TÉCNICO LISBOA

Learning and Decision Making 2017-2018

MSc in Computer Science and Engineering

Second test – June 18, 2018

Instructions

- You have 90 minutes to complete the test.
- Make sure that your test has a total of 8 pages and is not missing any sheets, then write your full name and student n. on this page (and all others if you want to be safe).
- The test has a total of 5 questions, with a maximum score of 20 points. The questions have different levels of difficulty. The point value of each question is provided next to the question number.
- *If you get stuck in a question, move on.* You should start with the easier questions to secure those points, before moving on to the harder questions.
- *No interaction with the faculty is allowed during the exam.* If you are unclear about a question, clearly indicate it and answer to the best of your ability.
- Please provide your answer in the space below each question. If you make a mess, clearly indicate your answer.
- The exam is open book and open notes. You may use a calculator, but any other type of electronic or communication equipment is not allowed.
- Good luck.

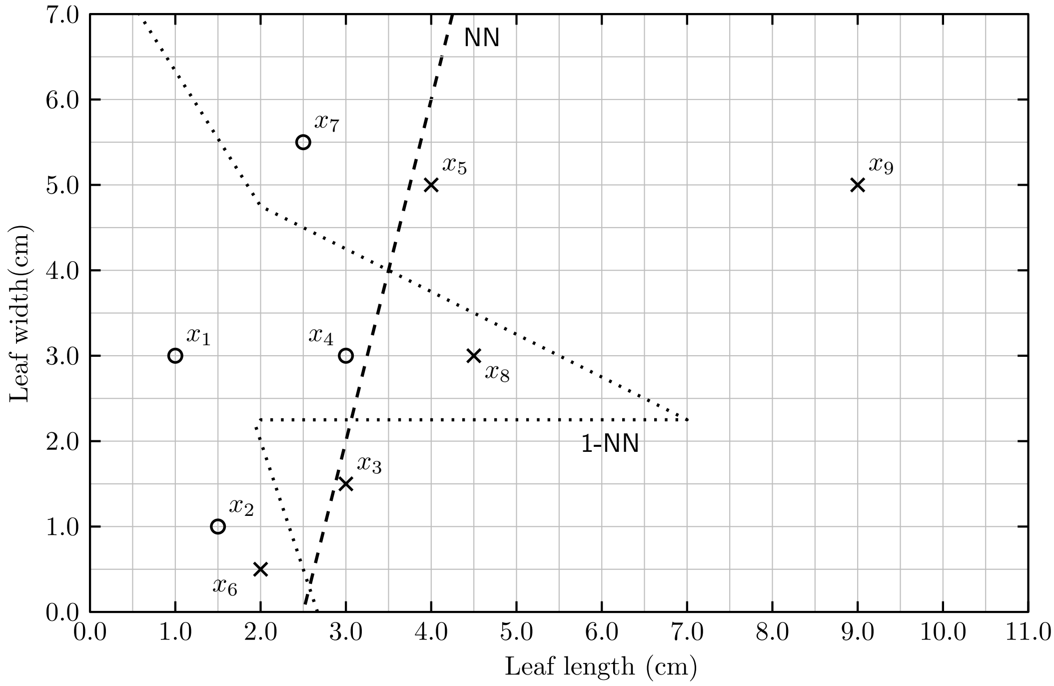
Question 1. **(4 pts.)**

Consider the following data, corresponding to the dimensions of several samples from two species of plants.

	x_1	x_2	x_3	x_4	x_5
Leaf length (cm)	1.0	1.5	3.0	3.0	4.0
Leaf width (cm)	3.0	1.0	1.5	3.0	5.0
Plant species	A	A	B	A	B

Suppose that you want to train a system that automatically classifies plants into one of the two species, A and B, according to the leaf dimensions, and use the data above to train the system.

(a) **(0.5 pts.)** Represent the training data in a scatter plot using the grid below.



(b) **(1.5 pts.)** In the same grid, indicate the decision boundary corresponding to the 1-nearest neighbor (1-NN) classifier.

(c) **(2 pts.)** Consider the test set:

	x_6	x_7	x_8	x_9
Leaf length (cm)	2.0	2.5	4.5	9.0
Leaf width (cm)	0.5	5.5	3.0	5.0
Plant species	B	A	B	B

Compute the accuracy of the 1-NN classifier using the test set above. Explicitly indicate the class that the classifier assigns to each test point. You can use a geometric reasoning.

Solution 1.

