MSAI-437, Winter 2025
Homework #1: Perceptrons
Due Date: Wednesday, February 5th @ 11:59PM
Total Points: 10.0

In this assignment, you will work with your group to explore various aspects perceptrons. You may discuss the homework with other groups, but do not take any written record from the discussions. Each group must submit their own work. Do not submit another group's work as your own, and do not allow another group to submit your work as their own. Also, do not copy any source code from the Web or use code generated by AI.

Question #1 (2.0 points)

Consider the linear classifier with the weights and decision boundary above. Define $g(X) = X \cdot w$. We'll define our classification prediction $h(X) = sign(g(X))$. You can assume the sign (not sine) function is implemented as in numpy. Using the parameters $w = \langle 1, 0.9, -1.3 \rangle$ shown below, we can calculate $g(X)$ and $h(x)$ as:

$\mathbf{X}$	y	$g(\mathbf{X})$	$h(\mathbf{X})$
$(1,2)$	1	-0.7	-1
$(2,1)$	-1	1.5	1
$(2,3)$	-1	-1.1	-1
$(4,3)$	1	0.7	1
$(10,3)$	1	6.1	1

Right now, the classifier gets 60% accuracy. We want to think about how different loss functions could help the model improve. In class, we've discussed the perceptron loss and mean squared error as ways of evaluating the quality of model predictions. First, read up on Hinge loss and Binary Cross Entropy loss to understand how those work. We include definitions below, but it'll be helpful to understand the "margin" aspect of hinge loss and the probabilistic interpretation of BCE loss. Using our $g(X)$ definition above, we can define each loss as the following:

1. Perceptron (zero-one) loss:

$$L(x,y,\theta) = \begin{cases} 0 & \text{if } y \cdot g(x) > 0 \\ 1 & \text{otherwise} \end{cases}$$

2. Squared error loss:

$$L(x,y,\theta) = (y - g(x))^2$$

3. Binary cross-entropy loss (written in an atypical way):

$$L(x,y,\theta) = \begin{cases} \ln(1+\exp(-g(x)) & \text{if } y > 0 \\ \ln(1+\exp(g(x)) & \text{otherwise} \end{cases}$$

4. Hinge loss:

$$L(x,y,\theta) = \max(0, 1 - y \cdot g(x))$$

For each loss, answer the following questions. You must provide an explanation for full points.

a. For this loss, which point(s) have the highest loss value? Why?
b. For this loss, which point(s) have the lowest loss value? Why?
c. With this dataset and loss, is it possible to find a different set of model parameters w` that would have a lower total loss? Don't calculate the loss for any specific w` values; just look at the plot and use your intuition. You may answer "Yes", "No", or "Maybe".

   If you answer "Yes", give a general description of what the new decision boundary would look like and why that would decrease the loss (e.g., "if you rotated the current boundary clockwise, then …").

   If you answer "No", give a general argument for why there is no decision boundary that could decrease the loss.

   If you answer "Maybe", give a general description of a new decision boundary that might decrease the loss, but it's hard to tell without doing the calculations. Also, provide a general argument for why there's no decision boundary that obviously decreases the loss.
d. Is this loss function a good choice for training a multilayer perceptron on a binary classification task? Why or why not?

Question #2 (4.0 points)

Augment the dataset used in Question #1 with at least 100 additional observations that are representative of the original dataset. Sample from the augmented dataset to construct train, validation and test splits. For each of the cost functions listed in Question #1, train a Single Layer Perceptron using gradient descent without using PyTorch or any other deep learning platform. You should:

- Discuss all design choices (e.g., architectural, equations, hyper-parameters, hyper-parameter values, etc.) and explain why you made these choices,
- Discuss whether it is possible or not possible to use another learning method (e.g, the perceptron algorithm), why or why not, and the benefits and disadvantages of doing so,
- Submit the code associated with using gradient descent for each cost function,
- Use the plotting functions in the starter code to generate a graphic of the augmented dataset and the learned decision boundary,
- Include you augmented dataset as a comma-separated values file, and
- Include detailed results on the training, validation and test datasets.

Question #3 (2.0 points)

Read through the code provided with the assignment to get a sense of how we're setting up these experiments. Then, you can run all the cells.

a. In the `run_one_epoch` function, the code uses a combination of `where`, `argmax`, and `torch.nn.CrossEntropyLoss`. How do these calculations work with the model's output to compute the loss and accuracy? Why does the model output a real-valued tensor of shape (N, 2)? You may want to look at the documentation for the loss function.

b. Read through the `run_experiment`, `pretrain_and_train`, and `plot_results` functions to understand how the figure is being built. In your own words, describe in general what the six panels show. What data is being used in which panels?

Question #4 (2.0 points)

The example experiment we provide has the following keyword arguments:

```
kwargs = {
"title": "Example Experiment",
"radii": (2, 6, 10),
"examples_per_ring": 100,
"layer_sizes": [100, 10],
"activation": torch.tanh,
"learning_rate": 0.001,
"n_pretrain_epochs": 100,
"n_train_epochs": 1000,
}
```

For your experiments, you should change these values and rerun run_experiment(**kwargs) and track what happens. You can run as many or as few experiments as you want, but they should have unique title values (i.e., not "Example Experiment"), and your written answers to the following questions should reference the figures produced by specific experiments. That is, don't say "the model tends to overfit"; say "in my Example #4 above, we can see that the model is overfitting because …". You can copy the example experiment we provide into new Code cells, edit the kwargs, and run the cell to save the output to your notebook.

a. Overfitting Experiments: Include at least two experiments that show the model overfitting. Describe what those experiments show and how you know the model is overfitting. What arguments had the most effect on whether your model overfits?

b. Pretraining Experiments: The point of pretraining is to get the model "started" with an easier task (rings) so it can more quickly learn the task we care about (spirals). You can control the amount of ring data, the amount of spiral data, and the number of epochs the model train on each. In your experiments, assume that all we care about is maximizing the final mean test set accuracy. Based on your experiments, when does pretraining help? When does it hurt? Your answer should be at least two paragraphs and reference at least three different experiments you ran.

c. Best Experiment: Across all the experiments you ran, with which arguments did you achieve the highest Final Mean Test Accuracy? What patterns led you to find these arguments? Which arguments had the largest impact on your experimental results? What was more difficult than you expected? Your answer should be at least two paragraphs and reference at least three different experiments you ran. At least two of these experiments must be different from those included in Pretraining Experiments above.

Submission Instructions

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

1. Create a single pdf file `hw1.pdf` with the answers to the questions above, and your graphs.
2. Create a single ZIP file containing:
   - `hw1.pdf`
   - All of your `.py` or `.ipynb` code files
3. Turn the zip file in under Homework #1 in Canvas.

Note: You may also complete the assignment and include all responses in a single Jupyter Notebook in a single ZIP file.

Good luck, and have fun!