Assignment 2: Expectations, Decision Trees, Linear Regression

Due Oct 18 at 11:59pm
40 marks total on programming part
(30 marks for theoretical part, not graded)

This assignment is to be done individually.

Important Note: The university policy on academic dishonesty (cheating) will be taken very seriously in this course. You may not provide or use any solution, in whole or in part, to or by another student.

You are encouraged to discuss the concepts involved in the questions with other students. If you are in doubt as to what constitutes acceptable discussion, please ask! Further, please take advantage of office hours offered by the instructor and the TA if you are having difficulties with this assignment.

DO NOT:

- Give/receive code or proofs to/from other students
- Use Google to find solutions for assignment

DO:

- Meet with other students to discuss assignment (it is best not to take any notes during such meetings, and to re-work assignment on your own)
- Use online resources (e.g. Wikipedia) to understand the concepts needed to solve the assignment

Submitting Your Assignment

You must create your assignment in electronic form in **PDF format**. Submit your assignment using the *new* online assignment submission server at: `https://courses.cs.sfu.ca`. **Check early that you are set up for this, especially that you can log in to the server.**

You should create a report with the answers to questions and figures described above. Make sure it is clear what is shown in each figure. **DO NOT INCLUDE SOURCE CODE.**

Question 1 (5 marks)

Any function $f(X)$ of a discrete random variable X defines a random variable. (Define $p(f(X) = y) = \sum_{x:f(x)=y} p(x)$.) Similarly, given a joint probability $p(X_1, \ldots, X_n)$, any function $f(X_1, \ldots, X_n)$ is also a random variable. Show that the expected value of the sum of three random variables is the sum of the expectations. In symbols, show that

$$E[X_1 + X_2 + X_3] = E[X_1] + E[X_2] + E[X_3].$$

Question 2 (5 marks)

Exercise 1.5 in PRML.

Question 3 (5 marks)

Exercise 1.6 in PRML.

Question 4 (10 marks)

Show that the minimizer for least-squares linear regression with L_2 regularization is $\boldsymbol{w} = \left(\lambda \boldsymbol{I} + \boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{t}$, as in Eqn. 3.28 in PRML.

Question 5 (10 marks)

Figure 1 provides data about whether a customer will wait for a table in a restaurant or not. Assume that ID3 splits first on the Pat attribute (for Patrons). Show the following for the branch $Pat = Full$.

1. The next attribute chosen by ID3.
2. The expected information gain associated with the next attribute. Compare this with the expected information gain for Hungry.
3. How you calculated the expected information gain.

Question 6 (40 marks)

In this question you will implement linear basis function regression with polynomial and Gaussian bases.

Start by downloading the code and dataset from the website, provided under Assignment 1. The dataset is the `AutoMPG` dataset from the UCI repository. The task is to predict fuel efficiency (miles per gallon) from 7 features describing a car.

Functions are provided for loading the data[1], and normalizing the features and targets to have 0 mean and unit variance.

[1]Note that loadData reorders the datapoints using a fixed permutation. Use this fixed permutation for the questions in this assignment. If you are interested in what happens in "reality", try using a random permutation afterwards. Results will not always be as clean as you will get with the fixed permutation provided.

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

Figure 1: Data Set for Question 5, Decision Tree Learning

```
[t,X] = loadData();
X_n = normalizeData(X);
t = normalizeData(t);
```

For the following, use these normalized features `X_n` and targets.

You may also find the provided function `designMatrix.m` useful.

Polynomial basis functions

Implement linear basis function regression with polynomial basis functions. Perform the following experiments:

1. Using the first 100 points as training data, and the remainder as test data, fit a polynomial basis function regression for degree 1 to degree 10 polynomials. Do not use any regularization. Plot training error and test error (in RMS error) versus polynomial degree. **Put this plot, along with a brief comment on what you see, in your report.** For the basis functions:

   (a) Include the bias function as always.

   (b) For each input variable $x_i, i = 1, \ldots, 7$, and for each power $k = 1, \ldots, \mathit{degree}$, there is a basis function $\mathbf{\Phi}_{i,k}$ that returns x_i^k. So you should have $7 \cdot \mathit{degree}$ basis functions, plus the bias function, and hence that many weights. In words, treat each input variable as separate single variable and then follow the treatment in the

book and use the single-variable powers up to the degree. The degree (maximum power) should run from 1 to 10.

(c) If you encounter difficulties with singularity computing the pseudo-inverse, here are two suggestions for what you can try.

i. The creative approach: Use the fact that the regularized pseudo-inverse with the λ parameter is guaranteed to be singular.

ii. The uncreative approach: Check the Matlab help for "pseudo-inverse".

2. It is difficult to visualize the results of high-dimensional regression. Instead, only use one of the features (use `X_n(:,3)`) and again perform polynomial regression. Produce plots of the training data points, learned polynomial, and test data points. The code `visualize_1d.m` may be useful. **Put 2 or 3 of these plots, for interesting (low-order, high-order) results, in your report. Include brief comments.**

3. Implement L_2-regularized regression. Again, use the first 100 points, and only use the 3rd feature. Fit a degree 8 polynomial using $\lambda = \{0, 0.01, 0.1, 1, 10, 100, 1000\}$. Use 10-fold cross-validation to decide on the best value for λ. Produce a plot whose x-axis is the regularizer value, and that shows for each regularizer value (a) the error on the test data and (b) the error on the 100 training data points computed directly and (c) the error on the 100 training data points as estimated by cross-validation. Use a `semilogx` plot, putting regularizer value on a log scale. **Put this plot in your report, and note which regularizer value you would choose from the cross-validation.**

Gaussian basis functions

Implement linear basis function regression with Gaussian basis functions. You may use the supplied `dist2.m` function. For the centers $\boldsymbol{\mu}_j$ use randomly chosen training data points (use `randperm` in MATLAB). Set $s = 2$. Perform the following experiments:

1. Using the first 100 points as training data, and the remainder as test data, fit a Gaussian basis function regression using 5, 15, 25, ..., 95 basis functions. Do not use any regularization. Plot training error and test error (in RMS error) versus number of basis functions. **Put this plot, along with a brief comment on what you see, in your report.**

2. Implement L_2-regularized regression. Again, use the first 100 points (do **not** only use the 3rd feature, use them all). Fit a regression model with 90 basis functions using $\lambda = \{0, 0.01, 0.1, 1, 10, 100, 1000\}$. Use 10-fold cross-validation to decide on the best value for λ. Produce a plot of test set error versus regularizer value. Use a `semilogx` plot, putting regularizer value on a log scale. **Put this plot in your report, and note which regularizer value you would choose from the cross-validation.**