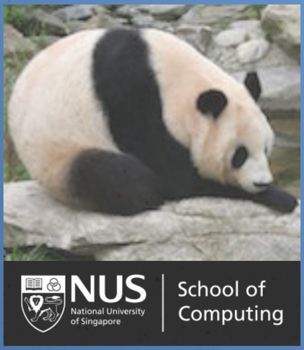

Machine Learning

NUS SoC, **2017/2018**, Semester I
Lecture Theatre 19 (LT 19) / Tuesdays 14:00-16:00

Last Updated: Monday, August 28, 2017 12:05:39 AM SGT - Creation of page.

Homework #1 » The Linear Model

In this homework assignment, you will be implementing logistic regression using batch gradient descent for binary classification. After implementing the programme, you will need to carry out a small set of experiments based on your working code. Complete the code in the template `hw1-1.ipynb` file provided. Make sure you document your code liberally using inline python comments (and Markdown cells where appropriate).

Implement a fixed learning rate algorithm for logistic regression (*LFD*, page 97). Use the "batch" formulation of gradient descent (except where indicated), instead of the more efficient stochastic gradient descent (this is to assist with automated grading). Initialize the weight with a zero vector **0**, a (column) vector of all 0's. Name your defined function `lr` for doing one iteration of logistic regression. It should work similar to the `pla` function in Tutorial 0, where the original weights are input and updated weights output by running one iteration.

Use the cross-entropy error measure for minimization (this is "equivalent" to maximum likelihood estimation); pointwise, this is $e_n(\mathbf{w}) = ln\ (1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n})$.

Note that your code will also need to calculate E_{in} (for gradient calculation) and E_{out} (for evaluation on the testing set); so creating a function that can calculate the actual error rate is needed. In this problem, although we'll run it as regression problem, the outputs are real valued at +1 and -1 and is deemed as classification (think about the issues described in lecture with respect to using regression for classification). Thus, the out-of-sample error E_{out} is given by 0/1 loss (the proportion of the misclassified points for all points) Your `lr` function also needs to take a learning rate variable, η (eta) as a parameter, which controls the step size.

After implementation, run the following experiments in different cells. In particular, pay attention to part (c) which may require you to rework part of your `lr` function to take a parameter that modifies its instance selection.

a. Run the algorithm with η = 0.05 for T = 2333 rounds. Output the eventual weight vector as a list $\boldsymbol{w}^T$ and E_{out}.
b. Run the algorithm with η = 0.005 for T = 2333 rounds. Output the eventual weight vector as a list $\boldsymbol{w}^T$ and E_{out}.
c. Instead of using batch selection, use a deterministic version of gradient descent where we select and update weights on iterations based on a single point (like SGD), based on cycling through the data with restart, where *i* = 1, 2, 3, 4, 5, .. N, 1, 2, ... Output the eventual weight vector as a list $\boldsymbol{w}^T$ and E_{out}. For this part, re-run both the first two parts, a) and b).

For a small amount of optional bonus marks (1 or 2; the assignment is worth 100 marks), you can also complete work to help show the visualization of your work on the training data. Note: This is more work that may not be commensurate to the amount of effort you need to do to accomplish the task. The below optional exercises only need to be done for part (a).

d. Use the first two input dimensions and make a 2D plot of the data and separators at every 333th iteration. Start with a light line segment (20% gray; where 100% is black and 0% is white) for the first iteration and uniformly change the color to black as you reach the final Tth iteration. Do not worry about early convergence. This part is similar to the code in Tutorial 0. As the plot only shows two inputs of the twenty, it will not show any linear separation boundary.
e. Use the first two inputs and make a 3D plot of E_{in}.
f. Trace the progress of **w** towards its termination or convergence by plotting. Start with a light line segment (20% gray) for the first iteration and uniformly change the color to black as you reach the final T^{th} iteration. Do not worry about early convergence. This plot should be superimposed on part (e), to make a plot similar to the ones shown in lecture.

Format of the input files

You'll find these files in the sample submission .zip.

- `hw1-train.txt:` the input file for you to train your logistic regression classifier model, which contains 1000 samples of 20-dimensional data, with the final class `1` or `-1` as an output, all separated by spaces.
- `hw1-test.txt:` a file of test data, containing an additional 3000 samples of the same data; again with the final class `1` or `-1` as an output, all separated by spaces.

Essay questions:

You are also asked to answer the following essay questions. These are to test your understanding of the lecture materials. Note that these are open-ended questions and do not have gold standard answers. Note that you may need to write math equations or notation in the files, so please familiarize yourself with this. You can either use a word processor or do a dump (i.e., print to .pdf) of another program that can display math (i.e., iPython notebook, R). Submit it as part of your .zip submission, exactly named as `essay1.pdf`.

