ECS607/766 Data Mining 2017/18

Lab 2: Regression

NOTES:

- PLAGARISING your neighbours' answers is not acceptable.
- About Grading:
  - The total assessed lab work is worth 30% of your final grade.
  - There will be 5 assignments over 10 labs/weeks, two labs per assignment, as illustrated below (note this is lab 2).

1st assignment	Labs 2 and 3
2nd assignment	Labs 4 and 5
3rd assignment	Labs 6 and 7
4th assignment	Labs 8 and 9
5th assignment	Labs 10 and 11

  - Each assignment will be worth 6 marks (percentage points of your final grade).
  - Each assignment is spread over two labs, which means that you have over a week to work out the answers.
  - Your answers must be marked by a demonstrator during the associated labs, as shown in the table above (e.g., assignment 1 can only be marked during labs 2 and 3).
  - You will have to upload your answers on QM-Plus after graded by a demonstrator.

To get your grade

Create a document with your answers:

1. Present the answers to a TA by the end of the session for checking.
2. Upload the result to qmplus for reference.

- Both must by done by the end of the subsequent lab session at latest. (7 days grace)

Introduction

The aim of this lab is to get experience with regression problems, and the concepts of under/over-fitting, regularization and cross-validation. This lab will use both matlab and weka. Start by downloading the zip file of lab materials from qmplus. Questions in *red* are assessed toward your final grade.

Matlab Background

Start Matlab. Matlab is a high level programming language that also has many visualization and data analytics features built in. If you are new to matlab, note a few features of the interface: You have an editor window, which works like a conventional programming IDE and a command window in which you can type commands and see the results appear interactively. So you have the choice about if to program interactively with the command window, or by writing and running a full program in

the editor window. There is also a workspace window (shows the current variables – useful when programming interactively), and current folder window (see files in the current folder). The matlab stack is persistent, so if you run one script the subsequent code can see its variables unless you clear them. If you want more guidance this (6 minute) youtube video provides an overview of the interface before walking through the other features in subsequent videos. Alternatively check out a getting started guide PDF.

1. Regression and over-fitting

The point of this exercise is to understand about (i) overfitting, (ii) how overfitting affects train and test error differently, and (iii) how it depends on the complexity of the model.

1. Start matlab. Navigate to the folder where you download this week's lab materials (use cd on the command window)
2. Load a matlab data file by typing: **load('dm_lab2_1.mat')**, or clicking on it in the current folder window.
3. Type 'whos' (or look in workspace window) to see the variables that have been loaded into memory. Note the xTrain variable, its dimensions are 10x1 meaning we will be learning a 1 dimensional regression from 10 instances.
4. Navigate to the regression toolbox. Apps => Curve fitting.
5. Set the input (X) train data to xTrain and the target out (Y) data to yTrain.
6. Set the validation data to xTest and yTest (Fit=>Set Validation data).
7. Observe the visualized x & y symbols in the main plot for train & validation.
8. Press "Fit" to fit a 1st order polynomial to the presented data. This finds the line that predicts the data with minimum SSE. The Results panel on the left shows the specific f(x)=ax+b type equation that it is using, and the estimated parameters for that equation.
9. Fit 1st,2nd,..,8th order polynomials. What do you observe about the different fits in terms of their match to training and validation data?
10. Note the train SSE (sum squared error) and validation SSE for each fit.
11. Save the resulting SSEs into array variables, like: **trainSSE=[0.34, 0.28, ..]; valSSE=[1.26, 1.22,...];**
12. Now visualise the train and validation SSEs for all those models. You can use a command like: **plot(1:8,trainSSE, 1:8,valSSE);** This corresponds to the plot of train and test error from the under+over fitting lecture slide.
13.
    a. Which polynomial has the lowest train SSE, which has the lowest validation SSE? (1 mark) (You can use the data cursor on the plot)
    b. What trend do you observe about the dependence of train and validation SSE on polynomial order? (1 mark)
    c. Why?

