EE708, Assignment 5 Submission

Dhruv Gupta, Roll no: 240354

Problem 1

Question

A neural network consists of N input neurons, H hidden neurons, and C output neurons. How many total weights are there in the network (excluding biases)?

a If the connections between input and hidden layers, and between hidden and output layers.

b If the connections between input and hidden layers, between hidden and output layers, and input and output layers.

Answer

Given:

- Number of input neurons: N
- Number of hidden neurons: H
- Number of output neurons: C

(a) Connections: Input $\rightarrow$ Hidden and Hidden $\rightarrow$ Output

We calculate the number of weights required between each layer:

- From input to hidden layer: $N \times H$ weights
- From hidden to output layer: $H \times C$ weights

Total number of weights:

$$\boxed{NH + HC}$$

(b) Additional connections: Input $\rightarrow$ Output also included

If we additionally connect the input layer directly to the output layer, we must include $N \times C$weights from input to output layer.

Total number of weights now becomes:

$$\boxed{NH + HC + NC}$$

Problem 2

Question

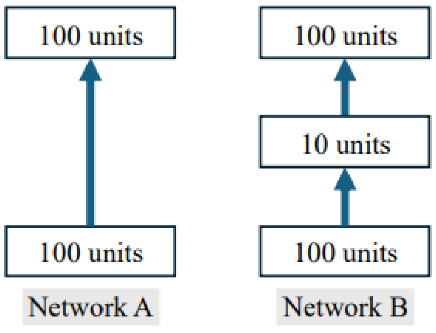
Consider the following two multilayer perceptrons, where all layers use linear activation functions.

a Give one advantage of Network A over Network B.

b Give one advantage of Network B over Network A.

Answer

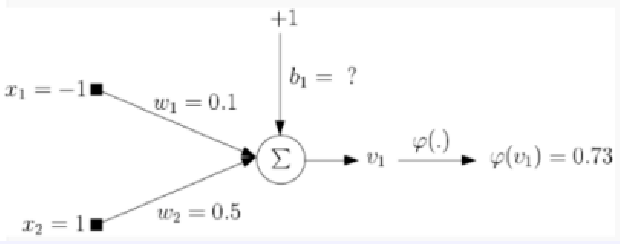
a) Network A can represent any linear transformation from a 100-dimensional input to a 100-dimensional output, as it directly implements a 100×100 weight matrix. *Hence, Network A is more expressive in terms of representing all possible linear mappings between input and output.*
(b) One advantage of Network B over Network A Network B uses a low-dimensional hidden layer (10 units), which reduces the number of parameters and enforces a low-rank approximation of the linear mapping. *Therefore, Network B offers better regularization, improved generalization, and reduced computational cost.*



Problem 3

Question

Consider a two-input neuron with a logistic activation function with slope parameter $a = 2$. Let the inputs be $[-1, 1]$ and the weights are $[0.1, 0.5]$ respectively. The output of the neuron is 0.73. What is the value of the bias b_1?



Answer

Given: Inputs $x = [-1, 1]$, weights $w = [0.1, 0.5]$, slope $a = 2$, output $= 0.73$.

$$z = -0.1 + 0.5 + b_1 = 0.4 + b_1$$

Activation:

$$0.73 = \frac{1}{1 + e^{-2(0.4+b_1)}} \Rightarrow e^{-2(0.4+b_1)} \approx 0.36986$$

Solving:

$$-2(0.4 + b_1) = \ln(0.36986) \approx -0.995 \Rightarrow b_1 = \frac{0.995}{2} - 0.4 = 0.0975$$

$$\therefore b_1 \approx 0.0975$$

Problem 4

Question

Show the perceptron that calculates:

a NOT of its input.

b NAND of its two inputs.

Answer

(a) NOT Gate:

Let the input be $x \in \{0, 1\}$. Define the perceptron with:

- Weight: $w = -1.37$
- Bias: $b = 0.69$
- Activation: Step function

Then the output is:

$$\text{Step}(w \cdot x + b) = \begin{cases} 1 & \text{if } x = 0 \Rightarrow -1.37 \cdot 0 + 0.69 = 0.69 > 0 \\ 0 & \text{if } x = 1 \Rightarrow -1.37 \cdot 1 + 0.69 = -0.68 < 0 \end{cases}$$

Hence, this configuration computes the logical NOT function.

(b) NAND Gate:

Let the inputs be $x_1, x_2 \in \{0, 1\}$. Define the perceptron with:

- Weights: $w_1 = -0.9$, $w_2 = -0.9$
- Bias: $b = 1.29$
- Activation: Step function

Then the weighted sum is:

$$z = -0.9x_1 - 0.9x_2 + 1.29$$

Evaluating all inputs:

- $(0, 0) : z = 1.29 \Rightarrow 1$
- $(0, 1)$ or $(1, 0) : z = -0.9 + 1.29 = 0.39 \Rightarrow 1$
- $(1, 1) : z = -1.8 + 1.29 = -0.51 \Rightarrow 0$

This configuration correctly computes the logical NAND function.

