Homework 9

Due: Thursday, April 18, 2019 at 12:00pm **(Noon)**

Neural Networks

Introduction

In this assignment, you will be implementing feed forward neural networks using stochastic gradient descent. You will implement two neural networks: a single layer neural network and a two-layer neural network. You will compare the performance of all two models on the UCI Wine Dataset, which you previously used in HW2. The task is to predict the quality of a wine (scored out of 10) given various attributes of the wine (for example, acidity, alcohol content). The book section relevant to this assignment is 20.1 on page 269.

The Assignment

For this assignment, we will be evaluating each model using total squared loss (or L2 loss). Recall that the L2 loss function is defined as:

$$L(f) = \sum_{s=1}^{n} (y_s - f(x_s))^2 \ ,$$

where y_s is the target value of s^{th} sample and $f(x_s)$ is the predicted value given the learned model weights. Each of the two model will use stochastic gradient descent to minimize this loss function.

For this assignment, you will be implementing two models:

- `OneLayerNN`: The one-layer neural network is an equivalent model to Linear Regression. It also learns linear functions of the inputs:

  $$f_w(x) = w \cdot x.$$

  Therefore, when using squared loss, the ERM hypothesis has weights

  $$w = \text{argmin}_w \sum_{s=1}^{n} (y_s - f_w(x_s))^2.$$

  To find the optimal set of weights, you should use Stochastic Gradient Descent. *Hint:* Compute the derivative of the loss with respect to w. Then, use the SGD algorithm to minimize the loss.

- `TwoLayerNN`: For this model, you will be implementing the neural network described in Problem 2 of the written questions.

  For an input x, the output of the first layer of the network is

  $$h = \sigma(Wx + b_1)$$

  and the output of the second layer is

  $$z = v \cdot h + b_2$$

  This leads to an overall regression function

  $$f(x) = z(h(x))$$

  σ is an activation function. In your implementation, you will take in the activation function $\sigma(a)$ as a parameter to `TwoLayerNN`. Additionally, you will need to pass in the derivative of the activation

function $\sigma'(a)$ for training. Doing so will allow you to easily swap out the sigmoid activation function with other activation functions, such as ReLu or tanh. (You can explore these other activation functions for extra credit.)

To complete this assignment, however, you only need to train the network with the sigmoid activation function. Recall that the sigmoid activation function is (in the above formula, we apply it element-wise),

$$\sigma_{\text{sigmoid}}(a) = \frac{1}{1+e^{-a}}.$$

Important Note: External libraries that make the implementation trivial are prohibited. Specifically, `numpy.linalg.lstsqnp` (and similar functions) cannot be used in your implementation. Additionally, you **cannot** use Tensorflow or other neural network libraries. You should implement the neural networks using only Python and Numpy.

Stencil Code Data

You will be using the UCI Wine Dataset, which contains information about various attributes of a wine and its corresponding quality rating (out of 10). You can find the stencil code and data for this assignment in the course directory. You can copy the files to your personal directory on a department by machine running the command

```
cp -r /course/cs1420/pub/hw9/* <DEST DIRECTORY>
```

where `<DEST DIRECTORY>` is the directory where you would like to deposit the files. If you are working remotely, 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/hw9/* <DEST DIRECTORY>
```

We have provided the following stencil code:

- `main.py` is the entry point of program which will read in the dataset, run the models and print the results.
- `models.py` contains the `OneLayerNN` model and the `TwoLayerNN` model which you will be implementing.

You should not need to modify any code in the `main.py`. If you do for debugging or other purposes, please make sure all of your additions are commented out in the final handin. All the functions you need to fill in reside in `models.py`, marked by `TODO`s. You can see a full description of them in the section below. To run the program, run `python main.py` in a terminal with the course environment set up.

Your program assumes the data is formatted as follows: The first column of data in each file is the dependent variable (the observations y) and all other columns are the independent input variables $(x_1, x_2, \ldots, x_n)$. We have taken care of all data preprocessing, as usual.

If you're curious and would like to read about the datatset, you can find more information **here**, but it is strongly recommended that you use the versions that we've provided in the course directory to maintain consistent formatting.

Written Report

Guiding Questions

- Compare the average loss of the two models. Provide an explanation for what you observe. (10 points)
- Comment on your parameter choices. These include the learning rate, the hidden layer size and the number of epochs for training. (10 points)

- (Extra Credit) Train your neural network on every activation function provided on Slide 15 from Lecture 17. Comment on the results.
- (Extra Credit) Construct a fake dataset that has really low training error on the 2-layer neural network and high training error the 1-layer neural network. In a file `generate_data.py`, you should write a function `generate_data` that returns arrays X and Y that can be passed to the train functions of any of the three models. Comment on your approach and the results in your report.

Grading

Loss Targets

We are expecting the following training losses for each of the models on the Wine dataset:

- `OneLayerNN`: < 0.55
- `TwoLayerNN`: < 0.50

As always, we will be grading your code based on correctness and not based on whether or not you meet these targets.

Breakdown

1-Layer Neural Network	40%
2-Layer Neural Network	40%
Report	20%
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 hw9` 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.