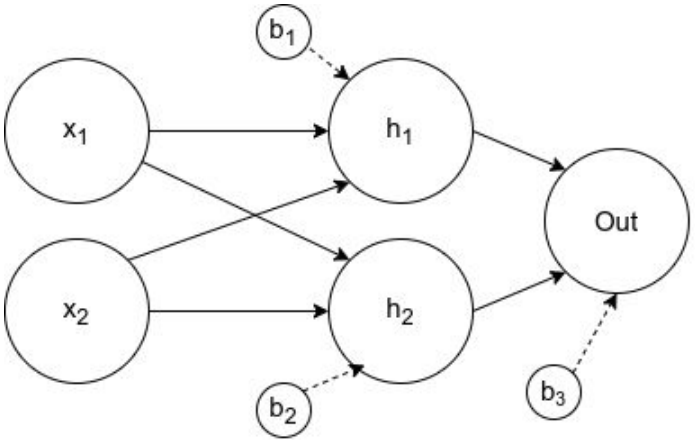
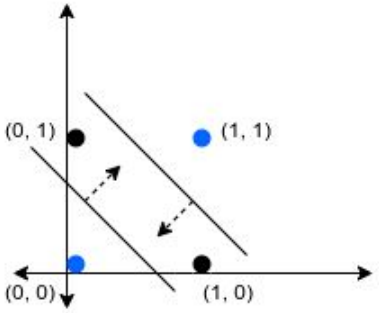
Sheet 3

Q1) A multi-layer perceptron network is used to solve the XOR problem. It has two nodes in the hidden layer. All nodes have a sign activation function. Tune mathematically the weighting parameters of such a network. Visualize the classification boundaries.

We shall draw our Neural Network first based on the XOR table

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0





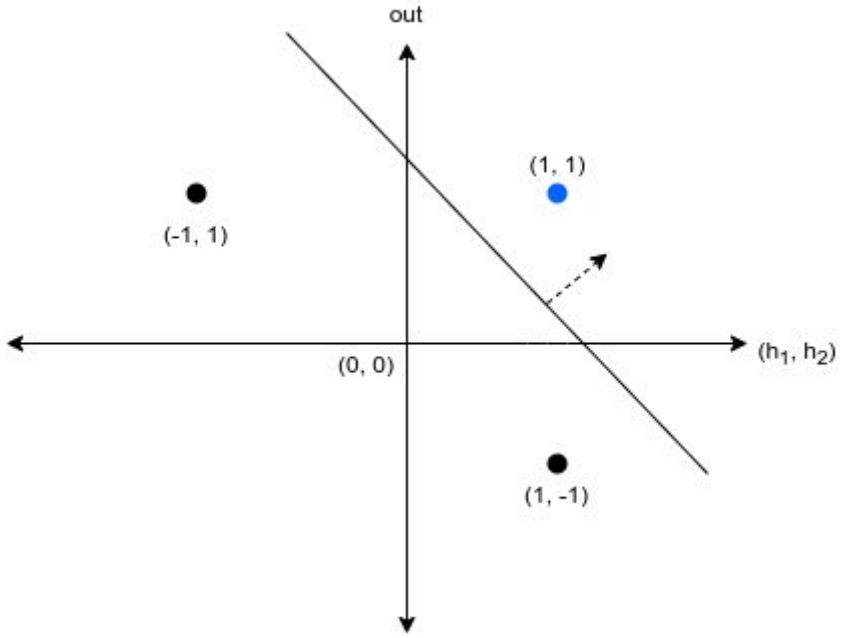
$$x = [x_1\ x_2]\ \&\ h\ =\ [h_1\ h_2]$$

- $h_1 = Sign(w_{h_1} * x + b_1)$
- $h_2 = Sign(w_{h_2} * x + b_2)$
- $out = Sign(w_{out} * h + b_3)$

Important Note: $Sign(0) = 1$

From the graph we can conclude that $w_{h_1} = [-1,-1]$ & $b_1 = 1.5$,
$w_{h_2} = [1,1]$ & $b_2 = -0.5$

We can redo what we made for h_1, h_2 with out by redrawing the outputs of the previous layer with their labels and estimating the new classifier boundary.



