732A96/TDDE15 ADVANCED MACHINE LEARNING

EXAM 2019-10-23

Teacher

Jose M. Peña. Will visit the room. Available by phone too.

Grades

- For 732A96 (A-E means pass):
  - A=19-20 points
  - B=17-18 points
  - C=12-16 points
  - D=10-11 points
  - E=8-9 points
  - F=0-7 points
- For TDDE15 (3-5 means pass):
  - 5=18-20 points
  - 4=12-17 points
  - 3=8-11 points
  - U=0-7 points

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

Allowed Material

Hard copy of Bishop's book, and the content of the folder given_files in the exam system.

Instructions

The answers to the exam should be submitted in a single PDF file using the communication client. 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). Submission starts by clicking the button "Skicka in uppgift" in the communication client. Then, follow the instructions. Note that the system will let you know that the exam has been submitted, but will not tell you that it was received. This is OK and your solution has actually been received.

Do not ask question through the communication client. The teacher will be reachable by phone, and he will visit the room too.

1. Graphical Models (6 p)

(1) (3 p) You are asked to solve the Monty Hall problem with the help of a BN. Wikipedia describes the problem as follows:

"Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host [Monty Hall], who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?"

To solve this problem with the help of a BN, you should consider three random variables (C, D and M) with three states each: C represents the door containing the car, D represents your door choice, and M represents Monty Hall's door choice. Monty Hall never opens your door or the door containing the car. If he can open several doors, then he chooses one at random.

First, you are asked to build a BN (both structure and parameters) by hand to model the problem description above. You may want to use the functions `model2network` and `custom.fit` from the `bnlearn` package. Second, you are asked to perform exact inference using the `gRain` package, or approximate inference using the `cpdist` function from the `bnlearn` package. Specifically, you are asked to compute $p(C)$, which represents where the car is a priori. You should now choose the most promising door according to this distribution. Say that you choose the first door, i.e. $D = d_1$. Then, you should compute $p(C|D = d_1, M = d_2)$, which represents where the car is a posteriori, i.e. after Monty Hall decides to open the second door. In the light of this a posteriori distribution, you should reconsider whether you stick to $D = d_1$ or change to $D = d_3$. Finally, you should do the same for $p(C|D = d_1, M = d_3)$.

(2) (3 p) You are asked to model the exclusive OR (XOR) gate as a BN. That is, consider three binary random variables A, B and C. Let A and B be uniformly distributed. Let $C = XOR(A, B)$, i.e. $C = 1$ if and only if ($A = 1$ and $B = 0$) or ($A = 0$ and $B = 1$).

First, you are asked to build a BN (both structure and parameters) by hand to model the XOR gate. You may want to use the functions `model2network` and `custom.fit` from the `bnlearn` package. Second, you are asked to sample 1000 instances from the joint probability distribution represented by the BN built. You may want to use the function `rbn` from the `bnlearn` package. Third, you are asked to learn a BN structure from the instances sampled by running the hill-climbing algorithm. You are **not** allowed to use restarts. Finally, you are asked to repeat the learning process 10 times with different samples of the joint distribution and answer the following question: Given that the problem at hand is rather easy (i.e., many observations and few variables), why does the hill-climbing algorithm fail to recover the true BN structure in most runs ?

2. Hidden Markov Models (3 p)

You are asked to extend the HMM built in Lab 2 as follows. The observed random variable has now 11 possible values, corresponding to the 10 sectors of the ring plus an 11th value to represent that the tracking device is malfunctioning. If the robot is in the sector i, then the device will report that it is malfunctioning with probability 0.5 and that the robot is in the sector interval $[i$ - 2, i + 2] with probability 0.1 each. Implement the extension just described by using the `HMM` package. Moreover, consider the observations 1, 11, 11, 11, i.e. the tracking device reports sector 1 first, and then malfunctioning for three time steps. Compute the most probable path using the smoothed distribution and the Viterbi algorithm.

3. State Space Models (5 p)

You are asked to implement the Kalman filter and compare it with the particle filter that you implemented in Lab 3. You should implement the Kalman filter as it appears in the course slides or in the book by Thrun et al. Use $A_t = B_t = C_t = u_t = 1$, $\mu_0 = 50$ and $\Sigma_0 = 100^2/12$ which are the expected value and variance of a $Uniform(0, 100)$ distribution, which is the initial model in Lab 3. Note that the Kalman filter is described in the slides and the book in terms of variances of the transition and emission models, whereas you may have used standard deviations in your particle filter. Note also that the Kalmar filter in the slides and the book uses x_t for the hidden states and z_t for the observations, whereas your particle filter may use the same notation or the inverse notation depending on which version of it you implemented.

Finally, run the particle filter and the Kalman filter for the SSM in Lab 3, except that now the standard deviation of each component in the transition and emission models should be equal to 0.1 and 1, respectively. For the Kalman filter, use also 0.1 and 1 for the standard deviation of the transition and emission model, respectively. Report the average error for the last 10 iterations. Repeat this 10 times, sampling the SSM each time. Answer to the following question: Which method performs best and why ?

4. Gaussian Processes (6 p)

(1) (3 p) You are asked to extend your work on Lab 4 with the Tullinge temperatures. In the lab, you predicted the temperature as function of time with the following hyperparameter values: $\sigma_f = 20$ and $\ell = 0.2$. Now, you are asked to search for the best hyperparameter values by maximizing the log marginal likelihood. You may want to check the corresponding slides for the theoretical details. Recall that you implemented Algorithm 2.1 in the book by Rasmussen and Williams, which already returns the log marginal likelihood.

Your search for the best hyperparameter values may be a grid search (i.e., you try different values while time permits) or you may use the function `optim` as follows:

```
optim(par = c(1,0.1),
fn = LM, X=scale(time),y=scale(temp),k=SEKernel,sigmaNoise=sigmaNoiseFit,
method="L-BFGS-B",
lower = c(.Machine$double.eps, .Machine$double.eps),
control=list(fnscale=-1))
```

where `par` are the initial hyperparameter values, `fn` is the function to maximize followed by its input parameters, `method` is the search method, `lower` indicates that the parameters must be positive, and `control` indicates that the goal is maximization. You do not need to modify the last three lines in the call above.

(2) (3 p) You are asked to extend your work on Lab 4 with the banknote fraud data. In the lab, you used the default squared exponential kernel (a.k.a. radial basis function) with automatic hyperparameter value determination. Now, you are asked to search for the best hyperparameter value by using a validation dataset. Use the four covariates to classify. You may use a grid search or the function `optim`. In the latter case, use `par = c(0.1)` and `lower = c(.Machine$double.eps)`.