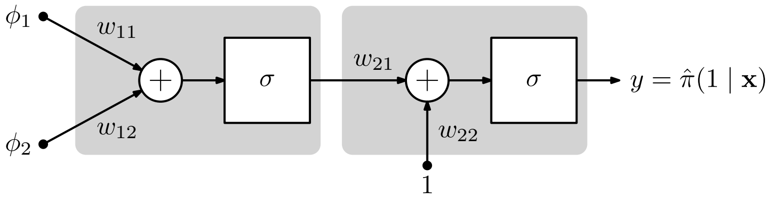
(c) We start by plotting the test points in the same scatter plot (see grid above). From the decision boundary computed in (b), we get

$$\pi(x_6) = A; \qquad \pi(x_7) = B; \qquad \pi(x_8) = A; \qquad \pi(x_9) = B;$$

which corresponds to an accuracy of 25%.

Question 2. (**5 pts.**)

Consider the following neural network, comprising 1 hidden layer with a single unit, and a single output unit.



Each w_{ij} is the weight associated with the jth input of unit i. Both units are sigmoid units with activation function

$$\sigma(u) = \frac{1}{1 + e^{-u}}.$$

Assuming that $\mathcal{A} = \{0, 1\}$, the network outputs the (learned) probability of action 1 given the input, i.e.,

$$y = \hat{\pi}(1 \mid x) \triangleq \mathbb{P}\left[\mathrm{y} = 1 \mid \mathrm{x} = x\right].$$

Suppose that, after training, the network weights take the following values:

$$w_{11} = -0.4; \qquad w_{12} = 0.1; \qquad w_{21} = -0.37; \qquad w_{22} = 0.1.$$

(a) **(2 pts.)** Compute the accuracy of the trained neural network using the *test set* from Question 1. Assume that ϕ_1 is the leaf length and ϕ_2 is the leaf width, while action 0 corresponds to class A and action 1 corresponds to class B. Explicitly indicate the class that the network assigns to each test point, indicating all relevant computations.

(b) **(3 pts.)** Compute the decision boundary for the trained neural network, indicating the relevant computations. Draw the decision boundary in the grid on page 2.

Solution 2.

(a) Using the provided weights, we get

- For x_6:

$$y_6 = \sigma\big(0.1 - 0.37\sigma(-0.4 \times 2 + 0.1 \times 0.5)\big) = \sigma\left(0.1 - \frac{0.37}{1 + e^{+0.75}}\right)$$
$$= \sigma(0.1 - 0.37 \times 0.32) = \frac{1}{1 + e^{+0.0184}} = 0.495 \qquad \text{(class } A\text{);}$$

- For x_7:

$$y_6 = \sigma\big(0.1 - 0.37\sigma(-0.4 \times 2.5 + 0.1 \times 5.5)\big) = \sigma\left(0.1 - \frac{0.37}{1 + e^{+0.45}}\right)$$
$$= \sigma(0.1 - 0.37 \times 0.389) = \frac{1}{1 + e^{+0.0044}} = 0.488 \qquad \text{(class } A\text{);}$$

- For x_8:

$$y_6 = \sigma\big(0.1 - 0.37\sigma(-0.4 \times 4.5 + 0.1 \times 3)\big) = \sigma\left(0.1 - \frac{0.37}{1 + e^{+1.5}}\right)$$
$$= \sigma(0.1 - 0.37 \times 0.182) = \frac{1}{1 + e^{-0.032}} = 0.501 \qquad \text{(class } B\text{);}$$

- For x_9:

$$y_6 = \sigma\big(0.1 - 0.37\sigma(-0.4 \times 9 + 0.1 \times 5)\big) = \sigma\left(0.1 - \frac{0.37}{1 + e^{+3.1}}\right)$$
$$= \sigma(0.1 - 0.37 \times 0.04) = \frac{1}{1 + e^{-0.08}} = 0.52 \qquad \text{(class } B\text{);}$$

which corresponds to an accuracy of 75%.

