732A96/TDDE15 ADVANCED MACHINE LEARNING

EXAM 2020-10-27

Teacher

Jose M. Peña. Available by Teams (or by e-mail if Teams does not work).

Grades

- For 732A96 (A-E means pass):
  - A=19-20 points
  - B=17-18 points
  - C=14-16 points
  - D=12-13 points
  - E=10-11 points
  - F=0-9 points
- For TDDE15 (3-5 means pass):
  - 5=18-20 points
  - 4=14-17 points
  - 3=10-13 points
  - U=0-9 points

The total number of points is rounded to the nearest integer. In each question, full points requires clear and well motivated answers and/or commented code.

Instructions

This is an individual exam. No help from others is allowed. No communication with others regarding the exam is allowed. Answers to the exam questions may be sent to Urkund.

The answers to the exam should be submitted in a single PDF file using LISAM. You can make a PDF from LibreOffice (similar to Microsoft Word). You can also use Markdown from RStudio. Include important code needed to grade the exam (inline or at the end of the PDF file). The exam is anonymous, i.e. do not write your name anywhere.

The exam consists of three exercises. To encourage you trying to solve every exercise, the exercises will be made available at different time points so that you allocate some time to all of them. The first exercise will be release at the beginning of the exam. The second one hour later. The third two hours later.

Allowed help

Everything in the course webpage. Your individual and group solutions to the lab.

1. Graphical Models (6 p)

(1) (1 p) Consider the following directed and acyclic graph:

$$\begin{tikzcd}
& C \arrow[ddl] \arrow[d] \arrow[ddr] & \\
& D & \\
A \arrow[rr] & & Y
\end{tikzcd}$$

Use `bnlearn` to implement it. Assume that all the random variables are binary.

(2) (5 p) You are asked to randomly parameterize 1000 times the graph constructed in the previous exercise. Each random parameterization should be obtained by drawing the parameters values for each conditional probability table from a uniform distribution. For each random parameterization, you should check whether $p(y|a,c)$ and $p(y|a,d)$ are monotone in C and D, respectively. See the definitions below. You are asked to report how many of the 1000 random parameterizations result in (i) $p(y|a,c)$ is monotone in C but $p(y|a,d)$ is not monotone in D, and (ii) $p(y|a,d)$ is monotone in D but $p(y|a,c)$ is not monotone in C. To solve this exercise, you must use `gRain`. You may want to use the function `as.grain` to convert the `bnlearn` object from the previous exercise into a `gRain` object. Before that, you may want to use the function `custom.fit` to parametrize the object. **Include your code commented**.

We say that $p(y|a,c)$ is nondecreasing in C if

$$p(Y=1|A=1,C=1) \geq p(Y=1|A=1,C=0) \text{ and } p(Y=1|A=0,C=1) \geq p(Y=1|A=0,C=0).$$

Likewise, $p(y|a,c)$ is nonincreasing in C if

$$p(Y=1|A=1,C=1) \leq p(Y=1|A=1,C=0) \text{ and } p(Y=1|A=0,C=1) \leq p(Y=1|A=0,C=0).$$

Moreover, $p(y|a,c)$ is monotone in C if it is nondecreasing or nonincreasing in C. We define similarly that $p(y|a,d)$ is monotone in D. Intuitively, $p(y|a,c)$ is monotone in C if the effect on Y of observing C is in the same direction among those that received the treatment A ($A=1$) and those that did not receive it ($A=0$). Likewise for D.

2. Reinforcement Learning (7 p)

- (3 p) The Q-learning algorithm is based on the following updating rule:

$$q(s,a) \leftarrow q_*(s,a) + \alpha(r + \gamma \max_{a'} q(s',a') - q(s,a))$$

  which implies updating $q(s,a)$ to make it closer to $r+\gamma \max_{a'} q(s',a')$. Some may object that this does not make sense since it assumes that the agent acts greedily in the next state s' due to $\max_{a'}$, but in reality the agent acts ϵ-greedily. Then, they may argue that it would be better to use the following updating rule:

$$q(s,a) \leftarrow q_*(s,a) + \alpha(r + \gamma q(s',a') - q(s,a))$$

  which implies updating $q(s,a)$ as a function of the next state s' and **next action** a'. In other words, one has to produce the next state and action in order to update the current state and action. You are asked to implement this algorithm. **Include your code commented**.

- (2 p) You are asked to run Q-learning and the new algorithm on a modified version of what we called environment C in the lab. The modified version differs from the original in that the reward for positions (1,2:5) is now -10 instead of -1, the reward for position (1,6) is still +10, and the reward for the rest of the positions is -1. **An episode ends now when the agent receives a reward of +10 och -10**, i.e. the episode does not end when the agent receives a reward of -1. Moreover, now $\epsilon = 0.5$, $\gamma = 1$, $\beta = 0$ and $\alpha = 0.1$. The rest of the environment is the same as the original environment C. You are asked to run Q-learning and the new algorithm for 5000 episodes and report the final q-table and policy. You should also plot the reward obtained in each episode (episode in the X-axis and reward in the Y-axis) for both algorithms. Use the function `MovingAverage` provided in the lab. Which algorithm performs best and why ?

- (2 p) The 5000 episodes in the previous question were meant to **train** the agent, i.e. the agent learned a q-table to produce a policy. Now, we want to **test** the agent, i.e. the training phase is over and thus neither the q-table nor the policy are updated any more. Run 5000 test episodes. Plot again the reward obtained in each episode. Which algorithm performs best in the test phase and why ?

3. Gaussian Processes (7 p)

(1) (4 p = 1 + 1 + 1 + 1) In the lab 4 exercise 1, you computed the posterior distribution of f using the five data points in the table below with $\sigma_f = 1$ and $\ell = 0.3$ and $\sigma_n = 0.1$.

x	-1.0	-0.6	-0.2	0.4	0.8
y	0.768	-0.044	-0.940	0.719	-0.664

Now, you are asked to compute the posterior distribution of f using the five data points in the table above with $\sigma_f = 1$ and $\ell = 0.3$ and $\boldsymbol{\sigma_n = 0}$. Then, complete the following tasks:

- Plot the **posterior** covariance of f for $x \in [-1, 1]$ and $x' = 0$. **Include your code commented**.
- Why is the **posterior** covariance zero at the training points ?
- Why does the **posterior** covariance not decrease as a function of the distance between x and x' ?
- Draw or plot the **prior** covariance of f for $x \in [-1, 1]$ and $x' = 0$. You can draw it by hand or write R code to plot it.

Help: Recall that Algorithm 2.1 from the book by Rasmussen and Williams already returns the posterior covariance matrix of f.

(2) (3 p) In the lab 4 exercise 2, you predicted the temperature in Tullinge as function of time $\sigma_f = 20$ and $\ell = 0.2$, and where σ_n^2 was the residual variance from a simple quadratic regression fit.

Now, you are asked to search for the best σ_n value by maximizing the log marginal likelihood. Use a grid search to search in the interval [0.1, 10] (i.e., try different hyperparameter values while time permits) or use the `optim` function from R. **Include your code commented**.

Help: Check the slides on hyperparameter search. Recall that Algorithm 2.1 from the book by Rasmussen and Williams already returns the log marginal likelihood. Predict temperature from time, scale the data, and keep $\sigma_f = 20$ and $\ell = 0.2$.