1. [*LFD Exercise 1.2*] Suppose that we use a perceptron to detect spam messages. Let's say that each email messages represented by the frequency of occurrence of keywords, and the output is +1 if the message is considered spam.
    1. Can you think of some keywords that will end up with a large positive weight into perceptron?
    2. How about keywords that will get a negative weight?
    3. What parameter in the perceptron directly affects how many borderline messages end up classified as spam?
2. Consider a coin tossing experiment.
    1. Given only that the true probability, Θ, satisfies $0 \leq p \leq 1$, what is a best estimate of Θ?

   We denote the probability of *heads* of this coin as Θ. For any value of Θ, the probability of *k heads* in *n* tooses is given by the Bernoulli distribution: $\Pr(k|\Theta, n) = (n \text{ choose } k)(\Theta)^k(1 - \Theta)^{n-k}$. Say that 100 tosses of the same coin results in *heads* 70 times and *tails* 30 times.

    2. Build a model using maximum lilkelihood estimation (MLE) to infer Θ. Which value of Θ is most likely?
    3. Can we judge that this is an unfair coin? Explain your answer.

   For a small amount of extra credit, you may add simulations in your Python notebook that support your answers for Problem #2. Make sure to clearly delineate this section if you do this task.
3. In the programming logistic regression, part (c), compare our round-robin version of gradient descent which deterministically uses the next point in turn to perform the gradient descent, versus the standard, stochastic form which chooses a single point at random for an iteration. Describe whether you think this is a good robust idea or not for datasets in general.

It can be useful to repeat the question in your essay submission, for readability purposes, but it's not mandatory.

Submission Formatting

For us to grade this assignment in a timely manner, we need you to adhere strictly to the following submission guidelines. We have over 100 students in the class, and assuming it takes 10 minutes to grade each submission -- that would be over 1.3 days of uninterrupted grading with no sleep to clear. Following the submission guideliness will help us grade the assignment in an appropriate manner. You will be penalized if you do not follow these instructions. Your matric number in all of the following statements should not have any spaces and all letters should be in CAPITALS (inclusive of the ending check letter). You are to turn in the following files:

- Your source code file `hw1-1.ipynb` for this homework assignment (the "-1" suffix for the first programming problem, we only give one problem here). We will be reading your code, so please do us a favor and format it nicely. To expedite grading, we need to require all of the assignments to use Python 3.x

  In your source code `.ipynb`, describe any information you want us to know about your submission, inclusive of your **suitably filled out independent work statement**. You should not include any identifiable information about your assignment (your name, phone number, etc.) except your matric number and email (we need the email to contact you about your grade, please use your *[a|e]*******@nus.edu.sg* address, not your email alias). This is to help you get an objective grade in your assignment, as we won't associate matric numbers with student names. You can include this information at the top of your source code in Markdown cell(s) in the (just the first `-1.ipynb`, when you have multiple problems) `.ipynb` notebook source file.

  At the end of your (first) notebook source file, you need to put your **Statement of Independent Work**. Use a Markdown cell and follow the instructions given in tutorial about how to fill this section out. Please be honest and if you do not follow the course policy, suggest a plausible alternative evaluation for us to consider.

  Also put pointers or text to any (online or offline) references you have utilised in making your submission in an appropriate **References** cell.

- A .PDF file `essay1.pdf` that contains your answers to the essay questions (no word processing files, please export to PDF with any unusual fonts embedded within the document if needed.) Again, do not disclose any identifiable information in your essay answers.

- Any ancillary files that your submission uses. We may or may not read them, so please put the core part of your programming answers in the `hw1-<x>.ipynb` file.

You do not need to submit the sample files that are provided for you in the sample .zip file, these are just for your reference.

These files will need to be suitably zipped in a single file called `<matric number>.zip`. Please use a zip archive and not tar.gz, bzip, rar or cab files. Make sure when the archive unzips that all of the necessary files are found in a directory called `<matric number>` (note to Mac users, please test your zip archives on the command line; as OSX "automagically" creates the parent folder but it is often actually not contained in the archive). Upload the resulting zip file to the IVLE workbin by the due date: 19 Sep 2017, 11:59:59 pm SGT. There absolutely will be no extensions to the deadline of this assignment. Read the late policy if you're not sure about grade penalties for lateness.

Hints

New: It's suggested that you do this homework assignment in stages, and not attempting to tackle the whole thing in one step. In particular, those new to Python should try to first import the data into your program. You can write the code to import the data yourself, or use a software library, such as pandas to help you import. You may find some online utilities to do logistic regression (or desktop programs such as Microsoft Excel), which you can run with a smaller subset of the data (both in terms of the input dimensions as well as number of instances), to do some sanity checking against your own `lr` routine.

Grading Guidelines

The grading criteria for the assignment is tentatively:

- 50% Programming exercises.
- 35% Essay questions.
- 15% Documentation quality, inclusive of correctly following these submission guidelines.