TÉCNICO LISBOA

Planning, Learning and Decision Making

MSc in Computer Science and Engineering

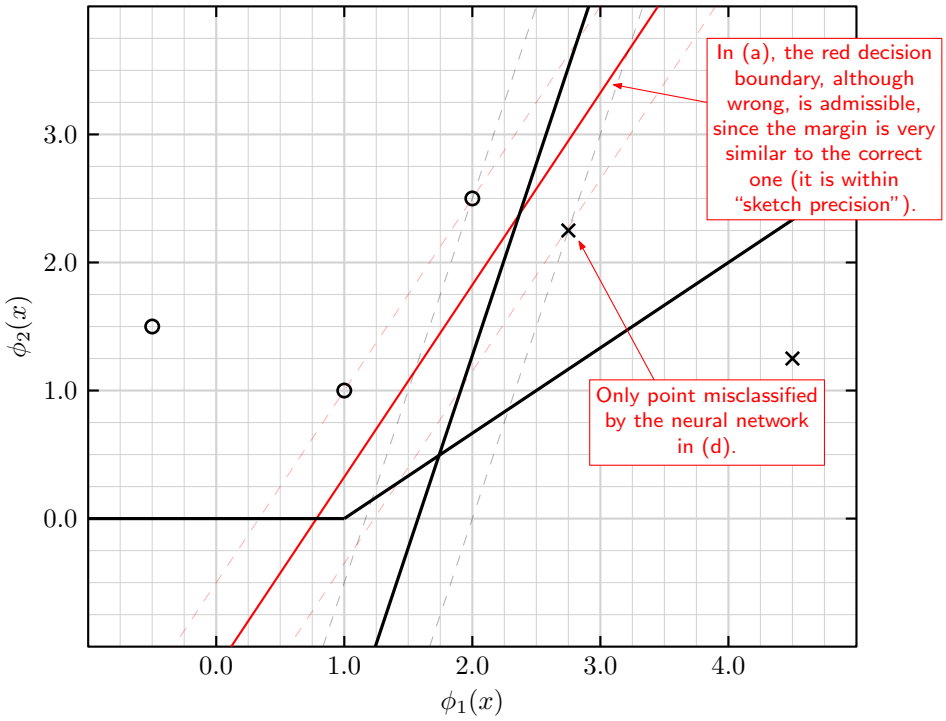
Second test – January 21, 2019

Instructions

- You have 90 minutes to complete the test.
- Make sure that your test has a total of 7 pages and is not missing any sheets, then write your full name and student n. on this page (and your number in all others).
- The test has a total of 3 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.

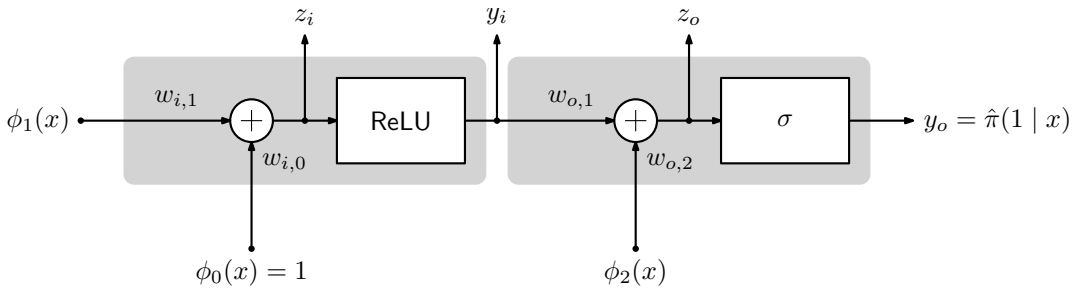
Question 1. (**9 pts.**)

Consider the dataset depicted in the grid below, where the points marked with "×" correspond to class 0 and the points marked with "○" correspond to class 1.



(a) **(1.5 pts.)** Is the dataset above linearly separable? If so, indicate in the grid the decision boundary for the max-margin classifier. If not, explain why.

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



Note that z_i, y_i, and z_o *are not network outputs*, but merely auxiliary variables included for ease of reference.

(b) **(3 pts.)** Suppose that the current weights of the network take the following values:

$$w_{i,0} = -1; \qquad w_{i,1} = 1; \qquad w_{o,1} = -1 \qquad w_{o,2} = 1.3.$$

Compute the weights after a *single iteration* of back-propagation using only the point marked in the grid of the previous page. Use a step size $\alpha = 1$. Indicate the relevant computations.

