Intelligent Multi Agent Systems

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer Semester 2016, Homework 3 (65 points + 11 bonus)

G. Neumann, G. Gebhardt, O. Arenz

Due date: Tuesday 28th June, 2016, 23:55 CEST

How to hand in: Besides the Matlab code, the solutions have to be handed in on paper (either in human readable calligraphy or LaTeX) by putting them into the mailbox in front of S2|02 E315. The well commented Matlab code has to be handed in via moodle. Working in groups of up to three people is allowed, however every student needs to submit on its own (both, paper work and Matlab code) and name his collaborators. Late hand-ins will be accepted, however, we will deduct 25% of the achieved points per 24 h or part thereof after the deadline (separately for written exercises and Matlab code.) The department rules regarding plagiarism apply.

Problem 3.1 Theoretical Questions

Keep your answers short! As a rule of thumb: not more than two sentences per point! Explain your answers.

a) **Single Agent Learning** **[5 Points]**

1. What is the purpose of the discount factor?
2. What are the prerequisites for using policy iteration or value iteration?
3. Explain the temporal difference error.
4. Why is exploration important?
5. What is the difference between q-learning and SARSA?

b) **Multi-Agent Learning** **[3 Points]**

1. Describe the curse of dimensionality in relation to multi-agent reinforcement learning.
2. Why might too much exploration lead to an unstable learning process? Think of the minmax algorithm.
3. What is the problem with the estimation of the opponents policies in the joint-action learner, if we do not down-weight older samples?

Problem 3.2 Pen & Paper Exercises: Learning Control Policies for Unknown Worlds

a) **The Bellman Equation for the State-Action Values** **[2 Points]**
Given

- the reward function $R(s,a)$,
- the discount factor γ,
- the probability $P(s'|s,a)$ of making a transition to state s' when taking action a in state s,
- the probability $\pi(a|s)$ of taking action a in state s and
- the finite sets of states S and the finite set of actions A,

write down the Bellman equation for the state-action values $Q^{\pi}(s,a)$. For which $Q^{\pi}(s,a)$ is the Bellman equation true?

Name, Vorname: ______________________ Matrikelnummer: ⌴⌴⌴⌴⌴⌴⌴

b) **Vector-Matrix Notation of the Bellman Equation** **[4 Points]**
Assume that the state-action values and the rewards are stored in the column vectors $\mathbf{q}^\pi \in \mathbb{R}^{|S||A|}$ and $\mathbf{r} \in \mathbb{R}^{|S||A|}$, respectively. So both of these vectors have one entry for each state action pair. The transition probability is given as a matrix $\mathbf{P} \in \mathbb{R}^{|S||A|\times|S|}$, where the entry $\mathbf{P}_{ij}$ gives the probability for ending up in state s_j when being in state s and taking action a from the state action pair $(s,a)_j$. Similarly, the policy is given as a matrix $\mathbf{\Pi}_\pi \in \mathbb{R}^{|S|\times|S||A|}$, where the entry $\mathbf{\Pi}_{ij}$ gives the probability of taking action a in state s_i which is combined in the state action pair $(s,a)_j$. Rewrite the Bellman equation of the state-action values in vector matrix notation (without the sigma notation of the summation).

c) **Towards a Continuous World** **[1 Points]**
So far, states and actions were values of the respective finite sets S and A. However, the world is usually not discrete. Assuming that you have a set of k linearly independent feature functions $\phi_i : S \times A \to \mathbb{R}$, give an approximation $\hat{Q}^\pi$ of the state-action value function as a linear parametric combination of the features.

d) **The Bellman Equation of the Approximate State-Action Value Function** **[2 Points]**
Assuming that we are given a data set of n state-action pairs $\{(s_i, a_i)\}_{i=1}^n$, we can define a feature matrix of the data set as

$$\mathbf{\Phi} = \begin{pmatrix} \phi_1(s_1,a_1) & \dots & \phi_k(s_1,a_1) \\ \vdots & \ddots & \vdots \\ \phi_1(s_n,a_n) & \dots & \phi_k(s_n,a_n) \end{pmatrix} \tag{1}$$

Rewrite the vector-matrix notation of the Belman equation using the results of task c) and the matrix $\mathbf{\Phi}$ (you can assume that the set of n state-action pairs is the cartesian product $S \times A$, however, it could also be a set of random samples). The equation should have the form $\mathbf{\Phi w} = b + C\mathbf{\Phi w}$, where the left-hand side is the approximation of the state-action values and the right-hand side is the Bellman operator applied to the state-action values.

e) **Derivation of the Parameters** **[5 Points]**
Consider your result of task d). From the left-hand side of the equation we see that the approximation $\hat{Q}^\pi$ lies in the space spanned by the feature functions ϕ_i. However, the right-hand side may in general be out of that space. Hence, we want the right-hand side to be projected into the subspace of the feature functions. This can be obtained by the regularized orthogonal projection $\mathbf{\Phi}\left(\mathbf{\Phi}^\intercal\mathbf{\Phi} + \lambda\mathbf{I}\right)^{-1}\mathbf{\Phi}^\intercal$. Apply this projection to the right-hand side of the equation and derive the solution for the parameters w^π. Your solution should in the end have the form $w^\pi = \mathbf{A}^{-1}b$.