You may find it helpful for the matlab exercises, that you can run the program cell-by-cell (%% double comment blocks). Or line-by-line (paste each line into the command window one at a time, or by click the right of the line number and use breakpoints. Then you can inspect the values of each variable at every step.

2. Regularization

The aim of this exercise is to see some actual code for fitting regression, that corresponds to the equations seen in the lectures. And to understand the impact of data volume on the need for regularization.

1. Edit '**dm_lab2_2.m**'. This contains code to try to fit data a polynomial.
2. You can compare the line that actually sets the weight matrix **w_ml**, with the solution to polynomial regression equation in the lecture notes.
3. Note that Run (arrow) icon will run the whole file. Alternatively you can also put the cursor in a comment block "Cell" which will turn yellow. Press Control-Enter will then run this block only. Good for seeing what a program does step by step.
4. Run the file and observe the fit, and train and test error. (Stretch the window sideways to make it easier to see).
5. Note the parameter lambda: this is the regularizer you learned in lectures.
6. Try a few values for lambda parameter and observe the shown fits as well as the reported train and test error. (e.g., 0, 0.001, 0.01, 0.1, 1, 10).
    - (You can do this manually by editing LAMBDA and re-running, or program a loop over lambda.)
7. Compute the sum squared values of the weights for the order 9 model by tying in the command window: **w_ml'*w_ml**, or **sum(w_ml.^2)**.
    a) How does this differ for lambda=0.001 versus lambda=1? (1 mark)
    b) What is happening here?
8. Which value for lambda gets you the minimum train error? Fill out the below:

Best Fit Lambda	Min Train Error	Corresponding Lambda	Corresponding Test Error
Order 9 Data = 10			
Order 9 Data = 100			

a. What happened in the first filled in row?
b. Now edit the parameter in the call to **utilCreatePolyData(10)** and change it to 100. This generates ten times as many data points.
c. What is the change in test error observed, and why? (1 mark)

3. (Cross)-validation

The point of this exercise is to clarify the roles of train, validation, and testing data in the process of performing machine learning and data mining.

1. Edit **dm_lab2_3.m**. (This works with a new datafile, a 5th order polynomial.)
2. Observe that after loading the data there are matricies, xTrain, and xTest.
3. This script contains a skeleton of code required to select Lambda by a simple validation: It splits the input data into two subsets: **trainset** and **valset**. For simplicity we just use validation, rather than cross-validation. It simply searches a range of lambdas and decides the best one to use (SSE). Note that **logspace(-3,1,5)** creates the range to try $[10^{-3},10^{-2},10^{-1},10^{0},10^{1}]$.
4. The incomplete part of this script is missing which portion of the data we should use as the criteria for selecting lambda. It could be Train, Val or Test.

Fix this section by commenting out two lines, leaving the correct line in, and run the code.

5. *Which is the correct way to complete the script? What is the correct train and test error? (1 mark)

4. A Real application of multivariate regression

1. Close matlab, start weka.
2. Load last week's LondonCars dataset (Explore => Preprocess => Open)
3. This time we will use the categorical variables. However first remove Model (check its Attribute view box and then press remove). This will speed up the next significantly.
4. Observe the available regression models: Classify => Choose => Functions. Choose LinearRegression for now.
5. Make sure price is selected as the target.
6. Observe that under "Test Options" weka can report the results on (i) the train data, (ii) a test split created by holding out a specified percentage, or (iii) cross-validation by averaging the performance over many folds.
7. Select use train set and start. Observe the resulting MAE.
    a. Try also 2-fold cross-validation and **50%** test split (last option).
    b. Note the train error, test error, and the difference between them. (Using the results list you can review previous settings and outputs)
8. Now lets try a strong non-linear model: Classify => Choose => Trees => M5P.
    a. Again using the "Test options" panel, get the train error; 50% split test error; and 2-fold cross-validation error.
    b. Notice the errors are now better than the linear regression model.
    c. What is the difference between the training and the testing error? Why? (1 mark)