Recall:

$$\mathsf{ReLU}(z) = \max\{0, z\} \qquad \text{and} \qquad \sigma(z) = \frac{1}{1+e^{-z}}.$$

For the network above, back-propagation updates take the form

$$\begin{aligned} w_{o,\bullet} &\leftarrow w_{o,\bullet} - \alpha\delta_o x_{o,\bullet}; && \text{with} && \delta_o = y_o - a; \\ w_{i,\bullet} &\leftarrow w_{i,\bullet} - \alpha\delta_i x_{i,\bullet}; && \text{with} && \delta_i = \delta_o w_{o,1}\mathbb{I}[z_i > 0], \end{aligned}$$

where a is the desired network output and the different x_u denote the (unit) inputs associated with the different weights w_u.

(c) **(3 pts.)** Compute the decision boundary for the neural network after the update in (b), and plot it in the grid of page 3.

Note: If you did not answer Question (b), use the original weights.

(d) **(1.5 pts.)** Compute the accuracy of the neural network for the data in the grid of page 3. If you solved (c), you can use geometric reasoning.

Solution 1.

(a) The dataset is linearly separable, since it is possible to perfectly separate the two classes using a hyperplane. The decision boundary for the max-margin classifier is depicted in the plot.

(b) The datapoint in the plot verifies $\phi_1(x) = 1$, $\phi_2(x) = 1$ and $a = 1$. We start by performing a forward pass through the network, yielding

$$\begin{aligned} z_i &= w_{i,1}\phi_1(x) + w_{i,0} = 1 - 1 = 0; && y_i = \max\{z_i, 0\} = 0; \\ z_o &= w_{o,1}y_i + w_{o,2}\phi_2(x) = 0 + 1.3 = 1.3; && y_o = \sigma(1.3) = 0.79. \end{aligned}$$

We can now perform back-propagation to compute the gradient updates to the weights using the expressions provided, to get

$$\begin{aligned} \delta_o &= y_o - a = 0.79 - 1 = -0.21 \\ w_{o,1} &= w_{o,1} - \alpha\delta_o y_i = w_{o,1} = -1 \\ w_{o,2} &= w_{o,2} - \alpha\delta_o\phi_2(x) = 1.3 + 0.21 = 1.51 \\ \delta_i &= \delta_o w_{o,1}\mathbb{I}[z_i > 0] = 0 \\ w_{i,0} &= w_{i,0} - \alpha\delta_i y_i = w_{o,1} = -1 \\ w_{i,1} &= w_{i,1} - \alpha\delta_i\phi_2(x) = 1. \end{aligned}$$

(c) To compute the decision boundary, we determine the set of points such that $\pi(1 \mid x) = y_o = 0.5$. Since

$$y_o = \sigma(z_o) = 0.5,$$

The decision boundary corresponds to the set of points where $z_o = 0$ which, in turn, corresponds to

$$w_{o,1}y_i + w_{o,2}\phi_2(x) = 0.$$

On the other hand,

$$y_i = \max\{0, z_i\},$$

with $z_i = w_{i,1}\phi_1(x) + w_{i,0}$. Putting everything together,

$$w_{o,1} \max\{0, w_{i,1}\phi_1(x) + w_{i,0}\} + w_{o,2}\phi_2(x) = 0$$

or, equivalently,

$$\phi_2(x) = -\frac{w_{o,1}}{w_{o,2}} \max\{0, w_{i,1}\phi_1(x) + w_{i,0}\}.$$

Replacing the weight values, we finally get

$$\phi_2(x) = \frac{1}{1.51} \max\{0, \phi_1(x) - 1\} = \max\{0, 0.66\phi_1(x) - 0.66\},$$

which corresponds to the piecewise linear decision boundary depicted in the grid of page 3.

(d) Given the decision boundary determined in (c), we can immediate observe that the neural network correctly classifies all points except the one signaled in the grid. This means that the network has an accuracy of $\frac{4}{5} = 80\%$.