From the graph we can conclude that $w_{out} = [1,1]$ & $b_3 = -1$

Q2) Find the derivative of the sigmoid function. List the advantages and disadvantages of this function.

$$
\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] \\
&= \frac{d}{dx}\left(1+\mathrm{e}^{-x}\right)^{-1} \\
&= -(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right)
\end{aligned}
$$

$$d(x)/dx = (x) * (1 - (x))$$

Advantages:

1. Ease of differentiation & nonlinearity
2. Suitable for Backward propagation
3. Encodes probabilistic output

Disadvantages:

1. Vanishing Gradient Problem (When the activation function is close to the saturation region, the change is too low, the derivative is close to 0)
2. Not Zero Centered (Limitated with Normalization)

Q3) Find the derivative of the tanh function. List the advantages and disadvantages of this function.

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\frac{d}{dx}tanh(x) = \frac{d}{dx}\left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)$$

$$= \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2}$$

$$= 1 - \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)^2$$

$$= 1 - \tanh^2(x)$$

Advantages:

1. Zero Centered
2. Easy differentiated

Disadvantages:

It also has the problem of gradient disappearance.

Q4) The following data set of 2D points, {(-1, -1), (+1, -1), (-1, +1), (+1, +1)} and their corresponding labels {+1, +1, +1, -1} is trained with a single neuron neural network.

- Find the total loss if, **W** = [0.5 -0.3 0.8] T if the activation function is the identity

$$g(X) = 0.5x_1 - 0.3x_2 + 0.8$$

$$I(x) = x,\ Y(x) = I(g(x))$$

$$L(y, y') = log_e(1 + exp(-y * y'))$$

X_n	Y_n	$L(y,y')$

[-1 -1]	0.6	0.43
[1 -1]	1.6	0.18
[-1 1]	0.0	0.69
[1 1]	1.0	1.313
		Total Loss=2.63

- Find the total loss if, **W** = [0.5 -0.3 0.8] T if the activation function is sigmoid.

$$g(X) = 0.5x_1 - 0.3x_2 + 0.8$$
$$Sigmoid(x) = 1/(1 + exp(-x))$$
$$Y(x) = Sigmoid(g(x))$$
$$L(y, y') = -log(|y/2 - 0.5 + y'|)$$

X_n	Y_n	$L(y, y')$
[-1 -1]	0.645	0.43
[1 -1]	0.832	0.183
[-1 1]	0.5	0.69
[1 1]	0.731	1.313
		Total Loss=2.63

- Find the total loss if, **W** = [0.5 -0.3 0.8] T if the activation function is the sign function. Use the bipolar perceptron criteria.

$$g(X) = 0.5x_1 - 0.3x_2 + 0.8$$
$$Sign(x) = 1 \text{ if } x \geq 0, \ -1 \text{ otherwise}$$
$$Y(x) = Sign(g(x))$$
$$L(y, g(x)) = max(0, -y * g(x))$$

X_n	Y_n	$L(y, g(x))$

[-1 -1]	1	0
[1 -1]	1	0
[-1 1]	1	0
[1 1]	1	1
		Total Loss=1

- Find the total loss if, **W** = [0.5 -0.3 0.8] T if the activation function is the sign function. Use the SVM criterion

$$g(X) = 0.5x_1 - 0.3x_2 + 0.8$$
$$Sign(x) = 1 \; if \; x \geq 0, \; -1 \; otherwise$$
$$Y(x) = Sign(g(x))$$
$$L(y, \; g(x)) = max(0, \; 1 - y * g(x))$$

X_n	Y_n	$L(y, g(x))$
[-1 -1]	1	0.4
[1 -1]	1	0.0
[-1 1]	1	1.0
[1 1]	1	2.0
		Total Loss=3.4

Q5) Given a data set of RGB colors, {(0, 0, 0), (255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (0, 255, 255), (255, 0, 255), (255, 255, 255)} and their corresponding labels {+1, +1, +1, -1, +1, -1, -1, +1}.