Problem 5

Question

Show the perceptron that calculates the Parity of its three inputs. (The parity of a binary number refers to whether the number of 1-bits in its binary representation is even or odd.)

Answer

Using Two Perceptrons to Compute Parity We need to get the parity function for three binary inputs:

$$\text{Parity}(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$$

This can be achieved in two stages: first compute $x_1 \oplus x_2$, then XOR the result with x_3.

Perceptron 1: Compute $z = x_1 \oplus x_2$

We define a small MLP to compute XOR:

- Hidden Layer:
  - $h_1 = \text{step}(0.91x_1 + 0.88x_2 - 0.43)$
  - $h_2 = \text{step}(-0.97x_1 - 0.92x_2 + 1.67)$
- Output:

$$z = \text{step}(1.02h_1 + 1.01h_2 - 1.49)$$

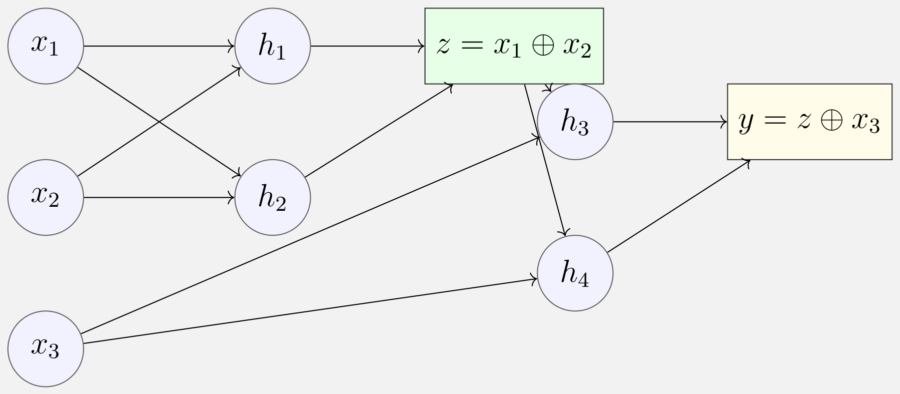
Perceptron 2: Compute $y = z \oplus x_3$

Again, using the XOR architecture:

- Hidden Layer:
  - $h_3 = \text{step}(1.07z + 0.95x_3 - 0.51)$
  - $h_4 = \text{step}(-1.11z - 1.03x_3 + 1.59)$
- Output:

$$y = \text{step}(1.03h_3 + 0.99h_4 - 1.41)$$

Final Output: $y = 1$ if the number of 1's among x_1, x_2, x_3 is odd (i.e., parity is 1), else 0.



Problem 6

Question

Derive the weight update equations for an MLP that uses ReLU in its hidden units. Assume MLP with one hidden layer of H units and one output trained for regression.

Answer

Let:

$$\mathbf{x} \in \mathbb{R}^m \quad \text{(input vector with } m \text{ features)}$$

$$\mathbf{W_1} \in \mathbb{R}^{H \times m} \quad \text{(weights from input to hidden)}$$

$$\mathbf{a} \in \mathbb{R}^H \quad \text{(bias vector for hidden layer)} \qquad \mathbf{s} = \mathbf{U}\mathbf{x} + \mathbf{a}$$

$$\mathbf{h} = \text{ReLU}(\mathbf{s}) \qquad \mathbf{v} \in \mathbb{R}^{1 \times H} \quad \text{(weights from hidden to output)}$$

$$c \in \mathbb{R} \quad \text{(output bias)} \hat{y} = \mathbf{v} \cdot \mathbf{h} + c y \in \mathbb{R} \quad \text{(true target)}$$

$$\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2 \quad \text{(squared error loss)}$$

Gradients:
Output layer:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{v}} = (\hat{y} - y) \cdot \mathbf{h}^\top \qquad \frac{\partial \mathcal{L}}{\partial c} = \hat{y} - y$$

Hidden layer:

$$\text{ReLU}'(s_j) = \begin{cases} 1 & \text{if } s_j > 0 \\ 0 & \text{otherwise} \end{cases} \qquad \boldsymbol{\delta}^{(1)} = (\hat{y} - y) \cdot \mathbf{v}^\top \circ \text{ReLU}'(\mathbf{s})$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}} = \boldsymbol{\delta}^{(1)} \cdot \mathbf{x}^\top \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{a}} = \boldsymbol{\delta}^{(1)}$$

Weight Updates (learning rate η):

$$\mathbf{v} := \mathbf{v} - \eta \cdot (\hat{y} - y) \cdot \mathbf{h}^\top \qquad c := c - \eta \cdot (\hat{y} - y)$$

$$\mathbf{U} := \mathbf{U} - \eta \cdot \boldsymbol{\delta}^{(1)} \cdot \mathbf{x}^\top \qquad \mathbf{a} := \mathbf{a} - \eta \cdot \boldsymbol{\delta}^{(1)}$$

Problem 7

Question

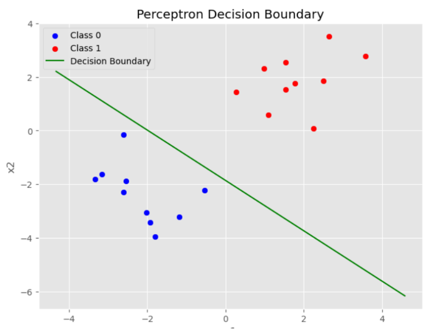
Perceptron Learning Algorithm for Binary Classification: For the dataset in A5-P1.csv, implement a Perceptron using gradient descent to classify the data points.

