# Homework 10

Due: Thursday, April 1, 2021 at 12:00pm **(Noon)**

## Written Assignment

### Problem 1

(25 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 $W$, $b_1$, $v$, and $b_2$. $W$ is a $m \times d$ matrix, $b_1$ is an $m$-dimensional vector, $v$ is an $m$-dimensional vector, and $b_2$ is a scalar.
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:

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

where $\sigma$ is an activation function. For this question, let $\sigma$ be the sigmoid activation function $\sigma_{\text{sigmoid}}$ (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 $i$th row and $j$th column of $W$. Let $v_i$ be the $i$th component of $v$. Let $b_{1i}$ be the $i$th component of $b_1$. (Note that $1 \leq i \leq m$ and $1 \leq j \leq d$.)
(Hint: For each part of the problem, apply the chain rule.)

a. Calculate $\dfrac{\partial L(z)}{\partial b_2}$. Show your work.

b. Calculate $\dfrac{\partial L(z)}{\partial v_i}$. Show your work.

c. Calculate $\dfrac{\partial L(z)}{\partial b_{1i}}$. Show your work.

d. Calculate $\dfrac{\partial L(z)}{\partial w_{ij}}$. Show your work.

# Programming Assignment

## 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 both 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.

## Neural Networks

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:

$$L(h) = \frac{1}{m}\sum_{i=1}^{m} \ell(h(\mathbf{x}_i), y_i) = \frac{1}{m}\sum_{i=1}^{m} (y_i - h(\mathbf{x}_i))^2 \ ,$$

where $y_i$ is the target value of $i^{th}$ sample and $h(\mathbf{x}_i)$ is the predicted value given the learned model weights. Each of the two models 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:

  $$h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b.$$

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

  $$\mathbf{w} = \text{argmin}_{\mathbf{w}} \sum_{i=1}^{m} (y_i - h(\mathbf{x}_i))^2.$$

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

- `TwoLayerNN`: For this model, you will be implement a neural network with a fully connected hidden layer.

  For an input $\mathbf{x}$, the output of the first layer of the network is

  $$\mathbf{v} = \sigma(W_1 \mathbf{x} + \mathbf{b}_1)$$

  and the output of the second layer is

  $$h = \langle \mathbf{w}_2, \mathbf{v} \rangle + b_2 \ .$$

  $\sigma$ 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. (You can explore 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),

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

## Training Neural Networks

The primary objective of training a neural network is to find a set of weights and biases that minimize the loss of our network, which in this case, is L2 loss. If these weights are all initialized to the same constant value, then they will all learn the same features. To avoid this, be sure that your implementation randomly initializes the weights. Numpy functions such as `np.random.normal` or `np.random.uniform` may be useful.

When training the two layer neural network, first calculate the gradients of the weights and biases for both layers before updating them. In the stencil, we have given you four methods for computing gradients: `_layer1_weights_gradient`, `_layer1_bias_gradient`, `_layer2_weights_gradient`, and `_layer2_bias_gradient`. Each of these methods should be called before performing gradient descent, i.e. before updating all of the gradients.

We also expect that you implement backpropagation as outlined in lecture i.e. computing all the outputs in the forward pass and saving them for use in the backward pass so that backpropagation achieves O(E) complexity, where E represents the number of edges in the network.

## Computing Gradients

Please refer to slide 28 of the Backpropagation lecture for the definition of the gradient computations.

Remember that $\sigma_1$ and $\sigma_2$ are the activation functions at each layer, and not necessarily the same! Keep in mind that in your implementation for this assignment there should be no activation applied to the output layer of the network.

Finally, note that the initial input matrix is comprised of rows, but the input to each gradient function is a vector. This is due to contradicting conventions about how to represent training data and neural network inputs. To resolve this, you can choose to transpose the input matrix immediately in the train method.

**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/hw10/* <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/hw10/* <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 add any code to `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. To run the program, run `python main.py` in a terminal with the course environment set up.

We have provided unit tests for the functions that compute gradients for the 2-layer neural network. These are `_layer1_weights_gradient`, `_layer1_bias_gradient`, `_layer2_weights_gradient`, and `_layer2_bias_gradient`. To enable these unit tests, uncomment `test_gradients` in the `main` function of `main.py`.

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 dataset, 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. (5 points)
- Comment on your parameter choices. These include the learning rate, the hidden layer size and the number of epochs for training. (5 points)
- Among machine learning techniques, neural networks have a reputation for being 'black boxes', where the logic of their decision making is difficult or impossible for humans to interpret. (5 points)

  a) If a 'black box' model gives an answer that disagrees with a human expert, which answer should be believed? Are there circumstances where we should believe one party more often than the other?

  b) Companies often hide the logic behind their products by making their software closed-source. Are there differences between companies selling closed source software, where the logic is known but hidden, versus companies selling 'black box' artificial intelligence software, where the logic might be altogether unknown? Is using one type of software more justifiable than using the other?

  Please credit any outside sources you use in your answer.

- (Extra Credit) Train your neural network using the ReLU activation function provided on Slide 15 from Lecture 18. 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.

### Hyperparameters

To verify the correctness of your implementation, check that your model satisfies the above training loss benchmarks using the given hyperparameters. After your model has satisfied these benchmarks, feel free to change the values of the hyperparameters. We will use the hyperparameter values that you choose when testing your model on the wine datasets. However, we also test your model on synthetic datasets with our own hyperparameter values, and so we strongly suggest that you first verify the correctness of your implementation before modifying hyperparameters.

### Breakdown

| | |
|---|---|
| Written Question | 16% |
| L2 Loss & Sigmoid Derivative | 4% |
| 1-Layer Neural Network | 20% |
| 2-Layer Neural Network | 48% |
| Report | 12% |
| Total | 100% |

## Handing In

### Programming Assignment

To hand in 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 hw10` 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.

## 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.