Task 3: General Questions

1 Question 1

You are given a binary classification problem with two features: $X1$ and $X2$. The perceptron algorithm is applied to train a model for this problem. After several iterations, the decision boundary converges, and the final weights are $W1 = 2$, $W2 = -1$ and the bias $b = 0.5$. The activation function is a step function with the threshold set at 0.

1.1 (a) Prediction of Class Labels

We use the perceptron decision rule:

$$y = \text{sign}(W1 \cdot X1 + W2 \cdot X2 + b)$$

For the given points:

- Point 1: $X1 = 2$, $X2 = 4$

  $$y = \text{sign}(2 \cdot 2 + (-1) \cdot 4 + 0.5) = \text{sign}(4 - 4 + 0.5) = \text{sign}(0.5) = 1$$

  Predicted class: 1

- Point 2: $X1 = 4$, $X2 = 2$

  $$y = \text{sign}(2 \cdot 4 + (-1) \cdot 2 + 0.5) = \text{sign}(8 - 2 + 0.5) = \text{sign}(6.5) = 1$$

  Predicted class: 1

- Point 3: $X1 = 2$, $X2 = 2$

  $$y = \text{sign}(2 \cdot 2 + (-1) \cdot 2 + 0.5) = \text{sign}(4 - 2 + 0.5) = \text{sign}(2.5) = 1$$

  Predicted class: 1

1.2 (b) Drawing the Decision Boundary

The decision boundary can be defined using the weights and bias:

$$W1 \cdot X1 + W2 \cdot X2 + b = 0$$

Substituting the values:

$$2 \cdot X1 - 1 \cdot X2 + 0.5 = 0$$

$$X2 = 2 \cdot X1 + 0.5$$

This is the equation of the decision boundary. Below is the plot representing the decision boundary and the input points.

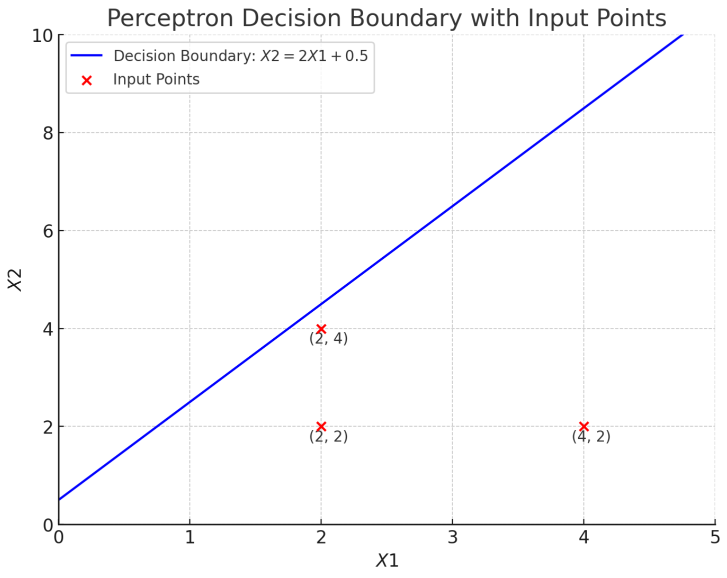

Figure 1: Perceptron Decision Boundary with Input Points

1.3 (c) Limitations of the Perceptron Algorithm

The perceptron algorithm has the following limitations:

- It can only solve linearly separable problems. If the data is not linearly separable, the algorithm will not converge.
- The solution provided by the perceptron is not unique. Different decision boundaries may exist that can separate the classes.
- The perceptron does not produce probabilities or confidence scores. It only provides a hard classification.

2 Question 2

For the neural network shown in the diagram, answer the following questions:

2.1 (a) How many neurons?

The neural network consists of the following neurons:

- Input layer: 4 neurons
- First hidden layer: 2 neurons
- Second hidden layer: 4 neurons
- Third hidden layer: 2 neurons
- Fourth hidden layer: 4 neurons
- Output layer: 2 neurons

Thus, the total number of neurons is $4 + 2 + 4 + 2 + 4 + 2 = 18$.