(b) The decision boundary corresponds to those points where $\hat{\pi}(1 \mid x) = \hat{\pi}(0 \mid x)$ or, equivalently, when $y = 0.5$. Solving this equation with respect to the input features ϕ_1 and ϕ_2 (we omit the dependence on the input x to avoid cluttering the expressions) yields

$$\sigma\big(0.1 - 0.37\sigma(-0.4\phi_1 + 0.1\phi_2) = \frac{1}{1 + e^{-0.1 + 0.37\sigma(-0.4\phi_1 + 0.1\phi_2)}} = 0.5,$$

which is equivalent to

$$0.1 - 0.37\sigma(-0.4\phi_1 + 0.1\phi_2) = 0.$$

Replacing the definition of σ further yields

$$\frac{1}{1 + e^{0.4\phi_1 - 0.1\phi_2}} = 0.27$$

which, after some algebraic manipulation, finally leads to

$$0.4\phi_1 - 0.1\phi_2 = 1 \Leftrightarrow \phi_2 = 4\phi_1 - 10.$$

This corresponds to a linear decision boundary, plotted in the grid of page 2.

Question 3. (6.5 pts.)

Consider a reinforcement learning agent moving in the following environment.

1	2	3
4	5	6

The agent can be described by an MDP $(\mathcal{X}, \mathcal{A}, \{\mathbf{P}_a\}, c, \gamma)$, where $\mathcal{X} = \{1, 2, 3, 4, 5, 6\}$ and $\mathcal{A} = \{d, r\}$. The transition probabilities $\{\mathbf{P}_a, a \in \mathcal{A}\}$ and the cost function c are unknown. Consider $\gamma = 0.9$.

(a) **(2.5 pts.)** After following some fixed policy π for t time steps, the agent's estimate of the function J^π is

$$\boldsymbol{J}^{(t)} = [3.7, 3.4, 2.4, 3.3, 2.4, 1.4]^\top.$$

The estimate above was obtained using TD(λ), with $\lambda = 0.4$ and a step size $\alpha = 0.3$. The corresponding eligibility trace at time step t is

$$\boldsymbol{z}^{(t)} = [0.0, 0.0, 0.0, 1.0, 0.1, 0.4]^\top.$$

The agent then observes

$$x_t = 5; \quad a_t = r; \quad c_t = 1; \quad x_{t+1} = 6; \quad a_{t+1} = d; \quad c_{t+1} = 0; \quad x_{t+2} = 6; \quad a_{t+2} = r.$$

Compute the updated estimates $\boldsymbol{J}^{(t+1)}$ and $\boldsymbol{J}^{(t+2)}$ using the transition data above. Indicate the relevant calculations.

(b) **(1.5 pts.)** Discuss the role of λ in TD(λ) and the potential benefits of considering $\lambda > 0$.

(c) **(2.5 pts.)** Consider once again the situation in (a), but now consider that the agent is using SARSA with linear function approximation, with the following features defined over $\mathcal{X} \times \mathcal{A}$:

$$\phi_1(x,a) = \begin{cases} 1 & \text{if } x \in \{1,2,3\} \\ 0 & \text{otherwise} \end{cases} \qquad \phi_2(x,a) = \begin{cases} 1 & \text{if } a = r \\ 0 & \text{otherwise.} \end{cases}$$

Suppose that, at time step t, the agent's estimate of the function Q is given by

$$Q^{(t)}(x,a) = \boldsymbol{\phi}^\top(x,a)\boldsymbol{w}^{(t)},$$

where

$$\boldsymbol{\phi}(x,a) = [\phi_1(x,a), \phi_2(x,a)]^\top,$$

and

$$\boldsymbol{w}^{(t)} = [2.5, 2.1]^\top.$$

Compute the updated estimates $Q^{(t+1)}$ and $Q^{(t+2)}$ using the transition data in (a), repeated here for convenience. Indicate the relevant calculations.

$$x_t = 5; \quad a_t = r; \quad c_t = 1; \quad x_{t+1} = 6; \quad a_{t+1} = d; \quad c_{t+1} = 0; \quad x_{t+2} = 6; \quad a_{t+2} = r.$$

Solution 3.

(a) From the provided trajectory, we get two samples, $(5,1,6)$ and $(6,0,6)$. Denoting by $\boldsymbol{e}_i$ the ith unit vector (i.e., the vector with all elements equal to 0 except i, which is equal to 1), the first update takes the form

$$\begin{aligned} \boldsymbol{z}^{(t+1)} &= \lambda\gamma\boldsymbol{z}^{(t)} + \boldsymbol{e}_{x_t} \\ \boldsymbol{J}^{(t+1)} &= \boldsymbol{J}^{(t)} + \alpha\boldsymbol{z}^{(t+1)}(c_t + \gamma J^{(t)}(x_{t+1}) - J^{(t)}(x_t)) \end{aligned}$$

yielding

$$\boldsymbol{z}^{(t+1)} = 0.36\boldsymbol{z}^{(t)} + \boldsymbol{e}_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.36 \\ 1.04 \\ 0.14 \end{bmatrix}; \quad \boldsymbol{J}^{(t+1)} = \boldsymbol{J}^{(t)} + 0.3\boldsymbol{z}^{(t+1)}(1 + 0.9 \times 1.4 - 2.4) = \begin{bmatrix} 3.7 \\ 3.4 \\ 2.4 \\ 3.29 \\ 2.35 \\ 1.39 \end{bmatrix}.$$

