CMPS142-Spring 2018
Homework 3 (Part 2)

Handed out: May 27, 2018
Due: June 4, 2018 at 11:59 PM

- This homework will be graded more strictly than previous ones. Please be extra careful.
- This homework **has to be done in groups of 2 or 3**. Collaborating with any one not in your group (except the course staff), or taking help from any online resources for the homework problems is strictly forbidden.
- The homework is due at 11:59 PM on the due date. There is a 10% penalty for each late day, upto 3 days. After that you will not get any points for this homework. Note that your submission will be considered late even if you are late for one (or more) of the problems.
- One (and only one) member of the group has to submit the homework using his/her account on canvas. All group members will get points for that submission.
- How to submit your solutions: Your group's solution to each problem must be typed up separately (in at least an 11-point font) and submitted in the appropriate 'Problem' box on the Canvas website as a PDF file. This means that if the homework has N problems, you will submit N separate pdf files on Canvas, one for each problem per group! For example, submit the pdf that contains your group's solution to the first problem in the box titled 'Problem 1'.
- Each pdf file should clearly mention the names, email addresses and student ids of all group members. If you forget a group member's name, they will not get points for that problem.
- It is your responsibility to ensure that the files that you submit are not corrupted in any way. After you submit a file, please download it and verify that there aren't any issues. Any related requests will not be entertained after the due date.
- You are very strongly encouraged, but not required, to format your solutions in LATEX. You can use other softwares but handwritten solutions are not acceptable.
- Please try to keep the solution brief and clear.
- The Computer Science Department of UCSC has a zero tolerance policy for any incident of academic dishonesty. If cheating occurs, consequences within the context of the course may range from getting zero on a particular homework, to failing the course. In addition, every case of academic dishonesty will be referred to the student's college Provost, who sets in motion an official disciplinary process. Cheating in any part of the course may lead to failing the course and suspension or dismissal from the university.

Problem 1: Feature Engineering [50 Points]

Previous Homework. You don't need to submit anything for this problem.

Problem 2: Implementing Logistic Regression [50 Points]

In the different parts of this problem (described below), you will implement (i) Logistic Regression without a bias term, (ii) Logistic Regression with a bias term, and (ii) L2-regularized Logistic Regression. In all parts, the training will use Stochastic Gradient Ascent, and the implementation has to be in Java. To help you, we are providing the skeleton of the code in files named *.java. The files contain incomplete code, and you would need to complete the methods marked as 'TODO'. **Note that you are not allowed to delete or modify existing code from the file. You can only add lines of code in the marked sections.**

Using your completed implementations, train and test the Logistic Regression classifier(s) on the input files provided with this homework named 'HW3_TianyiLuo_train.csv' and 'HW3_TianyiLuo_test.csv'. These files contain the features that you extracted in HW3 Part 1. We have also converted the label space from *ham-spam* to *0-1*. Even though you might have generated similar files as a solution to the previous part of the homework, we require you to use the provided data files in this homework for the sake of consistency.

Implementation details (Required for consistency):

- You will notice that in all parts of the problem, the learning rate and the number of training iterations are set to be 0.01 and 200. Do not change these settings.
- Problem 2.2 should be solved with $lambda = 0.0001$. The code already sets this value. Do not change it.
- While it is common to initialize the weight vector randomly, in this homework we require you to set it to the 0 vector.
- During predicion, it is common to resolve the boundary cases (probability of class 1 = probability of class 0 = 0.5) both randomly or deterministically. In this homework, we want you to use the following prediction rule: `PREDICTED CLASS = 1 iff Probability of class` $1 \geq 0.5$ (notice that the rule says $\geq$ and not $>$).

Reminders:

- See Lecture slides from April 26th about how to implement (Regularized) Logistic Regression.
- Definitions of
  TP (True Positives): Number of instances of the positive class that were predicted correctly
  TN (True Negatives): Number of instances of the negative class that were predicted correctly
  FP (False Positives): Number of instances of the negative class that were incorrectly predicted as positives
  FN (False Negatives): Number of instances of the postives class that were incorrectly predicted as negatives

What to submit:

1. [18 points] Submit your completed code as a .zip file. The zip file should contain LogisticRegression.java, LogisticRegression_withBias.java, and LogisticRegression_withRegularization.java. Each file already has a main() and shouldn't need any additional classes to run. Note that it is your responsibility to make sure that your code runs successfully and you provide clear instructions for doing so. You will not get points if we cannot run your code. Also, for full credit, your code should be well-documented with comments.
2. [32 points] Also submit a Report (a pdf file) answering the questions asked in this homework. Even though your code should output answers to most of the questions, we want you to include them in your report. When reporting performance figures, round them off to 2 places after decimal. For questions that ask for a reason, there are more points for the explanation, than for the observation.

