Homework 5

Due: Thursday, February 27th, 2020 at 12:00pm **(Noon)**

Programming Assignment

Introduction

In this assignment, you'll implement Binary Logistic Regression with regularization to perform classification. This classification task is to predict whether or not a given patient has breast cancer based on health data. The regularization method that you will be using is Tikhonov regularization.

Stencil Code & Data

You can find the stencil code for this assignment on the course website. We have provided the following two files:

- `main.py` is the entry point of your program which will read in the data, run the classifier and print the results.
- `models.py` contains the `RegularizedLogisticRegression` model which you will be implementing.

To feed the data into the program successfully, please do *not* rename the data files and also make sure all the data files are in a directory named `data` located in the same directory as the `stencil` folder. To run the program, run `python main.py` in a terminal.

UCI Breast Cancer Wisconsin (Diagnostic) Data Set

You will be using a modified version of the Breast Cancer Wisconsin (Diagnostic) Data Set from UC Irvine's Machine Learning Repository site. You can read more about the dataset here at `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`. To modify it, we have added additional features which may or may not be informative. We have split it up into train and validation sets already for you and read them in in `main.py`.

On a department machine, you can copy the files to your working directory by running the command

```
cp -r /course/cs1420/pub/hw05/* <DEST DIRECTORY>
```

where `<DEST DIRECTORY>` is the directory where you would like to copy the four data files. If you are working locally, you will need to use the `scp` command to copy the files to your computer over `ssh`:

```
scp -r <login>@ssh.cs.brown.edu:/course/cs1420/pub/hw05/* <DEST DIRECTORY>
```

Data Format

We do a 70-15-15 split to the original dataset to produce the training, validation, and test set (you are only using train and validation, we will be using test when running your programs). We also add a constant column of ones to the dataset to account for the bias.

The Assignment

We provide you with a sigmoid function to use when training your data. In `models.py`, there are five functions you will implement. They are:

- `RegularizedLogisticRegression`:
    - `train()` uses batch stochastic gradient descent to learn the weights. You may find your solution from HW03 to be helpful, but in this assignment, we will train for a finite number of epochs rather than until we reach a particular convergence criteria. The weight update step for this assignment will also be different from HW03.
    - `predict()` predicts the labels using the inputs of test data.
    - `accuracy()` computes the percentage of the correctly predicted labels over a dataset.
    - `runTrainTestValSplit()` trains and evaluates for multiple values of the hyperparameter lambda. This function evaluates models by splitting data into train/test/validation sets, and returns lists of training and validation errors with respect to each value of lambda.
    - `runKFold()` evaluates models by implementing k-fold cross validation, and returns a list of errors with respect to each value of lambda. Note that we have defined `_kFoldSplitIndices()` for you, which you may find helpful when implementing this function.

Note: You are not allowed to use any off-the-shelf packages that have already implemented these models, such as scikit-learn. We're asking you to implement them yourself.

Binary Logistic Regression

Similar to homework 3, we are again implementing Logistic Regression for classification. However, note that there are a few key differences. For this assignment, we are performing binary classification, which is a special case of multi-class classification. For this problem, there are only two classes, which are denoted by $\{0, 1\}$ labels.

Our model will perform the following:

$$h(x) = \frac{1}{1 + e^{-<w,x>}}$$

where w is the model's weights and $h(x)$ is the probability that the data point x has a label of 1. We have implemented this as `sigmoid_function()` for you.

Our loss function will be Binary Cross Entropy:

$$L_S(h) = -\frac{1}{m}\sum_{i=1}^{m}(y_i \log h(x_i) + (1 - y_i)\log(1 - h(x_i)))$$

on a sample S of m data points. Therefore, the corresponding gradient of the Binary Cross Entropy loss With respect to the model's weights is

$$\frac{\partial L_S(h)}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)x_{ij}$$

Project Report

Guiding Questions

- Explain how you used batch stochastic gradient descent with regularization to learn the weights.
- Think back to when you implemented Logistic Regression on the Census dataset. How would it have been different if you applied Tikhonov regularization?
- Use `plotError()`, which we have implemented for you, to produce a model selection curve. Then, conclude what the best value of lambda is and explain why. NOTE: It takes about five minutes to generate a graph. Please set your default lambda in the constructor to your optimal lambda you discovered for TA testing purposes.
- In this project, you used validation data to select a model. Suppose that each patient might've had multiple samples (e.g., multiple lab tests or x-rays) collected and entered into the dataset. Would you need to account for this when splitting your train-validation-test data? If yes, how? If no, why not? (3-5 sentences)

Grading Breakdown

We expect the validation accuracy that `RegularizedLogisticRegression` reaches should be at least 83%. Because SGD is stochastic, we will run your code multiple times and use your highest accuracy to assess this objective. As always, you will primarily be graded on the correctness of your code and not based on whether it does or does not achieve the accuracy target.

The grading breakdown for the assignment is as follows:

Regularized Logistic Regression	65%
Report	35%
Total	100%

Handing in

Programming Assignment

To hand in the programming component of this assignment, first ensure that your code runs on *Python 3* using our course `virtualenv`. You can activate the `virtualenv` on a department machine by running the following command in a Terminal:

```
source /course/cs1420/cs142_env/bin/activate
```

Once the `virtualenv` is activated, run your program and ensure that there are no errors. We will be using this `virtualenv` to grade all programming assignments in this course so we recommend testing your code on a department machine each time before you hand in. Note that handing in code that does not run may result in a significant loss of credit.

To hand in the coding portion of the assignment, run `cs142_handin hw5` from the directory containing all of your source code and your report in a file named `report.pdf`.

Anonymous Grading

You need to be graded anonymously, so do not write your name anywhere on your handin. Instead, you should use the course ID that you generated when filling out the collaboration policy form. If you do not have a course ID, you should email the HTAs as soon as possible.

Obligatory Note on Academic Integrity

Plagiarism—don't do it.

As outlined in the Brown Academic Code, attempting to pass off another's work as your own can result in failing the assignment, failing this course, or even dismissal or expulsion from Brown. More than that, you will be missing out on the goal of your education, which is the cultivation of your own mind, thoughts, and abilities. Please review this course's collaboration policy and, if you have any questions, please contact a member of the course staff.