Alternatively, we can compute the output for each point by running them through the network. We get:

$(\phi_1(x), \phi_2(x)) = (-0.5, 1.5)$

$z_i = 1 \times (-0.5) - 1 = -1.5;$ $y_i = \max\{0, -1.5\} = 0;$

$z_o = -0 + 1.51 \times 1.5 = 2.27;$ $y_o = \sigma(2.27) = 0.906;$ (class 1)

$(\phi_1(x), \phi_2(x)) = (1.0, 1.0)$

$z_i = 1 \times (1.0) - 1 = 0;$ $y_i = \max\{0, 0\} = 0;$

$z_o = -0 + 1.51 \times 1.0 = 1.51;$ $y_o = \sigma(2.51) = 0.819;$ (class 1)

$(\phi_1(x), \phi_2(x)) = (2.0, 2.5)$

$z_i = 1 \times (2.0) - 1 = 1.0;$ $y_i = \max\{0, 1.0\} = 1.0;$

$z_o = -1.0 + 1.51 \times 2.5 = 2.775;$ $y_o = \sigma(2.775) = 0.942;$ (class 1)

$(\phi_1(x), \phi_2(x)) = (2.75, 2.25)$

$z_i = 1 \times (2.75) - 1 = 1.75;$ $y_i = \max\{0, 1.75\} = 1.75;$

$z_o = -1.75 + 1.51 \times 2.25 = 1.64;$ $y_o = \sigma(1.64) = 0.838;$ (class 1)

$(\phi_1(x), \phi_2(x)) = (4.5, 1.25)$

$z_i = 1 \times (4.5) - 1 = 3.5;$ $y_i = \max\{0, 3.5\} = 3.5;$

$z_o = -3.5 + 1.51 \times 1.25 = -1.61;$ $y_o = \sigma(-1.61) = 0.166.$ (class 0)

Question 2. (**8 pts.**)

Consider the MDP $(\mathcal{X}, \mathcal{A}, \{\mathbf{P}_a\}, c, \gamma)$, where $\mathcal{X} = \{1, 2, 3\}$ and $\mathcal{A} = \{a, b, c\}$. Admit throughout that $\gamma = 0.9$. Suppose that a reinforcement learning agent is running a model based algorithm to

compute Q^*. After t steps,

$$\mathbf{P}_a^{(t)} = \begin{bmatrix} 0.47 & 0.53 & 0 \\ 0.88 & 0.12 & 0 \\ 0.0 & 1.0 & 0 \end{bmatrix}; \qquad \mathbf{P}_b^{(t)} = \begin{bmatrix} 0.38 & 0.62 & 0 \\ 0.38 & 0.62 & 0 \\ 0.67 & 0.33 & 0 \end{bmatrix}; \qquad \mathbf{P}_c^{(t)} = \begin{bmatrix} 0.53 & 0.47 & 0.0 \\ 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix};$$

$$\boldsymbol{c}^{(t)} = \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}; \qquad \boldsymbol{N}_t = \begin{bmatrix} 15 & 13 & 15 \\ 17 & 16 & 10 \\ 6 & 3 & 5 \end{bmatrix}; \qquad \boldsymbol{Q}^{(t)} = \begin{bmatrix} 2.5 & 2.4 & 2.6 \\ 3.0 & 2.4 & 1.0 \\ 0.9 & 1.7 & 0.0 \end{bmatrix},$$

where N_t is the number of visits to each state-action pair. At time step t, the agent experienced the transition $(x_t, a_t, c_t, x_{t+1}) = (1, b, 1, 1)$.

(a) **(1.5 pts.)** Determine the greedy policy at time step t. In light your answer, explain whether the action b selected at time step t is an exploration or exploitation action.

(b) **(3 pts.)** Perform one update of the learning algorithm using the transition provided above. Indicate the relevant computations.

(c) **(2.5 pts.)** Dyna-Q is a well established approach in RL where the model being learned is used at each step to generate "extra" synthetic samples, which are then used to perform additional Q-learning updates.

Suppose then that, following the Dyna-Q approach, you use the model from Question (b) to generate a synthetic transition from state 2 using the greedy policy. Indicate what such transition would be like and perform the corresponding Q-learning update over the Q-values from Question (b). Use $\alpha = 0.1$.

Note: If you did not answer Question (b), use the original values.

(d) **(1 pts.)** Do you think that the Dyna-Q approach introduced in Question (c) is a good idea? Briefly explain why, highlighting potential advantages and disadvantages.

Solution 2.

(a) Given $Q^{(t)}$, the greedy policy is any policy that selects, for each state $x \in \mathcal{X}$, an action that minimizes $Q^{(t)}(x, \cdot)$. This yields

$$\pi_g^{Q^{(t)}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Since the action b selected at time t is greedy w.r.t. $Q^{(t)}$ it is an exploitation action.