Recall from Slides 2 & 3 the GD Algorithm

The Gradient Descent Algorithm

Step 0: $Select\ X_0 \in R^n,\ Set\ \alpha, and\ i = 0$

Step 1: Compute $\nabla f(X_i)$

Step 2: if $||\nabla f(X_i)|| < \varepsilon, Stop$
Otherwise Go To Step 3

Step 3: Compute $X_{i+1} = X_i - \alpha \nabla f(X_i)$

Step 4: Update i=i+1

Step 5: Go To Step 1

And remember that we will adopt it for Online and Batch Gradient Descent with different loss functions.
Important Note: The solver of this sheet only updates the table when the weights are updated.

- Using the bipolar perceptron criterion with gradient descent, find and visualize a classification boundary adopting the online training.

We will declare a new variable called **delta** to accumulate in it the gradient of the loss function with different input points and we will substitute alpha to be 1 and not to forget averaging the errors over the dataset, finally we will assume the bias to be the first value in the weights vector.

Online Training

Input: $(\mathbf{X}_m, y_m)$, m = 1, 2, …,N

Set $\mathbf{W} = [0, 0,.., 0]^T$

Repeat

  delta $= [0, 0,.., 0]^T$

  for m = 1 to N do

    if $y_m \mathbf{W}.\mathbf{X}_m <= 0$

      delta = **delta** $- y_m \mathbf{X}_m$

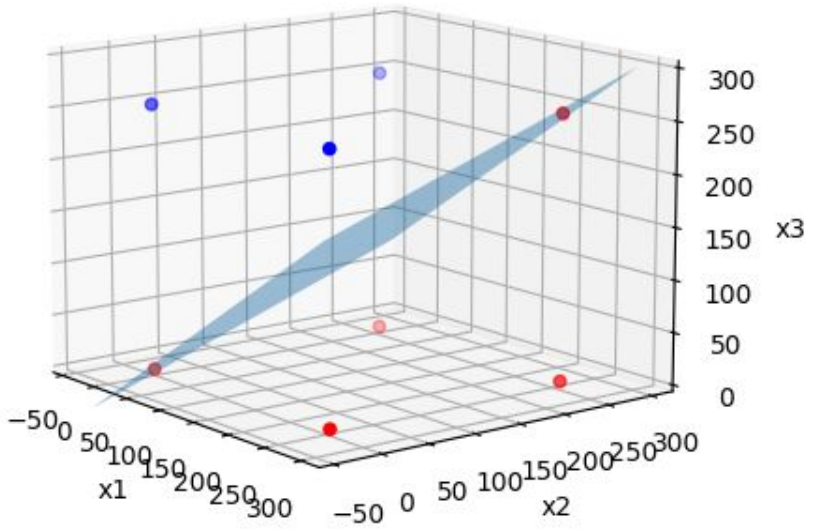
      W = **W** – **delta** / N

Until $||\mathbf{delta}|| < \epsilon$

In Online Training we update the weights in the for loop with the gradient of the perceptron criterion ($-y_m x_m$) and check with every iteration on the while loop to see if the delta (the grad of the loss) to be less than epsilon or not to stop the algorithm.

While	For	X_m	y_m	$y_m W.X_m$	$-y_m X_m$	delta	W
1	1	(1, 0, 0, 0)	1	0	[-1, 0, 0, 0]	[-1, 0, 0, 0]	[0.125,0,0,0]
1	4	(1,0,0,255)	-1	-0.125	[1,0,0,255]	[0,0,0,255]	[0.125,0,0,-31.875]
1	8	(1,255,255,255)	1	-8128	[-1,-255,-255,-255]	[-1,-255,-255,0]	[0.25,31.875,31.875,-31.875]
2	6	(1,0,255,255)	-1	-0.25	[1,0,255,255]	[1,0,255,255]	[0.125,31.875,0.0,-63.75]