f) **Approaching the Algorithm** **[5 Bonus Points]**
Show that $\mathbf{P\Pi}_\pi\mathbf{\Phi} = \mathbf{\Phi}'$, with $\mathbf{\Phi}' = [\phi\left(s_1', \pi\left(s_1'\right)\right), \dots, \phi\left(s_n', \pi\left(s_n'\right)\right)]^\intercal$ where each ϕ is a feature vector $\phi(s,a) = [\phi_1(s,a), \dots, \phi_k(s,a)]^\intercal$. Hint: rewrite the matrix $\mathbf{A}$ as an expectation with respect to the probability $P\left(s'|s,a\right)$. Exploit the following facts:

- the action a' is selected in state s' according to the current policy $\pi(s')$,
- the expectation is a linear operator for functions that do not have the random variables of the expectation as an argument, i.e., $\mathbb{E}_X[g(Y)f(X)] = g(Y)\mathbb{E}_X[f(X)]$ and $\mathbb{E}_X[g(Y)+f(X)] = g(Y)+\mathbb{E}_X[f(X)]$.

g) **Closing the Loop of the Algorithm** **[5 Points]**
So far, we have derived a rule to update the parameters for the state-action value function approximation.

- What is the analog of computing the parameters in standard policy iteration?
- Besides computing the parameters $\mathbf{w}^\pi$, what else do we need to do for LSPI?
- Propose a method for this second step of the algorithm.

Name, Vorname: ______________________ Matrikelnummer: ⊔⊔⊔⊔⊔⊔⊔

Problem 3.3 Matlab Exercises

For the single agent learning methods, we use the following toy task as a benchmark: an agent lives in a square grid world with a width of seven cells. Initially, the agent starts randomly in the southernmost two rows of the world and wants to reach a treasure of golden coins in the northernmost row. Satanically, there are some deathly hollows in world. If the agent slips into on of these abysses, he is doomed to a slow death of starvation and so he wants to avoid them. The agent is able to perform kings moves, which means that he can stay where he is or move to any of the eight neighboring fields, though, the movements are subject to transition noise. The treasure world is depicted in Figure 1.

Leave the parameters of the algorithms at first as they are. Once your implementation gives you nice results, you can play around with the parameters and see how this affects the outcome.

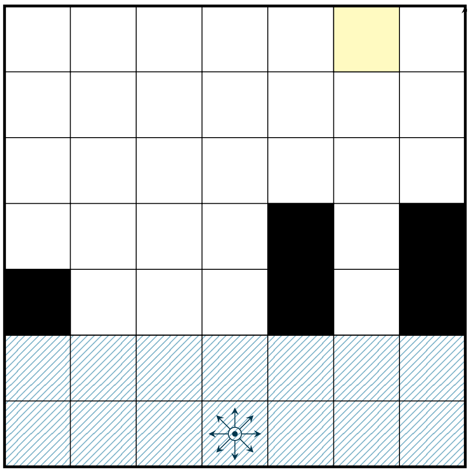

Figure 1: Treasure world with deathly hollows.

a) **Policy Iteration and Value Iteration** **[6 Bonus Points]**
Download the Matlab templates from moodle. Implement the scripts `policy_iteration.m` and `value_iteration.m`. You only get the bonus points, if you use at most two loops for policy iteration and at most one loop for value iteration. These loops are already given in the templates.

b) **Q-Learning and SARSA** **[8 Points]**
Download the Matlab templates from moodle. Implement the scripts `q_learning.m` and `sarsa.m`.

- Plot the learned value functions and policies.
- Plot learning curves for both algorithms for different values for the parameters `gamme` and `alpha`.

c) **Least-Squares Policy Iteration** **[15 Points]**
Download the Matlab templates from moodle. Implement the missing parts in the function `policy` and in the script `lspi.m`.

- Plot the learned value functions and policies.
- Plot learning curves for different values for the parameters `gamma` and `n_episodes`.

d) **The Defense Game** **[15 Points]**

The defense game is a game of two players in a grid world with 5 rows and four columns. One agent, the runner, starts randomly in the bottom row of the grid world and wants to reach the top row. He can move to the left, to the right, one row up or stay where he is. The other agent, the catcher, starts randomly at the second to the top row and wants to prevent the other player from reaching the top row by catching him. However, she can only move to the right, to the left or stay where she is. The actions of both agents are subject to transition noise. The runner gets a reward of 100 when he reaches the top row and the catcher gets a reward of 100 when she catches the runner. The respective other agent gets then always the negative reward. One episode of the game is over, when the runner is either caught or has reached the top row. We want to apply minmax Q-learning to obtain the Q functions for both agents. The grid world of the defense game is depicted in Figure 2.

Download the Matlab templates from moodle. Implement the missing parts in the functions `reward` and `transition` and in the script `defense.m`.

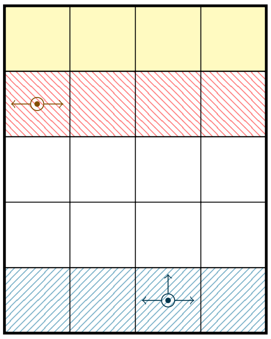

Figure 2: The defense game