National University of Singapore
School of Computing
CS2109S: Introduction to AI and Machine Learning
Semester I, 2023/2024

Tutorial 7
Neural Networks

These questions will be discussed during the tutorial session. Please be prepared to answer them.

Summary of Key Concepts

In this tutorial, we will discuss and explore the following learning points from Lecture:

1. Neural Network
   (a) Forward Pass
   (b) Intermediate Layers
2. Activation Function
   (a) Effects of using activation functions

Problems

1. **Logic Gates**

AND		
x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

OR		
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

NOT	
x	y
0	1
1	0

Figure 1: Logic Gates

   (a) Determine a function that can be used to model the decision boundaries of the logical NOT, OR, and AND gates. For this question, assume that AND, and OR functions take 2 inputs while the NOT function takes a single input. What are the weights and bias for each perceptron respectively? The steps are given below:
   - Initialise all your weights to 0 (including bias term)

- Positive class is 1 and negative class is 0
- Use the Perceptron Update Rule and the learning rate from Lecture 8 ($\eta = 0.1$)
- Update the model for each misclassified instance in the **given fixed order** of data samples
- Activation function is the modified sign function. $\text{sign}(z) = 1$ if $z \geq 0$ else 0

(b) Is it possible to model the XOR function using a single Perceptron? Refer to Figure 2 for the truth table of the XOR gate. Comment on your answer.

XOR		
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Figure 2: XOR Gate

(c) Model XOR function (takes 2 inputs) using a number of perceptrons that implement AND, OR, and NOT functions. Show the diagram of the final Perceptron network. What can you conclude now that you have intermediate functions acting on the inputs and not just a single perceptron unit?

(d) If we change the ordering of data samples in Perceptron Update Rule, will the model converges to a different model weight for the AND operator? What can you conclude from the observation? (Hint: Use your solution from (a), and try switching the row-orderings of the input data from $(0, 1, 2, 3)$ to $(0, 2, 3, 1)$ and $(0, 2, 1, 3)$ respectively)

(e) With regards to Figure 1, does your proposed model have high bias and high variance?

2. **Single vs Multi Layer Perceptron**

Suppose you have a dataset of 500 product sales, each with three input features (TV, radio, and newspaper advertising expenditure) and a continuous label indicating the sales.

You decide to use a single-layer perceptron and a multi-layer perceptron to make predictions. After training both networks, you obtain a mean squared error of 1000 on the training set and a mean squared error of 2000 on the validation set for the single-layer perceptron, and a mean squared error of 800 on the training set and a mean squared error of 1200 on the validation set for the multi-layer perceptron.

(a) What might be the reasons for the difference in performance between the single-layer perceptron and the multi-layer perceptron?

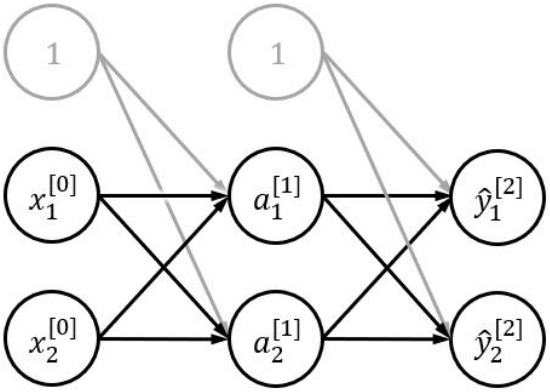
(b) How might you modify the single-layer perceptron to improve its performance, and what are the advantages and disadvantages of doing so?

(c) What techniques could you use to improve the performance of the multi-layer perceptron?

3. **Dot Product-Activate, Forward Propagation**

In this question, we are going to use a neural network with **a 2-D input, one hidden layer with two neurons, and two output neurons**. Additionally, the hidden neurons and the input will **include a bias**. We use **ReLU function** as the non-linear activation function. FYI: $\text{ReLU}(x) = \max(0, x)$.

Here's the basic structure:



Disclaimer: You may use `NumPy` to compute the following!

Suppose there is a data input $\mathbf{x} = (2, 3)^\top$ and the actual output label is $\mathbf{y} = (0.1, 0.9)^\top$. The weights for the network are

$$\boldsymbol{W}^{[1]} = \begin{bmatrix} 0.1 & 0.1 \\ -0.1 & 0.2 \\ 0.3 & -0.4 \end{bmatrix}, \boldsymbol{W}^{[2]} = \begin{bmatrix} 0.1 & 0.1 \\ 0.5 & -0.6 \\ 0.7 & -0.8 \end{bmatrix},$$

Of course, we also include biases of value $b = 1$ for both hidden and output layers. Calculate the following values after the forward propagation:
$\mathbf{a}^{[1]}$, $\hat{\mathbf{y}}^{[2]}$ and $L(\hat{\mathbf{y}}^{[2]}, \mathbf{y})$. Here, we use MSE (mean squared error) loss function.

4. **Let's Activate!**

We can define a neural network as follows:

$$\hat{y} = g(\mathbf{W^{[L]^T}} \dots g(\mathbf{W^{[2]^T}} \cdot g(\mathbf{W^{[1]^T}} x)))$$

where $\mathbf{W}^{[\mathbf{l}] \in \{\mathbf{1}, \cdots, \mathbf{L}\}}$ is a weight matrix. You're given the following weight matrices:

$$\mathbf{W^{[3]}} = \begin{bmatrix} 1.2 & -2.2 \\ 1.2 & 1.3 \end{bmatrix}, \mathbf{W^{[2]}} = \begin{bmatrix} 2.1 & -0.5 \\ 0.7 & 1.9 \end{bmatrix}, \mathbf{W^{[1]}} = \begin{bmatrix} 1.4 & 0.6 \\ 0.8 & 0.6 \end{bmatrix}$$

Furthermore, you are given $g(z) = \text{SiLU}(z) = \frac{z}{1+e^{-z}}$ between all layers *except the last layer*.

Disclaimer: Feel free to use `NumPy` to help with the following question!

Is it possible to replace the whole neural network with just one matrix in both cases **with** and **without** non-linear activations $g(z)$? For both cases, either shows that it is possible by providing such a matrix or prove that there exists no such matrix. What does this signify about the importance of the non-linear activation?

5. **Working with Dimensions**

You're building a self-driving car program that takes in grayscale images of size 32×32 where 32 is the image height and width. There are 4 classes your simplified program has to classify: $\{\text{car}, \text{person}, \text{traffic light}, \text{stop sign}\}$. You start off experimenting with a Multi-layer Perceptron composed of three linear layers of the form $y = W^T x$, where $x \in \mathcal{R}^d$ is the input vector, W is the weight matrix, and y is the network output.

What are the dimensions of the input vector, the weight matrix, and the output vector of the three linear layers, given the following details? Assume the batch size is 1.

layer	Input dim	Weight Matrix dim	Output dim
Linear layer 1	_____ $\times 1$	_____ $\times$ _____	512×1
Linear layer 2	512×1	_____ $\times 128$	_____ $\times 1$
Linear layer 3	128×1	_____ $\times 4$	_____ $\times 1$