COMP 411, Homework 6: *Implementing a Neural Network, Part 1*
Due by 11:59pm on November 9, 2021

---

1. Titanic Dataset

For this homework you will work with the titanic.csv dataset included in the code distribution. This dataset has 10 features: 1) Ticket Class, 2) Sex, 3) Age, 4) Number of siblings and spouses aboard, 5) Number of parents and children aboard, 6) Fare, 7) Embarked Cherbourg?, 8) Embarked Queenstown?, 9) Embarked Southampton?, 10) Survived. Our goal will be to use the first 9 features to predict the last feature, Survived.

Problem 1. *You will find in the neural_net.py that you have been given that the Titanic dataset is already loaded into a dataframe then numpy array. However, there is still a bit more work to do.*

(1) Use the dataframe describe function to get some statistical information about each feature column. Then use matplotlib to make a boxplot with whiskers for each feature (see https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.boxplot.html).
(2) For the Survived feature, please map the -1 and +1 values to 0 and 1 (so that the values match the range of the sigmoid activation function used at the output layer).
(3) Based on how the other feature values besides Survived are distributed, you may decide to normalize some of them. In addition to implementing any needed normalization, describe why you chose to normalize or not, and how you did any normalization and why you choses that approach.
(4) Split the data into training and test sets. If you'd like to do cross-validation, you will need to add addtional scaffolding.

2. Neural Network Functions to Implement

The focus of this homework and the next homework is to implement the code scaffolding to train a neural network and use a neural network for prediction.

Problem 2. *You will use the Titanic dataset to do initial testing of your neural network implementation for this homework. Because the Survival feature takes on only 2 values, we have a binary classification problem.*

(1) Implement the init_parameters and update_parameters functions. These functions initialize the weight and bias vectors.
(2) Implement the forward pass function. This computes the output of the neural network. Assume the output layer is a single node with a sigmoid activation unit. Assume the hidden layer uses tanh activation. You should be able to process a batch of examples at once (all of $\mathbf{X}$) rather than a single example (one example in $\mathbf{X}$) at at a time.
(3) Implement the cost aka loss function. The loss function you will use is the cross-entropy aka logarithmic loss aka log loss. The loss function is primarily used to evaluate the performance of the neural network optimization (such as over-fitting and convergence).

$$\textit{Log loss} \;=\; \sum_{i=1}^{N} -\left(Y_i^{True} \times \log(Y_i^{Estimate}) + (1 - Y_i^{True}) \times \log(1 - Y_i^{Estimate})\right)$$

You should add a very small value, such as 1e-15 to the $Y^{Estimate}$ *prediction to ensure that you never take a log of 0.*

(4) *Add a function to compute accuracy of the performance of the model. Since the neural network parameters are currently random, the performance should be terrible. Nonetheless, this function will be useful for the next homework. While the loss function is necessary for training, interpreting what the log loss function means for performance on the problem we are trying to solve may be difficult. That is, the goals of optimization may differ from what we want to use to measure model performance on different data. Thus, here you will use accuracy (the fraction of predictions that are correct) to evaluate performnce. What accuracy do you see for the dataset given random parameters?*

Do not fill out the backward_pass function just yet. You will do this on the next homework.