2	8	(1,255,255,255)	1	-8128	[-1,-255,-255,-255]	[0,-255,0,0]	[0.125,63.75,0.0,-63.75]
3	7	(1,255,0,255)	-1	-0.125	[1,255,0,255]	[1,255,0,255]	[0.0,31.875,0.0,-95.625]
3	8	(1,255,255,255)	1	-16256.25	[-1,-255,-255,-255]	[0,0,-255,0]	[0.0,31.875,31.875,-95.625]
4	1	(1, 0, 0, 0)	1	0	[-1,0,0,0]	[-1,0,0,0]	[0.125,31.875,31.875,-95.625]
4	8	(1,255,255,255)	1	-8128	[-1,-255,-255,-255]	[-2,-255,-255,-255]	[0.375,63.75,63.75,-63.75]
5	6	(1,0,255,255)	-1	-0.375	[1,0,255,255]	[1,0,255,255]	[0.25,63.75,31.875,-95.625]



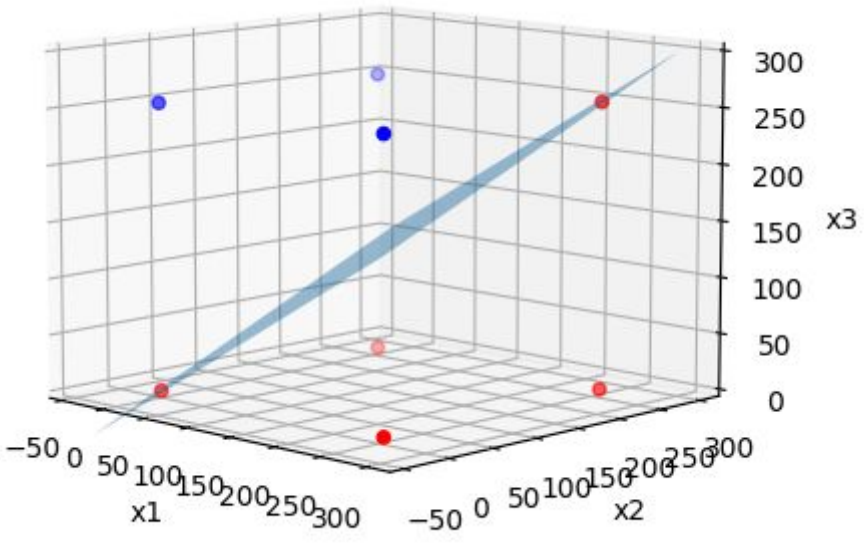
$$weights = [0.25 \quad 63.75 \quad 31.875 \quad -95.625]$$

- Using the bipolar SVM criterion with gradient descent, find and visualize a classification boundary adopting the online training.

The solution will be the same as above with one minor change in the gradient of the loss function, instead of the if condition be in zero it will be in one because of the soft margin we added to the precepton loss.

$$L = max(0, 1 - y_m W x_m)$$

While	For	Xm	ym	yWx	-yx	delta	Weights
1	1	[1, 0, 0, 0]	1	0	[-1, 0, 0, 0]	[-1, 0, 0, 0]	[0.125, 0.0, 0.0, 0.0]
1	2	[1, 255, 0, 0]	1	0.125	[-1, -255, 0, 0]	[-2, -255, 0, 0]	[0.375, 31.875, 0.0, 0.0]
1	3	[1, 0, 255, 0]	1	0.375	[-1, 0, -255, 0]	[-3, -255, -255, 0]	[0.75, 63.75, 31.875, 0.0]
1	4	[1, 0, 0, 255]	-1	-0.750	[1, 0, 0, 255]	[-2, -255, -255, 255]	[1.0, 95.625, 63.75, -31.875]
1	6	[1, 0, 255, 255]	-1	-8129.125	[1, 0, 255, 255]	[-1, -255, 0, 510]	[1.125, 127.5, 63.75, -95.625]
1	7	[1, 255, 0, 255]	-1	-8129.250	[1, 255, 0, 255]	[0, 0, 0, 765]	[1.125, 127.5, 63.75, -191.25]



Using Online Training and SVM Criterion,

$$weights = [1.125 \quad 127.5 \quad 63.75 \quad -191.25]$$

- Using the bipolar SVM criterion with gradient descent, find and visualize a classification boundary adopting the batch training.

