CS-449, Winter 2024
Homework #1: Multi-Layer Perceptrons
Due Date: Tuesday, February 6th @ 11:59PM
Total Points: 9.0 (plus optional 2.0 bonus point)

In this assignment, you will work with your group to implement and train a multi-layer perceptron **without using a deep learning platform** (*e.g.,* PyTorch, Keras, JAX, etc.) You may discuss the homework with other groups, but do not take any written record from the discussions. Also, do not copy any source code from the Web.

Guidelines

- Your architecture should consist of two layers -- one hidden layer with up to *k* nodes and a single output node.
- You can use any activation function (*e.g.,* sigmoid, tanh, etc.) in the hidden nodes.
- You should use a sigmoid for the output layer of your network.
- Your model must use a bias term at the input and hidden layers.
- You will train and test your implementation of four datasets: `two_gaussians`, `center_surround`, `xor` and `spiral`. The datasets are provided in CSV files as part of the assignment.
- You can use symbolic differentiation tools like WolramAlpha, Mathematica, etc., to compute gradients. You can also calculate the gradients by hand.
- You may want to calculate, code and verify gradients for individual components of your model and use the chain rule to build the gradients for specific weights and biases.
- You should use gradient descent to train your MLP.
- The dataset contains binary labels {0,1}. You should apply a threshold of 0.5 to the output of your MLP when assigning labels at test time.
- Although the code you turn in cannot use PyTorch (or similar packages), you may find it helpful to "calibrate" intermediate quantities in your implementation against a PyTorch implementation.
- You may find it helpful use a spreadsheet for a hardcoded example to check your code.
- You may find it helpful to use a random number seed for reproducibility when debugging.
- You may want to consider using `for loops` instead of matrix algebra in some parts of your code to avoid the ambiguity of broadcasting.
- You should use `numpy` (and numpy arrays) for most calculations.
- You do not need to use a GPU for this assignment, and your models should train in less than one minute each.
- You are responsible for selecting hyper-parameters (*e.g.,* number of hidden nodes, learning rate, training epochs, batch sizes, early stopping criteria, etc.). The goal is to get "good" performance from your model, but an exhaustive hyper-parameter search is unnecessary.
- You can complete the assignment entirely as a Jupyter Notebook, or you can turn in a single Python file with an accompanying PDF of the results.
- Your code should use parameters to control all functionality needed to complete specific tasks (see below).

Steps to complete the homework

1. (4.0 points) Implement and train an MLP network as specified above using binary cross entropy as the loss function. Experiment with each of the four datasets to find the best number of nodes *k* in the hidden layer. For the best *k* for each dataset:
    a. list hyper-parameters used in the model,
    b. plot the learning curves for training and validation loss as a function of training epochs,
    c. provide the final text accuracy, defined as the number of correct classifications divided by the total number of examples,
    d. plot the learned decision surface along with observations from the test set, and
    e. discuss any decisions or observations that you find relevant.
2. (2.0 points) Repeat Step 1 using mean squared error as the cost function.
3. (2.0 points) Select the worst-performing model (dataset, cost function and number of hidden nodes) from the above experiments and plot decision boundaries learned for each node in your hidden layer. Discuss why you selected this instance and speculate about factors contributing to poor performance.
4. (1.0 points) Discuss how you might encourage your model to learn "feature maps" that could improve the performance of the instance selected for Step 3.
5. (BONUS 2.0 points) Implement the approach specified in Step 4. Discuss and present results that illustrate "why" your approach did or did not enhance performance. Note: You may use PyTorch (or another deep learning package) to implement this part of the assignment.

Suggestion: You may find it helpful to look at https://karpathy.github.io/2019/04/25/recipe/. Despite blog's title, it is not meant to provide a step-by-step guide to complete this assignment. Instead it introduces some general principles that you should consider.

Submission Instructions

Turn in your homework as a single zip file or Jupyter Notebook, in Canvas. Specifically:

1. Create a single Jupyter Notebook or a single PDF file with the answers to the questions above, and your graphs.

2a. Create a single ZIP file containing:

- `homework1.pdf`
- Your `.py` code file

-or-

2b. Your Juptyer Notebook

3. Turn the zip or ipynb file in under Homework #1 in Canvas.

Good luck, and have fun!