2.1 Logistic Regression [20 points]

Complete the code in 'LogisticRegression.java'.

Questions to answer in the report:

1. [1 point] How many parameters are you learning in Problem 2.1?
2. [1 point] What is the L2-norm of the weight vector learned by your classifier?
3. [0.5 points] What is the train set Accuracy as printed by your code?
4. [1.5 points] What are the train set Precision, Recall and F1-measure (also called F1-score) of the positive (*spam*) class as printed by your code?

5. [1.5 points] What are the train set Precision, Recall and F1-measure (also called F1-score) of the negative (*ham*) class as printed by your code?
6. [2 point] What is the confusion matrix for the train set as printed by your code?
7. [0.5 points] What is the test set Accuracy as printed by your code?
8. [3 points] What are the test set Precision, Recall and F1-measure (also called F1-score) of the positive and the negative class as printed by your code?
9. [2 point] What is the confusion matrix for the test set as printed by your code?
10. [6 points] We discussed in class that a Logistic Regression model is trained by maximizing the (log) likelihood of the training data. For optimization with gradient methods, it is not necessary to compute the log-likelihood for training. You would only need to know its partial derivative (gradient). However, for this homework we want you to additionally compute log likelihood of the training data after each training iteration. We have included a variable named 'lik', whose purpose is to store the log-likelihood value. Draw a plot of log-likelihood of the training data (y-axis) versus training iteration (x-axis), and include it in your report. You don't need to include the code for generating the plot.
11. [1 point] Count the number of positive and negative instances in your train file (you don't need to report them). Keeping those counts in mind, what can you do to improve performance of your classifier on this corpus (apart from what you are already required to do in this homework)?

You will now include two improvements in the Logistic Regression code that you implemented in Problem 2.1. These variants can be implemented with small modifications to the code in LogisticRegression.java (infact a large part of the code will be a copy of that in LogisticRegression.java). In order to avoid any 'error propagation', we strongly recommend that you convince yourself that your solution to Problem 2.1 is correct before starting to work on these parts.

2.2 Logistic Regression with a Bias Term [5 points]

In this question, you will complete the code in LogisticRegression_withBias.java. Notice that the Logistic Regression implemented in Problem 2.1 does not have a bias term. In other words, it learned a decision boundary of the form $wx = 0$ instead of $wx + b = 0$. Now we want you to include this bias term, b, and learn it in your code. Hint: As discussed in class, the simplest way to do this is to implement the bias as an additional element of the weight vector by adding the 'always 1' feature at the end of the feature vector (you can do this in the constructor for the *LRInstance* class).

Questions to answer in the report:

1. [1 point] How many parameters are you learning in Problem 2.2?
2. [2 points] What is the confusion matrix for the test set as printed by your code?
3. [2 points] Compare the training and test performance of Logistic Regression with (Problem 2.2) and without (Problem 2.1) the bias term. Does including the bias term seem useful? Why?

2.3 L2-Regularized Logistic Regression [7 points]

Implement L2-regularized Logistic Regression, as discussed in class, by completing the code in LogisticRegression_withRegularization.java. The *amount of regularization* should be controlled by a regularization coefficient λ (As you can see in the slides, the regularizer is multiplied by $\lambda/2$). Note that we DO NOT want you to include the bias term. In other words, you have to regularize the Logistic Regression from Problem 2.1 and not Problem 2.2.

Questions to answer in the report:

1. [1 point] How many parameters are you learning in Problem 2.3?

2. [2 points] What is the confusion matrix for the test set as printed by your code?

3. [2 points] Compare the training and test set performance of your classifier before (Problem 2.1) and after (Problem 2.3). What do you observe? Explain this behavior.

4. [2 points] What is the L2-norm of the weight vector learned by your classifier in this Problem. Compare the L2-norm of the weight vector as learned by your classifier before (Problem 2.1) and after (Problem 2.3). What do you observe? Explain this behavior.