Similarly, for the second update, we get

$$\boldsymbol{z}^{(t+2)} = 0.36\boldsymbol{z}^{(t+1)} + \boldsymbol{e}_6 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.13 \\ 0.37 \\ 1.05 \end{bmatrix}; \quad \boldsymbol{J}^{(t+2)} = \boldsymbol{J}^{(t+1)} + 0.3\boldsymbol{z}^{(t+2)}(0 + 0.9 \times 1.39 - 1.39) = \begin{bmatrix} 3.7 \\ 3.4 \\ 2.4 \\ 3.28 \\ 2.33 \\ 1.35 \end{bmatrix}.$$

(b) Roughly speaking, the eligibility trace $\boldsymbol{z}^{(t)}$ propagates information from the transition observed at time step t, (x_t, c_t, x_{t+1}) back in time, using this information to update the cost-to-go of states visited before. To put it differently, $\boldsymbol{z}^{(t)}$ assigns "blame" for the cost incurred at time step t, c_t, to the states visited before t.

The scalar λ controls how far in the past such information is propagated. For example, for $\lambda = 0$, the information from time-step t is only used to update the cost-to-go of the state x_t. On the other hand, as $\lambda \to 1$, the information at time step t is used to update the cost-to-go of *all* states visited before time step t.

The use of eligibility traces (with $\lambda > 0$) allows for better use of information, leading to potentially faster convergence.

(c) As before, we get two samples, $(5, r, 1, 6, d)$ and $(6, d, 0, 6, r)$. The first update thus takes the form

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \alpha\boldsymbol{\phi}(x_t, a_t)(c_t + \gamma Q^{(t)}(x_{t+1}, a_{t+1}) - Q^{(t)}(x_t, a_t)).$$

To perform the update above, we start by computing $Q^{(t)}(x_t, a_t)$ and $Q^{(t)}(x_{t+1}, a_{t+1})$. We get

$$Q^{(t)}(x_t, a_t) = \boldsymbol{\phi}^\top(x_t, a_t)\boldsymbol{w}^{(t)} = [0 \quad 1] \begin{bmatrix} 2.5 \\ 2.1 \end{bmatrix} = 2.1;$$

$$Q^{(t)}(x_{t+1}, a_{t+1}) = \boldsymbol{\phi}^\top(x_{t+1}, a_{t+1})\boldsymbol{w}^{(t)} = [0 \quad 0] \begin{bmatrix} 2.5 \\ 2.1 \end{bmatrix} = 0.$$

The SARSA update then comes

$$\boldsymbol{w}^{(t+1)} = \begin{bmatrix} 2.5 \\ 2.1 \end{bmatrix} + 0.3 \begin{bmatrix} 0 \\ 1 \end{bmatrix} (1 + 0.9 \times 0 - 2.1) = \begin{bmatrix} 2.5 \\ 1.77 \end{bmatrix}.$$

Repeating for the second update, we get

$$Q^{(t+1)}(x_{t+1}, a_{t+1}) = \boldsymbol{\phi}^\top(x_{t+1}, a_{t+1})\boldsymbol{w}^{(t+1)} = [0 \quad 0] \begin{bmatrix} 2.5 \\ 1.77 \end{bmatrix} = 0;$$

$$Q^{(t+1)}(x_{t+2}, a_{t+2}) = \boldsymbol{\phi}^\top(x_{t+2}, a_{t+2})\boldsymbol{w}^{(t+2)} = [0 \quad 1] \begin{bmatrix} 2.5 \\ 1.77 \end{bmatrix} = 1.77.$$

The SARSA update then comes

$$\boldsymbol{w}^{(t+2)} = \begin{bmatrix} 2.5 \\ 1.77 \end{bmatrix} + 0.3 \begin{bmatrix} 0 \\ 0 \end{bmatrix} (0 + 0.9 \times 1.77 - 0) = \begin{bmatrix} 2.5 \\ 1.77 \end{bmatrix}.$$

Question 4. (1.5 pts.)