2.2 (b) How many hidden layers?

There are 4 hidden layers in the neural network.

2.3 (c) How many learnable parameters in total?

To calculate the number of learnable parameters, we count the weights and biases:

- Between the input layer (4 neurons) and the first hidden layer (2 neurons):
$$4 \times 2 = 8 \text{ weights}, \quad 2 \text{ biases} \quad \Rightarrow 8 + 2 = 10 \text{ parameters}$$
- Between the first hidden layer (2 neurons) and the second hidden layer (4 neurons):
$$2 \times 4 = 8 \text{ weights}, \quad 4 \text{ biases} \quad \Rightarrow 8 + 4 = 12 \text{ parameters}$$
- Between the second hidden layer (4 neurons) and the third hidden layer (2 neurons):
$$4 \times 2 = 8 \text{ weights}, \quad 2 \text{ biases} \quad \Rightarrow 8 + 2 = 10 \text{ parameters}$$
- Between the third hidden layer (2 neurons) and the fourth hidden layer (4 neurons):
$$2 \times 4 = 8 \text{ weights}, \quad 4 \text{ biases} \quad \Rightarrow 8 + 4 = 12 \text{ parameters}$$
- Between the fourth hidden layer (4 neurons) and the output layer (2 neurons):
$$4 \times 2 = 8 \text{ weights}, \quad 2 \text{ biases} \quad \Rightarrow 8 + 2 = 10 \text{ parameters}$$

Thus, the total number of learnable parameters is $10+12+10+12+10 = 54$.

3 Question 3: 1D Convolution

3.1 (a) Perform 1D Convolution

Given the array $[3, 4, 6, 3, 2, 3, 4]$ and filter $[1, 2, -1]$, we apply zero padding and compute the convolution:

- Padded array: $[0, 3, 4, 6, 3, 2, 3, 4, 0]$ - Convolution results:

$$[2, 5, 13, 10, 4, 4, 11]$$

4 Question 4: Output Length and Padding

For an array of length L and a filter of size F, the output length without padding is:

$$\text{Output length} = L - F + 1$$

To maintain the same output length as the input, padding P is required:

$$P = \frac{F - 1}{2}$$

With padding, the output length remains the same as L.

5 Question 5: 2D Convolution

We are given a 2D array and a 2x2 convolution filter.

5.1 (a) Perform 2D Convolution

Given the array:

$$\begin{bmatrix} 3 & 5 & 2 & 2 \\ 1 & -1 & 3 & -1 \\ 0 & 6 & 3 & 3 \\ 10 & -3 & 5 & 6 \end{bmatrix}$$

and the filter:

$$\begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix}$$

We compute the convolution for each region:

- Resulting convolution matrix:

$$\begin{bmatrix} 8 & 19 & 1 \\ 13 & 1 & 8 \\ -10 & 28 & 16 \end{bmatrix}$$

6 Question 6: AlexNet Parameters and Memory

For an AlexNet architecture with an input size of $512 \times 512 \times 3$, we calculate the number of parameters and the memory used by each layer as follows:

6.1 AlexNet Architecture Overview

- Input Layer: $512 \times 512 \times 3$ image.
- Conv Layer 1: 96 filters of size 11×11, stride $= 4$, padding $= 0$.
- Max Pooling Layer 1: 3×3 filter, stride $= 2$.
- Conv Layer 2: 256 filters of size 5×5, stride $= 1$, padding $= 2$.
- Max Pooling Layer 2: 3×3 filter, stride $= 2$.
- Conv Layer 3: 384 filters of size 3×3, stride $= 1$, padding $= 1$.
- Conv Layer 4: 384 filters of size 3×3, stride $= 1$, padding $= 1$.
- Conv Layer 5: 256 filters of size 3×3, stride $= 1$, padding $= 1$.
- Max Pooling Layer 3: 3×3 filter, stride $= 2$.
- Fully Connected Layer 1: 4096 neurons.
- Fully Connected Layer 2: 4096 neurons.
- Output Layer: 1000 neurons (for 1000 classes in ImageNet).