- Activation function: step
- Weight initialization: random (using numpy package)
- Learning rate: 0.01
- Number of weight update iterations: 20

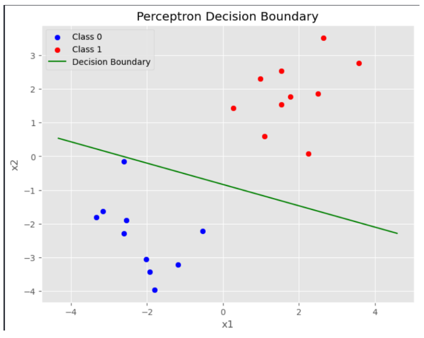
Visualize the decision boundary by plotting the dataset points and the separating line. (Mark data points for different classes with different colors.) Rerun the code with different random weight initializing to understand the importance of initialization.

Output

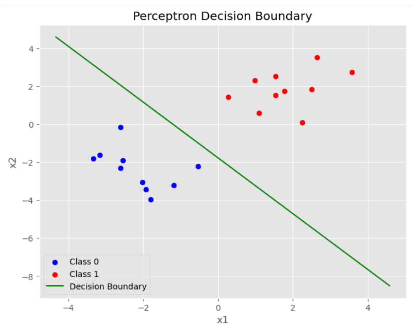
Some instances of random weight initialization and 20 iterations are:



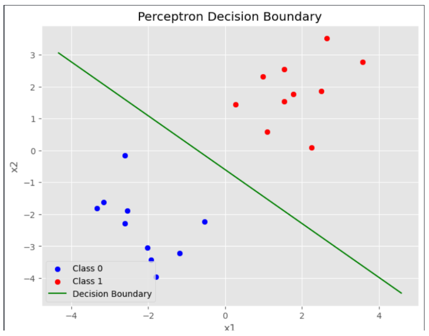
iteration 1



iteration 2



iteration 3



iteration 4

Problem 8

Question

Neural Network from Scratch

Using the Iris dataset, implement a feedforward neural network with one hidden layer (with five neurons) and train it using Stochastic Gradient Descent (SGD) to classify the species correctly.

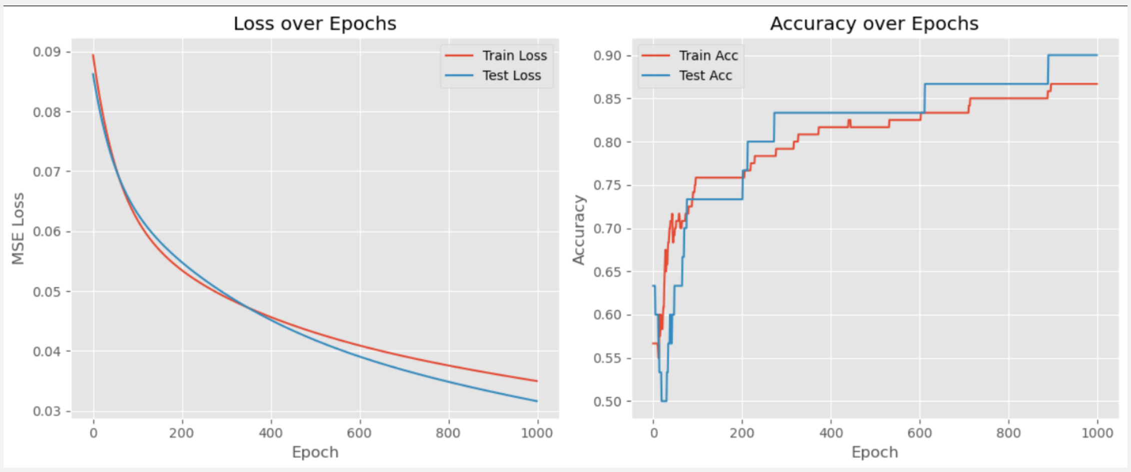
- Load the Iris dataset (from sklearn.datasets.load_iris().)
- Split the dataset into training (80%) and testing (20%) subsets.
- Standardize the features to have zero mean and unit variance.
- Activation functions: tanh for the hidden layer and softmax for the output layer.
- Loss function: Mean Squared Error (MSE)
- Weight initialization: random
- Learning rate: 0.01
- Number of weight update iterations: 1000

Implement forward and backward propagation.

a Plot the training and testing loss vs. epochs.

b Plot the training and testing accuracy over epochs.

Output



Plotted the loss and accuracy of the model over 1000 epochs.