Homework 5

Due: April 20, 2018 at 7:00PM

Written Questions

Problem 1

(25 points)
Recall that linear regression considers hypotheses that are linear functions of their inputs, $h_w(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle$. In lecture, we showed that the ERM can be efficiently computed by solving a system of linear equations for linear regression when the target loss function is squared loss:

$$L_S(h) = \frac{1}{m} \sum_{(\boldsymbol{x},y)\in S} \left(h(\boldsymbol{x}) - y\right)^2 .$$

a. Consider the "max loss" function defined as

$$L_S(h) = \max_{(\boldsymbol{x},y)\in S} |h(\boldsymbol{x}) - y|.$$

Given a dataset S, describe an algorithm for finding an ERM for linear regression with max loss.

b. Consider the "total absolute loss" function defined as

$$L_S(h) = \sum_{(\boldsymbol{x},y)\in S} |h(\boldsymbol{x}) - y|.$$

Given a dataset S, describe an algorithm for finding an ERM for linear regression with total absolute loss.

Problem 2

(15 points)
Here, we consider a 2-layer neural network that takes inputs of dimension d, has a hidden layer of size m, and produces scalar outputs.
The network's parameters are W, $\boldsymbol{b_1}$, $\boldsymbol{v}$, and b_2. W is a $m \times d$ matrix, $\boldsymbol{b_1}$ is an m-dimensional vector, $\boldsymbol{v}$ is an m-dimensional vector, and b_2 is a scalar.
For an input $\boldsymbol{x}$, the output of the first layer of the network is:

$$\boldsymbol{h} = \sigma(W\boldsymbol{x} + \boldsymbol{b_1})$$

and the output of the second layer is:

$$z = \boldsymbol{v} \cdot \boldsymbol{h} + b_2,$$

where σ is an activation function. For this question, let σ be the sigmoid activation function σ_{sigmoid} (in the formula below, we apply it element-wise):

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

We will be using the following loss function:

$$L(z) = (z - y)^2,$$

where y is a real-valued label and z is the network's output.
In this problem, you will calculate the partial derivative of $L(z)$ with respect to each of the network's parameters. Let w_{ij} be the entry at the ith row and jth column of W. Let v_i be the ith component of $\boldsymbol{v}$. Let b_{1i} be the ith component of $\boldsymbol{b_1}$. (Note that $1 \leq i \leq m$ and $1 \leq j \leq d$.)

a. Calculate $\frac{\partial L(z)}{\partial b_2}$. Show your work. (Hint: First find $\frac{\partial L(z)}{\partial z}$ and $\frac{\partial z}{\partial b_2}$, then apply the chain rule.)

b. Calculate $\frac{\partial L(z)}{\partial v_i}$. Show your work. (Hint: First find $\frac{\partial L(z)}{\partial z}$ and $\frac{\partial z}{\partial v_i}$, then apply the chain rule.)

c. Calculate $\frac{\partial L(z)}{\partial b_{1i}}$. Show your work. (Hint: First find $\frac{\partial L(z)}{\partial z}$ and $\frac{\partial z}{\partial h_i}$ and $\frac{\partial h_i}{\partial b_{1i}}$, then apply the chain rule.)

d. Calculate $\frac{\partial L(z)}{\partial w_{ij}}$. Show your work. (Hint: First find $\frac{\partial L(z)}{\partial z}$ and $\frac{\partial z}{\partial h_i}$ and $\frac{\partial h_i}{\partial w_{ij}}$, then apply the chain rule.)

Programming Assignment

Introduction

In this assignment, you will be implementing linear regression using matrix inversion and 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 three models on the UCI Wine Dataset. 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 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:

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

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

For this assignment, you will be implementing three models:

- `LinearRegression`: Linear Regression learns linear functions of the inputs:

  $$f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}.$$

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

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

  We showed in lecture that you can use matrix inversion to compute the set of weights $\boldsymbol{w}$ that minimizes the squared loss.

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

  $$f_{\boldsymbol{w}}(x) = \boldsymbol{w} \cdot \boldsymbol{x}.$$

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

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

  To find the optimal set of weights, you should use Stochastic Gradient Descent. *Hint:* Compute the derivative of the loss with respect to $\boldsymbol{w}$. 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 $\boldsymbol{x}$, the output of the first layer of the network is

  $$\boldsymbol{h} = \sigma(W\boldsymbol{x} + \boldsymbol{b_1})$$

and the output of the second layer is

$$z = \boldsymbol{v} \cdot \boldsymbol{h} + b_2$$

This leads to an overall regression function

$$f(\boldsymbol{x}) = z(\boldsymbol{h}(\boldsymbol{x}))$$

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),

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

For a single example $(\boldsymbol{x}, y) \in S$, the gradients of the loss with respect to each of the weights is

$$\frac{\partial L(\boldsymbol{x})}{\partial b_2} = 2(f(\boldsymbol{x}) - y)$$

$$\frac{\partial L(\boldsymbol{x})}{\partial b_{1i}} = 2(f(\boldsymbol{x}) - y)v_i\sigma'(W_i^T \cdot \boldsymbol{x} + b_{1i})$$

$$\frac{\partial L(\boldsymbol{x})}{\partial v_i} = 2(f(\boldsymbol{x}) - y)h_i$$

$$\frac{\partial L(\boldsymbol{x})}{\partial W_{ij}} = 2(f(\boldsymbol{x}) - y)v_i x_j \sigma'(W_i^T \cdot \boldsymbol{x} + b_{1i}).$$

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.

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). The dataset can be found in the `/course/cs1420/data/hw5` directory. You should copy it into a `data` folder in the same directory as your source code. On a department machine, you can copy the files to your working directory by running the command

```
cp /course/cs1420/data/hw5/* <DEST DIRECTORY>
```

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

```
scp <login>@ssh.cs.brown.edu:/course/cs1420/data/hw5/* <DEST DIRECTORY>
```

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.

Report

- Compare the average loss on all three models. Is there a significant difference between the loss of the 1-layer neural network and linear regression? 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. (5 points)
- (Extra Credit) Train your neural network on every activation function provided on Slide 46. 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:

- `LinearRegression`: < 0.55
- `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

Written Questions	40%
Linear Regression	20%
Network (Gradient Descent)	25%
Report	15%
Total	100%

Handing in

Written Questions

Answers to the written questions should be printed, stapled and labeled with the date, homework number and your unique course ID. You should place your handin in the CS1420 handin box located on the second floor of the CIT.

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.

Feedback

Many aspects of this course have been radically redesigned compared to previous iterations of the course. To continually improve this class and future iterations of the class, we are offering an online **anonymous feedback form**. If you have comments, we would love to hear your constructive feedback on this assignment and on any other aspects of the course.