6.2 Step-by-Step Calculations

6.2.1 Conv Layer 1:

- Input size: $512 \times 512 \times 3$
- Filter size: $11 \times 11 \times 3$
- Number of filters: 96
- Stride: 4
- Output size: $126 \times 126 \times 96$

  Number of parameters:

  $$(11 \times 11 \times 3 + 1) \times 96 = 34,944 \text{ parameters}$$

  Memory required:

  $$126 \times 126 \times 96 = 1,522,176 \text{ activations}$$

6.2.2 Max Pooling Layer 1:

- Pool size: 3×3, stride $= 2$.
- Output size: $63 \times 63 \times 96$

  Memory required:

$$63 \times 63 \times 96 = 380,352 \text{ activations}$$

6.2.3 Conv Layer 2:

- Input size: $63 \times 63 \times 96$
- Filter size: $5 \times 5 \times 96$
- Number of filters: 256
- Output size: $63 \times 63 \times 256$ (with padding)

  Number of parameters:

$$(5 \times 5 \times 96 + 1) \times 256 = 614,656 \text{ parameters}$$

  Memory required:

$$63 \times 63 \times 256 = 1,016,832 \text{ activations}$$

6.2.4 Max Pooling Layer 2:

- Pool size: 3×3, stride $= 2$.
- Output size: $31 \times 31 \times 256$

  Memory required:

$$31 \times 31 \times 256 = 246,016 \text{ activations}$$

6.2.5 Conv Layer 3:

- Input size: $31 \times 31 \times 256$
- Filter size: $3 \times 3 \times 256$
- Number of filters: 384
- Output size: $31 \times 31 \times 384$

  Number of parameters:

$$(3 \times 3 \times 256 + 1) \times 384 = 885,120 \text{ parameters}$$

  Memory required:

$$31 \times 31 \times 384 = 369,504 \text{ activations}$$

6.2.6 Conv Layer 4:

- Input size: $31 \times 31 \times 384$
- Filter size: $3 \times 3 \times 384$
- Number of filters: 384
- Output size: $31 \times 31 \times 384$

  Number of parameters:

  $$(3 \times 3 \times 384 + 1) \times 384 = 1,327,104 \text{ parameters}$$

  Memory required:

  $$31 \times 31 \times 384 = 369,504 \text{ activations}$$

6.2.7 Conv Layer 5:

- Input size: $31 \times 31 \times 384$
- Filter size: $3 \times 3 \times 384$
- Number of filters: 256
- Output size: $31 \times 31 \times 256$

  Number of parameters:

  $$(3 \times 3 \times 384 + 1) \times 256 = 884,992 \text{ parameters}$$

  Memory required:

  $$31 \times 31 \times 256 = 246,016 \text{ activations}$$

6.2.8 Max Pooling Layer 3:

- Pool size: 3×3, stride = 2.
- Output size: $15 \times 15 \times 256$

  Memory required:

  $$15 \times 15 \times 256 = 57,600 \text{ activations}$$

6.2.9 Fully Connected Layer 1:

- Input size: $15 \times 15 \times 256 = 57,600$
- Output size: 4096 neurons.

  Number of parameters:

  $$57,600 \times 4096 + 4096 = 236,560,896 \text{ parameters}$$

  Memory required:

  $$4096 \text{ activations}$$

6.2.10 Fully Connected Layer 2:

- Input size: 4096
- Output size: 4096

  Number of parameters:

  $$4096 \times 4096 + 4096 = 16,781,312 \text{ parameters}$$

  Memory required:

  $$4096 \text{ activations}$$

6.2.11 Output Layer:

- Input size: 4096
- Output size: 1000 neurons.

  Number of parameters:

  $$4096 \times 1000 + 1000 = 4,097,000 \text{ parameters}$$

  Memory required:

  $$1000 \text{ activations}$$

6.3 Final Results

- Total parameters:

  $236,560,896+16,781,312+4,097,000+884,992+1,327,104+885,120+614,656+34,944 = 260,185,024$

  parameters.