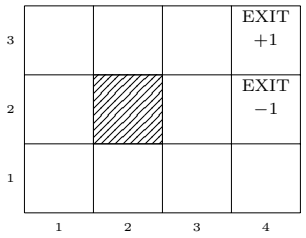
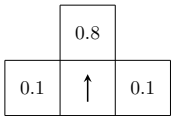
Aufgabe 23 (Markov Decision Process für 4×3 Spielfeld)



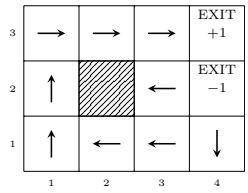


Wir betrachten das folgende Modell für einen Markov Decision Process: Der Zustandsraum $\mathcal{S}$ besteht aus den Feldern eines 4×3 Spielfelds, mit Ausnahme des nicht betretbaren Feldes $(2, 2)$, sowie einem "game over"-Zustand. In diesen geht der Prozess nach Betreten der Exit-Felder $(4, 3)$ und $(4, 2)$ mit Reward $+1$ bzw. -1 über. Die Menge der Aktionen in jedem Zustand besteht aus den vier Richtungen "oben", "unten", "links", "rechts", dargestellt als $\mathcal{A} = \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$.
Mit Ausnahme der Exit-Felder ist die Übergangswahrscheinlichkeit $\mathbb{P}(s_{t+1}|s_t, a_t)$ zum nächsten Zustand s_{t+1} dadurch festgelegt, dass sich der Agent mit 80% Wahrscheinlichkeit in die Pfeilrichtung der gewählten Aktion bewegt, und mit jeweils 10% Wahrscheinlichkeit in eine dazu senkrechte Richtung. (Falls der Agent auf ein unzulässiges Feld ziehen würde, verbleibt er auf dem aktuellen Feld.)
Die Reward-Funktion $R : \mathcal{S} \to \mathbb{R}$ ist definiert als

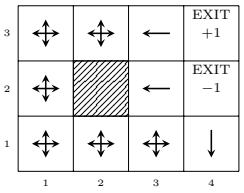
$$R(s) = \begin{cases} +1 & s = (4, 3) \\ -1 & s = (4, 2) \\ 0 & s = \text{"game over"} \\ r & \text{sonst} \end{cases}$$

wobei $r \in \mathbb{R}$ ein (noch festzulegender) Parameter des Modells ist.

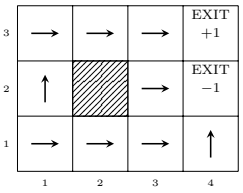
(a) Die folgenden Abbildungen zeigen die optimale Policy für verschiedene Werte von r und Abschlagfaktor $\gamma = 1$, wobei das Pfeilkreuz für eine beliebige Aktion steht. Ordnen Sie die Abbildungen den Werten $r = -1.8$, $r = -0.25$, $r = -0.01$, $r = 0.05$ zu, und begründen Sie Ihre Zuordnung kurz. [2 Punkte]



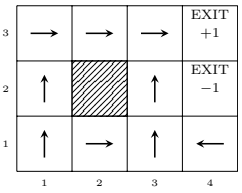
(i)



(ii)



(iii)



(iv)

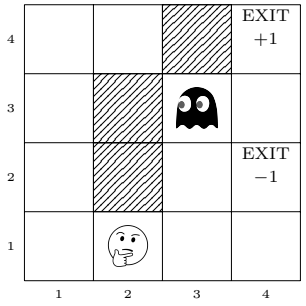
(b) Berechnen Sie (numerisch) jeweils die Bewertungsfunktion

$$U^\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \Big| \pi\right]$$

für beliebigen Startzustand s_0, wobei π die in den Abbildungen gezeigte Policy bezeichnet, r den zugeordneten Wert besitzt und jetzt $\gamma = 0.99$ sei. [4 Punkte]

Hinweis: Stellen Sie die Übergangswahrscheinlichkeiten (manuell oder mit einem Python-Programm) auf, und lösen Sie anschließend das im Rahmen der Policy-Iteration auftretende Gleichungssystem für U^π.

