CS 171 Homework 3

In this assignment, you will revisit the multilayer perceptron model that we examined in class. There are two options for this homework assignment, described below.

For either option, your task in this assignment is to create a new Jupyter notebook to tackle the 5 sub-problems described below. Your Jupyter notebook should be well-formatted with a mix of code and markdown cells with headers for each section. You should use homeworks 1 and 2 as guidelines i.e. your homework 3 should look quite similar, but should address the problems described below.

As for all other assignments, you should restart your kernel and run all cells in succession to create a clean output in your notebook after you finish your assignment. Then, create a PDF of your notebook and turn both files in on Canvas.

Option 1: Implement a different loss function

When we implemented our multilayer perceptron model, we used the mean square error (MSE) loss function – a familiar friend from previous examples. However, there are other options for loss functions. In this option, you will swap out the MSE loss function with a new loss function of your choosing. You'll need to do some research on other options out there. Note that your loss function gradient will change so you will need to re-compute your gradients and update your code. Create your notebook to address the following problems:

- Problem 3.1: Write down your loss function and describe how it compares to MSE.
- Problem 3.2: Compute the gradient of your loss function. Then, describe what needs to change in your approach to implement this new gradient.
- Problem 3.3: Modify the multilayer perceptron code to implement the new loss function and its gradient for training.
- Problem 3.4: Train your model on the MNIST dataset
- Problem 3.5: Compare and contrast your model results with the model results using MSE. Make plots to compare losses and accuracies. Describe what has changed and what has remained the same.

Option 2: Implement another hidden layer

When we implemented our multilayer perceptron model, we implemented a single hidden layer. In comparison to the single layer perceptron, we saw an improvement in the overall accuracy (see Problem of the Week 6). This raises the question: what if we had TWO hidden layers? In this option, you'll implement yet another hidden layer in the existing model.

Create your notebook to address the following problems:

- Problem 3.1: Create a diagram that depicts the layers of your model. Write down the sizes of each matrix in the model.
- Problem 3.2: Your new model should now have three weight matrices $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$, which need to be trained i.e. you will need to compute the $\partial L/\partial \boldsymbol{w}_1$, $\partial L/\partial \boldsymbol{w}_2$, and $\partial L/\partial \boldsymbol{w}_3$. The latter two will be very similar to the existing model but the first will be new in this application. Write down a matrix representation of $\partial L/\partial \boldsymbol{w}_1$ to implement in the backward step of your model. You can either use LaTeX-formatted equations in Markdown cells or you can write your equations on paper and turn them in separately on Canvas.
- Problem 3.3: Modify the multilayer perceptron code to implement the new layer.
- Problem 3.4: Train your model on the MNIST dataset
- Problem 3.5: Compare and contrast your model results with 2 hidden layers with the model results using 1 hidden layer. Make plots to compare losses and accuracies. Describe what has changed and what has remained the same.