Explain the differences between on-policy and off-policy learning. Provide examples of both on-policy and off-policy reinforcement learning algorithms, pointing out how these differences translate in terms of the corresponding update rules.

Solution 4.

Off-policy methods learn the value of a policy while the agent may be following a different policy. For example, Q-learning is an off-policy method, as is visible in its update rule:

$$Q\text{-learning:} \qquad Q(x_t, a_t) \leftarrow Q(x_t, a_t) + \alpha_t(c_t + \gamma \boxed{\min_{a' \in \mathcal{A}} Q(x_{t+1}, a')} - Q(x_t, a_t))$$

The temporal difference used to update the Q-value is computed from the greedy action at time-step $t+1$, independently of the action actually selected by the agent at that time step.

On-policy methods, on the other hand, learn the value of the policy that the agent is following. For example, SARSA is an on-policy method, as is visible in its update rule:

$$\text{SARSA:} \qquad Q(x_t, a_t) \leftarrow Q(x_t, a_t) + \alpha_t(c_t + \gamma \boxed{Q(x_{t+1}, a_{t+1})} - Q(x_t, a_t))$$

Unlike Q-learning, the temporal difference used to update the Q-value is computed from the actual policy used to sample the actions during learning.

Question 5. (3 pts.)

Consider an agent engaged in a sequential game with "Nature" where, at round t,

- The agent selects an action a_t from the set $\mathcal{A} = \{a, b, c\}$;
- "Nature" selects a cost function $\mathrm{c}_t : \mathcal{A} \to [0, 1]$ according to a predefined (but unknown) distribution;
- The agent executes the action a_t and observes the corresponding cost, $\mathrm{c}_t(\mathrm{a}_t)$.

(a) **(1 pt.)** Is the problem described above a multi-armed bandit problem? If so, what type? If not, why?

(b) **(2 pts.)** Suppose that the cost functions for $t = 1, \ldots, 5$ are

$$c_1 = [0.8, 0.4, 0.8];$$
$$c_2 = [0.7, 0.3, 0.9];$$
$$c_3 = [0.8, 0.2, 0.6];$$
$$c_4 = [0.6, 0.4, 1.0];$$
$$c_5 = [0.7, 0.2, 0.9].$$

Using the adequate algorithm, compute the actions selected by the agent for $t = 1, \ldots, 5$. Indicate the relevant computations.

Solution 5.

(a) The problem described *is* a multi-armed bandit problem. It consists of a sequential prediction problem in which the agent can only observe the cost associated with the action it selected, a distinctive aspect of multi-armed bandit problems. Since the costs are selected according to a predefined distribution, it is a *stochastic* multi-armed bandit problem.

(b) Since it is a stochastic multi-armed bandit problem, we follow the UCB algorithm. According to UCB, the first $|\mathcal{A}|$ steps the agent should select each action exactly once. In subsequent time steps, actions are selected as

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmin}}\, \hat{Q}_t(a) - \sqrt{\frac{2\log t}{N_a}}.$$

We thus get:

t=1: The agent selects $a_1 = a$, leading to $\hat{Q}(a) = 0.8, N_a = 1$;

t=2: The agent selects $a_2 = b$, leading to $\hat{Q}(b) = 0.3, N_b = 1$;

t=3: The agent selects $a_3 = c$, leading to $\hat{Q}(c) = 0.6, N_c = 1$;

t=4: Computing $\hat{Q}(a) - \sqrt{2\log t/N_a}$ for all $a \in \mathcal{A}$ yields

$$\begin{bmatrix} 0.8 & 0.3 & 0.6 \end{bmatrix} - \sqrt{\begin{bmatrix} \frac{2.77}{1} & \frac{2.77}{1} & \frac{2.77}{1} \end{bmatrix}} = \begin{bmatrix} -0.865 & -1.365 & -1.065 \end{bmatrix}.$$

The agent thus selects $a_4 = b$, leading to $\hat{Q}(b) = 0.35, N_b = 2$;

t=5: Once again computing $\hat{Q}(a) - \sqrt{2\log t/N_a}$ for all $a \in \mathcal{A}$ yields

$$\begin{bmatrix} 0.8 & 0.35 & 0.6 \end{bmatrix} - \sqrt{\begin{bmatrix} \frac{3.22}{1} & \frac{3.22}{2} & \frac{3.22}{1} \end{bmatrix}} = \begin{bmatrix} -0.99 & -0.92 & -1.19 \end{bmatrix}.$$

The agent now selects $a_5 = c$.