Aufgabe 24 (Iterative Algorithmen für Markov Decision Processes)
In `mdp_maze.zip` auf der Webseite zur Vorlesung findet sich eine Vorlage iterativer Algorithmen (siehe `mdp.py`) zur Bestimmung der optimalen value function $U(s)$, policy $\pi^*(s)$ oder Q-value function $Q^*(s,a)$ eines abstrakten MDPs, sowie deren Anwendung auf ein Labyrinth mit einem Geist (vereinfachte Version von Pac-Man). Die Spiellogik, d.h. die "environment" für den Agenten des MDP, wird in der Klasse `MazeGhostEnv` innerhalb `env.py` definiert. Der Zustandsraum besteht aus allen Kombinationen von Positionen des Spielers (Agenten) und des Geistes, sowie einem "game over"-Zustand. Das Spiel endet, wenn der Spieler ein Exit-Feld erreicht oder vom Geist "erwischt" wird (d.h. auf demselben Feld wie der Geist landet, mit "reward" $r = -2$). Die möglichen Aktionen und Bewegung des Spielers ist identisch zu A23; der Geist bewegt sich zufällig, mit Wahrscheinlichkeit 0.4 in Richtung des Spielers und mit Wahrscheinlichkeit 0.2 in eine der drei anderen Richtungen.



(a) Vervollständigen Sie den Code in `mdp.py` an den angegebenen Stellen, und verifizieren Sie Ihre Lösung mittels `python test_mdp.py`. [3 Punkte]

(b) Lassen Sie nun `run_ghost_maze.py` laufen, um die Iterationsalgorithmen aus `mdp.py` anzuwenden, und geben Sie die Ausgabe des Programms an. Modifizieren Sie anschließend die Geometrie des Labyrinths nach Belieben, indem Sie die Textdatei `ghost_maze.txt` bearbeiten und das Programm nochmals aufrufen. [1 Punkt]

(c) In `ghost_maze_qlearn.py` findet sich eine Vorlage für den Q-learning-Algorithmus, der die Q-value function anhand vieler Spielsimulationen (Episoden) approximativ berechnet, ohne die Übergangswahrscheinlichkeiten explizit zu verwenden. Vervollständigen Sie den Code an der angegebenen Stelle und lassen Sie das Programm laufen. [2 Punkte]

Das Programm gibt u.a. den Maximumsnorm-Abstand zur numerisch exakten Q-value function aus; dieser sollte ca. 0.3 betragen.

(d) Freiwillige Zusatzaufgabe: Experimentieren Sie mit der Spiellogik, indem Sie z.B. die `rewards` innerhalb der Klasse `MazeGhostEnv` ändern, oder die Funktion `_get_ghost_maze_transition_probabilities` in `env.py` so modifizieren, dass sich der Geist auch diagonal bewegen kann.

Tutoraufgabe 12 (Value-Iteration für die Bellman-Gleichung)
Wir untersuchen einen Markov Decision Process (MDP) mit endlichem Zustandsraum $\mathcal{S}$, möglichen Aktionen $\mathcal{A}$, Übergangswahrscheinlichkeit $\mathbb{P}(s_{t+1}|s_t, a_t)$ und Reward-Funktion $R : \mathcal{S} \to \mathbb{R}$. Die Bewertungsfunktion ("cumulative discounted reward") $U^\pi(s_0)$ eines Anfangszustands $s_0 \in \mathcal{S}$ für Policy $\pi : \mathcal{S} \to \mathcal{A}$ ist definiert als

$$U^\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \Big| \pi\right], \quad 0 < \gamma \leq 1,$$

wobei $\mathbb{E}$ den stochastischen Erwartungswert bezeichnet und "bedingt π" der Verhaltensregel $a_t = \pi(s_t)$ des Agenten entspricht. Es sei $U = U^{\pi^*}$ die Bewertungsfunktion der optimalen (von s_0 unabhängigen) Policy π^*:

$$\pi^* = \operatorname*{argmax}_\pi U^\pi(s_0).$$

Die *Bellman-Gleichung* für U besagt

$$U(s) = R(s) + \gamma \cdot \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s,a) U(s').$$

Darauf aufbauend lautet der Value-Iteration-Algorithmus zur iterativen Bestimmung von U:

$$U_{i+1}(s) = R(s) + \gamma \cdot \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s,a) U_i(s') \quad \forall s \in \mathcal{S}.$$

Zeigen Sie für $\gamma < 1$, dass diese Gleichung eine Kontraktion bezüglich der Maximumsnorm

$$\|U\| = \max_{s \in \mathcal{S}} |U(s)|$$

ist und der Algorithmus somit stets zum eindeutigen Fixpunkt konvergiert. Warum ist dieser Algorithmus in der Praxis dennoch häufig nicht einsetzbar?