

CSE 417T: Homework 2

Due: February 24 (Monday), 2020

Notes:

- Please submit your homework via Gradescope and check the [submission instructions](#).
- There will be two submission links for homework 2: one for report and the other for code. **Your score will be based on the report.** The code you submit is only used for checking the correctness and for running plagiarism checkers.
- Make sure you **specify the pages for each problem correctly**. You **will not get points** for problems that are not correctly connected to the corresponding pages.
- Homework is due **by 11:59 PM on the due date**. Remember that you may not use more than 2 late days on any one homework, and you only have a budget of 5 in total.
- Please keep in mind the collaboration policy as specified in the course syllabus. If you discuss questions with others you **must** write their names on your submission, and if you use any outside resources you **must** reference them. **Do not look at each others' writeups, including code.**
- There are 5 problems on 2 pages in this homework.

Problems:

1. (50 points) Please download the following files: two stub Matlab files and two data files
http://chienjuho.com/courses/cse417t/hw2/hw2_files.html

Complete `logistic_reg.m` and `find_test_error.m` and submit them to Gradescope. You can write supplementary scripts to help you complete the experiments below, but you do not need to submit any additional code you write.

If you are interested, you can read more about the “Cleveland” dataset we’ll be using here:
<https://archive.ics.uci.edu/ml/datasets/Heart+Disease>

Descriptions:

Learn a logistic regression model on the data in `clevelandtrain.csv`. Note that the labels in the data set are 0/1 so you will need to convert those to $-1/+1$ in order to ensure that everything we’ve done in class is still valid. Use a learning rate of $\eta = 10^{-5}$ and automatically terminate if the magnitude of every element of the gradient is less than 10^{-3} . Initialize the weight vector to a vector of all zeros. Train the model three times with a different bound on the maximum number of iterations each time: 10^4 , 10^5 , and 10^6 . (Note that the termination condition based on the magnitude of the gradient still applies). Use each model to classify the data using a cutoff probability of 0.5.

In your report submission:

- a For each of the maximum number of iterations, report E_{in} (the cross-entropy error), the binary classification error on both the training and test data sets, and how long the training process took (in seconds). What can you say about the generalization properties of the logistic regression model? How does increasing the maximum number of iterations affect the model's performance?
- b Now train and test a logistic regression model using the inbuilt matlab function `glmfit` (learn about and use the "binomial" option, and check the label format). Report E_{in} (the cross-entropy error), the binary classification error on both the training and test data sets, and how long the training process took (in seconds). (Note that `glmfit` returns a weight vector, so you might need to calculate E_{in} yourself.)
- c Now get back to your own implementation. Scale each feature by subtracting the mean and dividing by the standard deviation for each of the features in advance of calling the learning algorithm (hint: `zscore` function might be useful). Experiment with the learning rate $\eta = 0.01, 0.1, 1, 4, 6, 7$. Eliminate the iterations-based termination criteria and only terminate when the magnitude of every element of the gradient is less than 10^{-6} . For each learning rate, report E_{in} , the binary classification error on the test data, **the number of iterations required to terminate**, and the time required to train the model. Did normalizing the data affect the performance of the model? What is the effect of changing learning rate η ?

Additional note:

- You can present the numbers in a table, with each row being a method (e.g., your implementation with different max # iterations, `glmfit`, etc) and each column being a measure (E_{in} , etc) to make it clear to read.
- While you should feel free to try out stochastic gradient descent or mini-batch algorithms. In your submission to homework 2, please report your results using gradient descent algorithm (non-stochastic version).

2. (10 points) LFD Problem 2.22
3. (15 points) LFD Problem 2.24. You do not need to submit your codes for this problem.
4. (15 points) LFD Problem 3.4. For part (c), you will need to read the description of LFD Problem 1.5, but you do not need to complete LFD Problem 1.5.
5. (10 points) LFD Problem 3.19. Discuss in each case whether learning is feasible (i.e., when N goes large, does the generalization error become smaller) and explain why.

Below are some notation clarifications in the problem.

- We use $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$ to denote the training dataset. When you see \vec{x}_n (or \mathbf{x}_n in the textbook), it usually implies that it's the n -th point in the training dataset.
- In (b), $\phi_n(\vec{x})$ is the n -th element of $\Phi(\vec{x})$, i.e., $\Phi(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), \dots)$.
- In (c), (i, j) is the 2-d vector with the first element being i and second being j .