Batch Training

Input: $(\mathbf{X}_m, y_m)$, m = 1, 2, …,N

Set $\mathbf{W} = [0, 0,.., 0]^T$

Repeat

  delta $= [0, 0,.., 0]^T$

  for m = 1 to N do

    if $y_m \mathbf{W}.\mathbf{X}_m <= 0$

      delta = **delta** $- y_m \mathbf{X}_m$

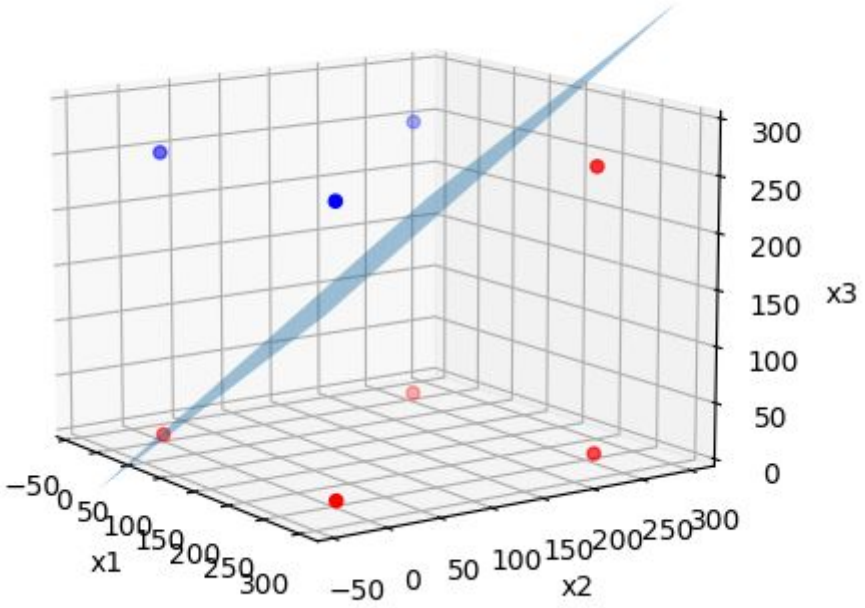
  delta = **delta** / N

  W = **W** – **delta**

Until ||**delta**|| < ϵ

Same as the previous questions, but this time the weights update occurs with every while loop iteration.

While	Xm	ym	yWx	-yx	delta	Weights
1	[1, 255, 255, 255]	1	0	[-1, -255, -255, -255]	[-0.25, -63.75, -63.75, 63.75]	[0.25, 63.75, 63.75, -63.75]
2	[1, 255, 255, 255]	1	16256.500	[-1, -255, -255, -255]	[0.125, 31.875, 31.875, 63.75]	[0.125, 31.875, 31.875, -127.5]
3	[1, 255, 255, 255]	1	-16256.125	[-1, -255, -255, -255]	[-0.25, -31.875, -31.875, -31.875]	[0.375, 63.75, 63.75, -95.625]
4	[1, 255, 255, 255]	1	8128.500	[-1, -255, -255, -255]	[-0.125, 0.0, 0.0, 0.0]	[0.5, 63.75, 63.75, -95.625]



Using Batch Training and SVM Criterion,

$$weights = [0.5 \quad 63.75 \quad 63.75 \quad -95.625]$$

Q6) Given a data set, D = {(X1, y1), (X2, y2)... (XN, yN)} of labelled feature vectors where X is the feature vector and y is the class label. Note that y ϵ {1, 2… m} and m represents the number of categories. A multilayer neural network is used for classification with a soft-max layer.

- What is the number of nodes in the input layer?