(b) Using the transition $(x_t, a_t, c_t, x_{t+1}) = (1, b, 1, 1)$, we get

$$\begin{aligned} N_{t+1}(1, b) &= N_t(1, b) + 1 = 14; \\ \mathbf{P}_a^{(t+1)} &= \mathbf{P}_a^{(t)}; \\ \mathbf{P}_c^{(t+1)} &= \mathbf{P}_c^{(t)}; \\ \mathbf{P}^{(t+1)}(1 \mid 1, b) &= \left(1 - \frac{1}{N_{t+1}(1, b)}\right) \mathbf{P}^{(t)}(1 \mid 1, b) + \frac{1}{N_{t+1}(1, b)} = 0.42; \end{aligned}$$

$$\mathbf{P}^{(t+1)}(2 \mid 1, b) = \left(1 - \frac{1}{N_{t+1}(1,b)}\right) \mathbf{P}^{(t)}(2 \mid 1, b)) = 0.58;$$
$$\mathbf{P}^{(t+1)}(3 \mid 1, b) = \left(1 - \frac{1}{N_{t+1}(3,b)}\right) \mathbf{P}^{(t)}(3 \mid 1, b)) = 0.0;$$
$$c^{(t+1)}(1, b) = \left(1 - \frac{1}{N_{t+1}(1,b)}\right) c^{(t)}(1, b) + \frac{1}{N_{t+1}(1,b)} = 1.0$$

and, finally,

$$\begin{aligned} Q^{(t+1)}(1, b) &= c^{(t+1)}(1, b) + \gamma \mathbf{P}^{(t+1)}(1, b) \min \boldsymbol{Q}^{(t)} \\ &= 1 + 0.9 \begin{bmatrix} 0.42 & 0.58 & 0 \end{bmatrix} \begin{bmatrix} 2.4 \\ 1.0 \\ 0.0 \end{bmatrix} = 2.43. \end{aligned}$$

(c) Using the updated $Q^{(t+1)}$, the greedy action in state 2 is action c. According to our learned model, performing action c in state 2 has a cost of 1 and leads the agent to state 3 with probability 1. Therefore, we get the synthetic transition $(x_{t+1}, a_{t+1}, c_{t+1}, x_{t+2}) = (2, c, 1, 3)$. We can now perform a Q-learning update

$$Q^{(t+2)}(x_{t+1}, a_{t+1}) = Q^{(t+1)}(x_{t+1}, a_{t+1}) + \alpha(c_{t+1} + \gamma \min_{a \in \mathcal{A}} Q^{(t+1)}(x_{t+1}, a) - Q^{(t+1)}(x_{t+1}, a_{t+1})),$$

yielding

$$\begin{aligned} Q^{(t+2)}(2, c) &= Q^{(t+1)}(2, c) + \alpha(1 + \gamma \min_{a \in \mathcal{A}} Q^{(t+1)}(3, a) - Q^{(t+1)}(2, c)) \\ &= 1 + 0.1(1 + 0.9 \times 0 - 1) = 1. \end{aligned}$$

(d) In general, it is a good idea. The main advantage of using Dyna-Q is that, at each iteration, it generates data that is "aligned" with the experience of the agent so far to perform additional iterations, propagating the information in the model more efficiently. It is very similar to performing multiple DP updates to Q per iteration. The main disadvantage lies in the added computational cost per update.

Question 3. (**3 pts.**)

Explain the main differences between UCB and EXP3, indicating in which situations each of the two algorithms is more adequate.

Solution 3.

Both algorithms are designed for multi-armed bandit problems, in which the agent is engaged in a sequential game with "Nature" and, at each step t,

- The agent selects an action a_t;
- Simultaneously, Nature selects a cost function $c_t : \mathcal{A} \to \mathbb{R}$
- The agent plays its action and observes (and pays) the cost $c_t(a_t)$.

In other words, the agent has no knowledge beforehand of the cost function used to evaluate its actions and, upon playing its action, gets to observe only the cost for that action.

UCB is designed for *stochastic bandit problems*, where the cost c_t selected by Nature is drawn from some fixed—albeit unknown—distribution. For this reason, UCB's action selection heuristic is deterministic, and yields a regret $\mathrm{O}(\sqrt{NT \log T})$.

In contrast, EXP3 is designed for *adversarial bandit problems*, where the assumption that the costs are drawn from some distribution does not hold. Adversarial bandit problems assume that the costs c_t may be selected adversarially. Therefore, in order for the agent to avoid being exploited by Nature, the action selection heuristic in EXP3 is stochastic. It yields a regret $\mathrm{O}(\sqrt{TN \log N})$.