# CS 615 - Deep Learning

Assignment 3 - Backprop and Basic Architectures
Winter 2022

## Introduction

In this assignment we will implement backpropagation and train/validate a few simple architectures using real datasets.

## Allowable Libraries/Functions

Recall that you **cannot** use any ML functions to do the training or evaluation for you. Using basic statistical and linear algebra function like *mean*, *std*, *cov* etc.. is fine, but using ones like *train* are not. Using any ML-related functions, may result in a **zero** for the programming component. In general, use the "spirit of the assignment" (where we're implementing things from scratch) as your guide, but if you want clarification on if can use a particular function, DM the professor on slack.

## Grading

**Do not modify the public interfaces of any code skeleton given to you. Class and variable names should be exactly the same as the skeleton code provided, and no default parameters should be added or removed.**

| | |
|---|---|
| Part 1 (Theory) | 20pts |
| Part 2 (Visualizing Gradient Descent) | 10pts |
| Part 4 (Linear Regression) | 35pts |
| Part 5 (Logistic Regression) | 35pts |
| **TOTAL** | 100pts |

Table 1: Grading Rubric

# Datasets

**Medical Cost Personal Dataset** For our regression task we'll once again use the medical cost dataset that consists of data for 1338 people in a CSV file. This data for each person includes:

1. age
2. sex
3. bmi
4. children
5. smoker
6. region
7. charges

This time I preprocessed the data for you to again convert the *sex* and *smoker* features into binary features and the *region* into a *set* of binary features. In addition, we now *included* the *charges* information as we will want to predict this.

For more information, see https://www.kaggle.com/mirichoi0218/insurance

**Kid Creative** We will use this dataset for binary classification. This dataset consists of data for 673 people in a CSV file. This data for each person includes:

1. Observation Number (we'll want to omit this)
2. Buy (binary target value)
3. Income
4. Is Female
5. Is Married
6. Has College
7. Is Professional
8. Is Retired
9. Unemployed
10. Residence Length
11. Dual Income
12. Minors
13. Own
14. House

15. White
16. English
17. Prev Child Mag
18. Prev Parent Mag

We'll omit the first column and use the second column for our binary target $Y$. The remaining 16 columns provide our feature data for our observation matrix $X$.

# 1 Theory

1. For the function $J = (x_1 w_1 - 5x_2 w_2 - 2)^2$, where $w = [w_1, w_2]^T$ are our weights to learn:

   (a) What are the partial gradients, $\frac{\partial J}{\partial w_1}$ and $\frac{\partial J}{\partial w_2}$? Show work to support your answer (6pts).

   (b) What are the value of the partial gradients, given current values of $w = [0, 0]^T, x = [1, 1]^T$ (4pts)?

2. Given the objective function $J = \frac{1}{4}(x_1 w_1)^4 - \frac{4}{3}(x_1 w_1)^3 + \frac{3}{2}(x_1 w_1)^2$:

   (a) What is the gradient $\frac{\partial J}{\partial w_1}$ (2pts)?

   (b) What are the locations of the extrema points for this objective function $J$ if $x_1 = 1$? Recall that to find these you set your equation to zero and solve for, in this case, $w_1$. (5pts)

   (c) What does $J$ evaluate to at each of your extrema points, again when $x_1 = 1$ (3pts)?

# 2 Visualizing Gradient Descent

In this section we want to visualize the gradient descent process for the following function (which was part of one of the theory questions):

$$J = (x_1 w_1 - 5 x_2 w_2 - 2)^2$$

Hyperparameter choices will be as follows:

- Initialize your parameters to zero.
- Set the learning rate to $\eta = 0.01$.
- Terminate after 100 epochs.

Using the partial gradients you computed in the theory question, perform gradient descent, using $x = [1, 1]^T$. After each training epoch, evaluate $J$ so that you can plot $w_1$ vs $w_2$, vs $J$ as a 3D line plot. Put this figure in your report.

# 3 Backpropagate and Update the Weights

To perform backwards propagation, your non-objective and non-input modules must have a *backward* method that takes as its implicit parameters an incoming (backcoming?) gradient and returns the gradient to be backpropagated. Since many of the modules have the same backpropagation rules, it might be logical to have a default one in your abstract class, then override it in your derived classes, as needed.

```
def backward(self,gradIn):
  #TODO
```

In addition, for the *FullyConnected* module, implement an *updateWeights* method that takes as parameters the backcoming gradient and a learning rate *eta* and updates the weights and biases of the layer.

```
def updateWeights(self,gradIn, eta = 0.0001):
  #TODO
```

# 4 Linear Regression

In this section you'll use your modules to assemble a linear regression model and train and validate it using the *medical cost dataset*. The architecture of your linear regression should be as follows:

$$Input \rightarrow \text{Fully-Connected} \rightarrow \text{Least-Squared-Objective}$$

Your code should do the following:

1. Read in the dataset to assemble $X$ and $Y$
2. *Shuffle* the dataset, extracting $\frac{2}{3}$ for training and $\frac{1}{3}$ for validation.
3. Train, via gradient learning, your linear regression system using the training data. Refer to the pseudocode in the lecture slides on how this training loop should look. Initialize your weights to be random values in the range of $\pm 10^{-4}$ and your learning rate to be $\eta = 10^{-4}$. Terminate the learning process when the absolute change in the mean absolution percent error (MAPE) on the training data is less than $10^{-10}$ or you pass $10,000$ epochs. During training, keep track of the RMSE and MAPE for both the training and validation sets so that we can plot these as a function of the epoch.

In your report provide:

1. Your plots of training and validation RMSE vs epoch.
2. Your plots of training and validation MAPE vs epoch.

# 5 Logistic Regression

Next we'll use a logistic regression model on the *kid creative* dataset to predict if a user will purchase a product. The architecture of this model should be:

$$Input \rightarrow \text{Fully-Connected} \rightarrow \text{Sigmoid-Activation} \rightarrow \text{Log-Loss-Objective}$$

Your code should do the following:

1. Read in the dataset to assemble $X$ and $Y$
2. *Shuffle* the dataset, extracting $\frac{2}{3}$ for training and $\frac{1}{3}$ for validation.
3. Train, via gradient learning, your logistic regression system using the training data. Initialize your weights to be random values in the range of $\pm 10^{-4}$ and your learning rate to be $\eta = 10^{-4}$. Terminate the learning process when the absolute change in the log loss is less than $10^{-10}$ or you pass $10,000$ epochs. During training, keep track of the log loss for both the training and validation sets so that we can plot these as a function of the epoch.

In your report provide:

1. Your plots of training and validation log loss vs epoch.
2. Assigning an observation to class 1 if the model outputs a value greater than 0.5, report the training and validation accuracy.

## Submission

For your submission, upload to Blackboard a single zip file containing:

1. PDF Writeup
2. Source Code
3. readme.txt file

The readme.txt file should contain information on how to run your code to reproduce results for each part of the assignment.

The PDF document should contain the following:

1. Part 1: Your solutions to the theory question
2. Part 2: Nothing
3. Part 3: Your plot.
4. Part 4: Your two plots.
5. Part 5: Your plot and your accuracies.