1. Installieren Sie, (sofern noch nicht geschehen) Python 3.6 → www.python.org Empfehlung: Benutzen Sie Anaconda (www.anaconda.com) um ihre Python Pakete und Installationen zu verwalten. Installieren Sie dafür Anaconda/Miniconda und erstellen Sie ein neues virtuelles Environment mit dem Namen pysc2 in dem Sie dann alle weiteren Pakete installieren können. Folgen Sie bei der Erstellung der Anleitung in PySC2_Env_Setup.pdf. Alle Informationen zu Anaconda finden Sie in der Dokumentation.
 Eine Einführung zu virtuellen Environments und Miniconda finden Sie hier.
- 2. Installieren Sie **pysc2** \rightarrow www.github.com/deepmind/pysc2 Folgen Sie dabei der Anleitung im *README.md*

2 Markov Decision Process (2 Punkte)

Gegeben ist der folgende Markov Decision Process:



- Der Agent beginnt in State (2,0) gekennzeichent mit einem großen S
- Felder mit einem X sind nicht betretbar
- Der Agent hat 4 Aktionsmöglichkeiten (N, O, S, W)
- Der Agent führt mit einer Wahrscheinlichkeit von 70% die Aktion aus die er gewählt hat und geht in die entsprechende Richtung. Mit einer Wahrscheinlichkeit von jeweils 10% geht er in eine der anderen Richtungen
- Wenn der Agent in einer Richtung nicht weiter kommt, dann bleibt er stehen
- Der Agent bekommt in jedem Schritt einen Reward von -2
- Wenn der Agent stattdessen eines der farbigen Felder verlässt und in den Endzustand gelangt, dann erhält er stattdessen den Reward der auf dem Feld steht
- 1. Berechnen Sie die V-Values $V^k(s)$ für alle Zustände $s \in S$, k = 5, $\gamma = 0.9$
- 2. Ermitteln Sie aus den V-Values aus der vorherigen Teilaufgabe die optimale Policy $\pi(s)$ für alle Zustände $s \in S$

3 Basic Agent (2 Punkte)

- 1. Erstellen Sie ein Python-Projekt (Beispielsweise mit PyCharm) in das alle folgenden Aufgaben (Blätter) includiert werden. Das Projekt soll nach ihrer RZ-Kennung benannt werden. Folgen Sie bei der Erstellung des Projektes der Anleitung in *SLS_Boilerplate.pdf*.
- 2. Implementieren Sie einen Agenten der das Minigame "MoveToBeacon" löst.
 - a) In jedem Schritt hat der Agent dabei 8 Möglichkeiten sich für eine Richtung zu entscheiden. (N, NO, O, SO, S, SW, W, NW)
 - b) Die Größe des Spielfeldes beträgt 64×64

- c) Der Agent wählt in jedem Schritt die Aktion, die den Abstand zum Ziel verringert.
- 3. Der Agent soll durch ein Python File (RunBasicAgent.py) gestartet werden können. Das File soll auf oberster Ebene des Projekts liegen. Kopieren Sie das Script *runScript.py* aus dem Boilerplate und passen Sie es entsprechend an.

4 Q-Learning (6 Punkte)

- 1. Erweitern Sie den Agenten aus der vorherigen Aufgabe um eine Q-Learning Komponente.
 - Ein Durchlauf geht 1920 frames (Voreinstellung der Minigame-Karte)
 - Es ist erlaubt im ersten Schritt eines Durchlaufes den Marine auszuwählen.
 - Eine Episode geht solange bis der Agent das Ziel erreicht oder der Durchlauf vorbei ist.
 - Für die Modellierung des Zustandsraumes soll nur der x- und y-Abstand vom Marine zum Beacon benutzt werden. Außerdem soll der Zustandsraum in Bereiche aufgeteilt werden:

$$x_1 < s_1 < x_2$$

$$y_1 < s_1 < y_2$$

Das dient dazu den Zustandsraum zu beschränken.

- Der Agent erhält einen Reward von 1 wenn er das Ziel erreicht. (benutze hierfür die bereits implementierte Reward-Funktion von pysc2)
- Als Explorationsstrategie soll ϵ -greedy mit linear abnehmendem ϵ verwendet werden.
- Die Q-Tabelle soll mit pandas (https://pandas.pydata.org/) angelegt und im Pickle-Format gespeichert werden.
- 2. Erstellen Sie mit Hilfe des Tensorboards zwei Graphen welche den Lernfortschritt des Agenten abbilden.
 - Erweitern Sie die *summarize*-Methode in der Klasse runner so, dass der Graph einen moving Average der Rewards mit einem sliding window = 50 abbildet.
 - Der zweite Graph soll das verwendete Epsilon abbilden.
- 3. Die Graphen die Ihren Trainingsvortschritt abbilden, sind Teil der Abgabe.
- 4. Der trainierte Q-Learning-Agent soll durch ein Python File (RunQLAgent.py) gestartet werden können. Der Agent soll die trainierte Q-Tabelle (von Ihnen trainiert) einlesen und diese benutzen ohne weiter zu lernen.

5. Das Training des Q-Learning-Agenten soll durch ein Python File (TrainQLAgent.py) gestartet werden können. Der Agent soll hierbei eine neue Q-Tabelle anlegen.
Viel Erfolg bei der Bearbeitung!
4
1