First Assignment

Shahid Beheshti University - Bachelor’s Program

Artificial Neural Networks - Winter 2024

Theoretical Exercises

Exercise 1

Why is it generally preferable to use a Logistic Regression classifier rather than a classical Perceptron (i.e., a single layer of threshold logic units trained using the Perceptron training algorithm)? How can you tweak a Perceptron to make it equivalent to a Logistic Regression classifier?

Exercise 2

Explain how Cross-Entropy and Mean Squared Error differ from one another. Which do you think is better for the classification problem, and please explain your decision. (After giving an explanation of each, go on to discuss how they differ from one another.)

Exercise 3

Explain why training MLPs requires accurate weight initialization. Examine various weight initialization strategies, including He, Xavier, and random initialization, and describe when each might be suitable. Talk about and solve potential issues that could arise if the weights in our model are symmetric.

Exercise 4

Talk in-depth about the benefits and drawbacks of using the ReLU activation function as opposed to the sigmoid activation function.

Exercise 5

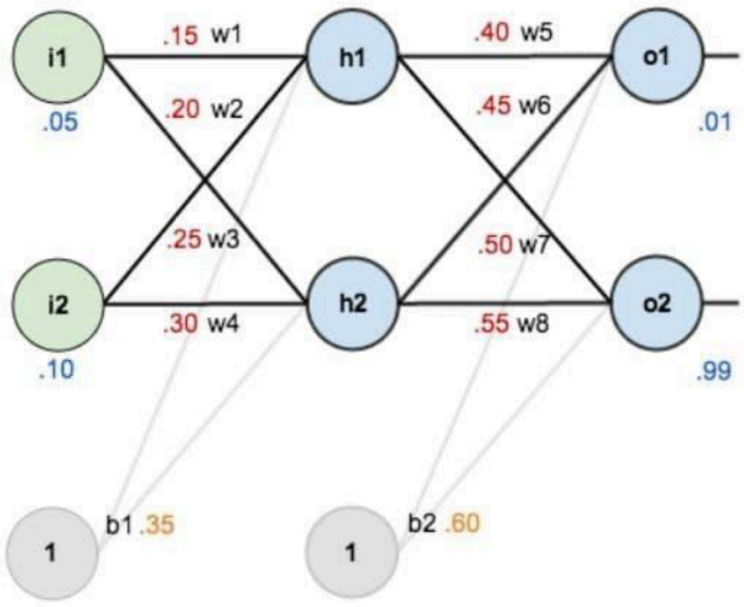
Examine the idea of sparse connectivity in multilayer perceptrons and how it affects the effectiveness of the model. Describe the effects of various pruning techniques on model size and performance, emphasizing the effects of optimal brain damage, sensitivity-based pruning, and magnitude-based pruning.

Exercise 6

A neural network learns by adjusting its weights. The commonly used technique for updating effectively is called "backward propagation of errors," or backpropagation. Now consider the neural network below, where the cost function is "Mean Squared Error," and the activation function of the neurons is Sigmoid.

a. Calculate the network error after one step of feed-forward.

b. Determine the backpropagation step with a learning rate of 0.3 .



Weights

W1 = .15 i1->h1	**W2 = .20 i1->h2**	**W3 = .25 i2->h1**
W4 = .30 i2->h2	**W5 = .40 h1->o1**	**W6 = .45 h1->o2**
W7 = .50 h2->o1	**W8 = .55 h2->o2**	

Practical Exercise

Problem Description

In this exercise, you will create a basic Multilayer Perceptron (MLP) from scratch that uses only NumPy to categorize wines according to specific attributes. The Wine Quality Dataset, which includes a variety of physicochemical characteristics of wine samples and their corresponding quality ratings, will be used by you.

Dataset Description

The Wine Quality Dataset contains two sets of data: one for red wine and another for white wine. Each dataset contains several physicochemical attributes of wine samples, such as acidity, pH, and alcohol content, along with a quality rating. For this exercise, we will be focusing on the red wine dataset.

The red wine dataset consists of the following attributes:

- Fixed acidity
- Volatile acidity
- Citric acid
- Residual sugar
- Chlorides
- Free sulfur dioxide
- Total sulfur dioxide
- Density

- pH
- Sulphates
- Alcohol
- Quality (score between 0 and 10)

Tasks

1. Data Loading: Load the red wine dataset (winequality-red.csv) using NumPy. You can download the dataset here.

2. Data Preprocessing: Perform any necessary preprocessing steps, such as normalization or standardization, on the dataset. Split the dataset into training and testing sets (e.g., 80% training, 20% testing).

3. Model Implementation: Implement a Multilayer Perceptron (MLP) from scratch using only NumPy. Design the architecture of the MLP, including the number of input nodes, hidden layers, neurons per hidden layer, and output nodes. You can start with a simple architecture with one hidden layer.

4. Training: Train your MLP model using the training data. Implement the backpropagation algorithm to update the weights of the network iteratively and also try training your network using 15 different learning rates ranging from [1e-8, 10] and report the plot of the loss reduction.

5. Evaluation: Evaluate the performance of your trained model using the testing data. Calculate relevant evaluation metrics such as accuracy, precision, recall, and F1-score.

6. Analysis: Analyze the results of your model and discuss any observations or insights you have gained. Consider factors such as model performance, convergence behavior, and potential areas for improvement.

7. Choose 10 different types of activation functions and plot their distribution and also plot their derivatives and their distributions as well .

Good Luck, Have Fun, Code a Lot