  R Nodes (where R is the size of the feature vector X).

- What is the number of nodes in the output layer?

  M nodes

- Write an expression for the loss function using sum of squared differences.

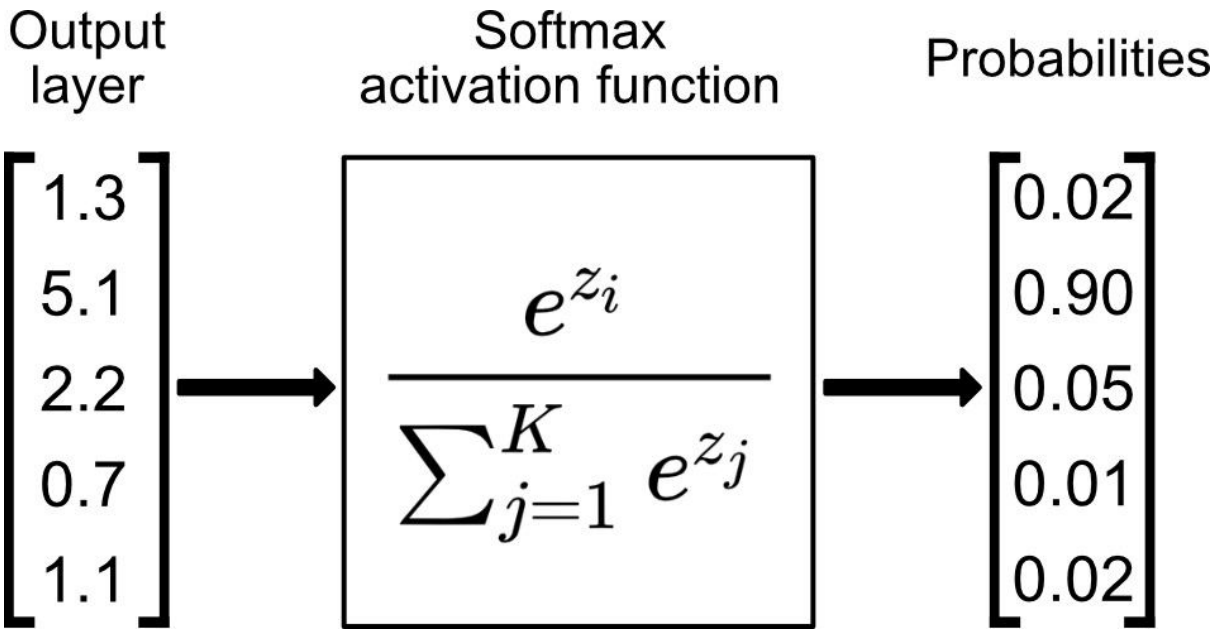
  $J(x,y) = \Sigma^{N}_{i=1}(X_i - y_i)^2$

- Write an expression for the loss function using the log-likelihood criterion

  $L = -log(y'_r)$ → where y'_r is the softmax output of the ground truth (correct label)

Q7) A neural network is designed to classify the handwritten numerical digits. It has 10 ten output nodes to represent the image categories 0, 1, 2… 9. Assume that the output vector for a certain image input is produced as (3.1, -9.3, 7, 8.7, 3.6, 5.2, 4.7, -2.2, 3.1, -6.6)

- Find the outputs of the soft-max layer.



Results = [2.97×10^{-3}, 1.22×10^{-8}, 0.14, 0.8, 4.89×10^{-3}, 0.02, 0.0147, 1.4829×10^{-5}, 2.97×10^{-3}, 1.82×10^{-7}]

- Find the log-likelihood loss if the correct class label is the fourth.

$$L = -log(y'_r)$$

$$L = -log(y'_r) = -log(0.8) = 0.09691$$

- Find the log-likelihood loss if the correct class label is the second.

$$L = -log(y'_r)$$

$$L = -log(y'_r) = -log(1.22e^{-8}) = 7.9136$$

Notice: When the probability of the correct answer was large the log likelihood loss was small and vice versa.