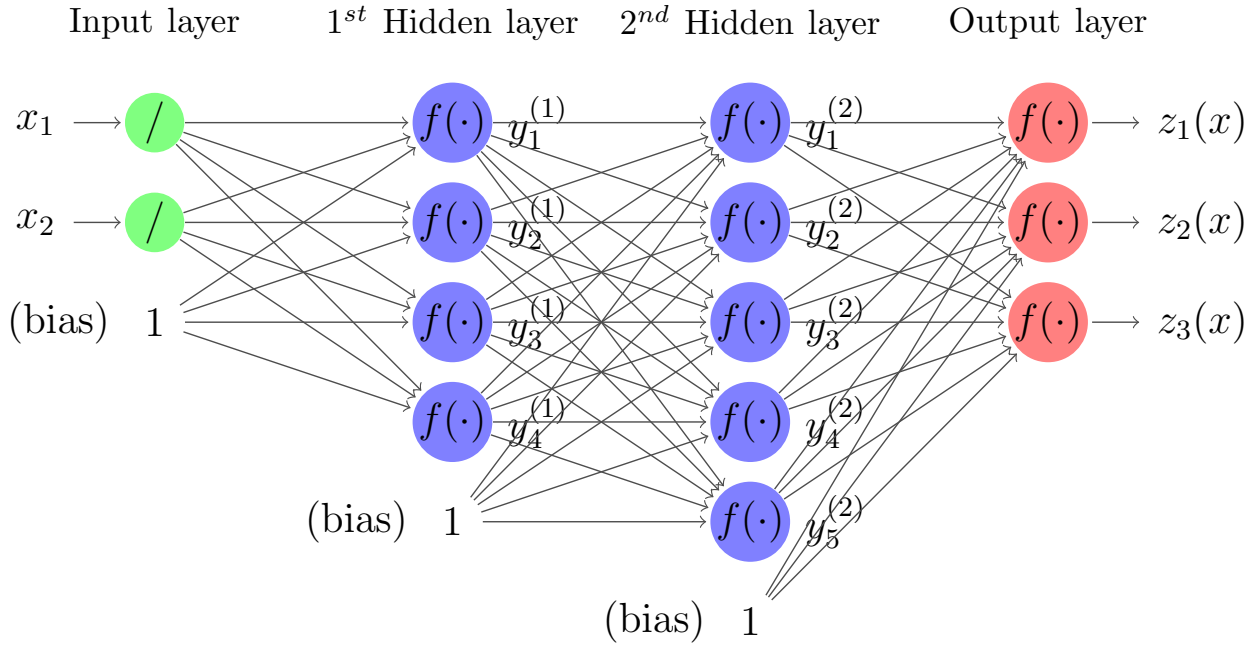
Pattern Recognition, Neural Networks and Deep Learning (7CCSMPNN)

Tutorial 6: Solutions

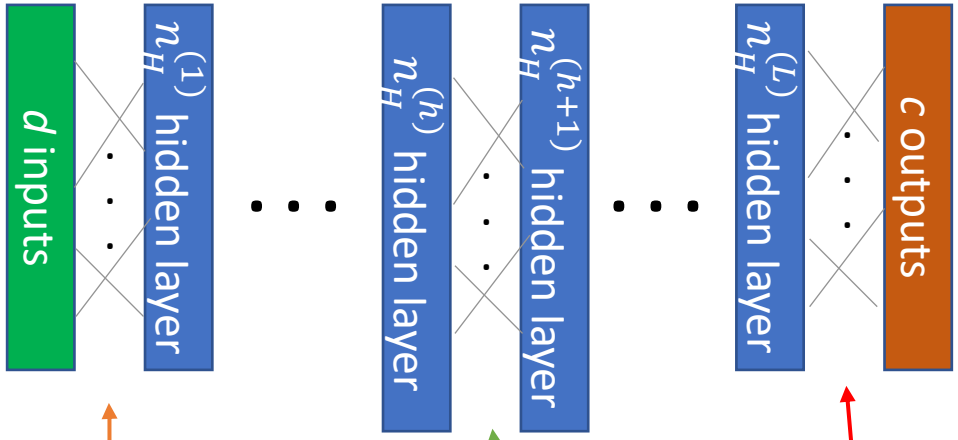
Q1. Draw the diagram of a 2-input-3-output multi-layer feedforward fully-connected neural network including biases. It has 2 hidden layers where the first hidden layer has 4 hidden units and the second hidden layer has 5 hidden units. How many connection weights it has in total?

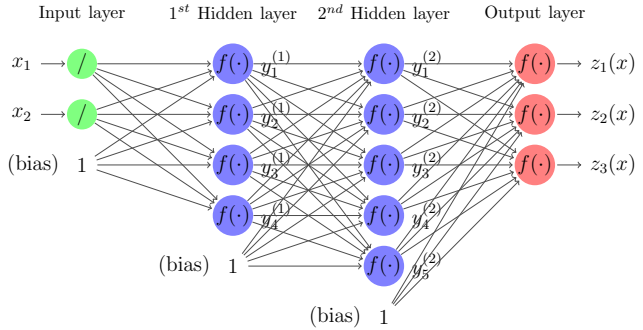


Number of weights in between the 1^{st} input layer and the 1^{st} hidden layer: $3 \times 4 = 12$.
Number of weights in between the 1^{st} hidden layer and the 2^{nd} hidden layer: $5 \times 5 = 25$.
Number of weights in between the 2^{nd} hidden layer and the output layer: $6 \times 3 = 18$.
Total number of weights: $12 + 25 + 18 = 55$.

Why multi-layers are needed?

Q2. Derive the equation for a multi-layer feedforward fully-connected multilayer neural network including biases in terms of d (number of inputs), $n_H^{(h)}$ (number of hidden units in the h^{th} hidden layer, $h = 1, 2, \ldots, L$), L (number of hidden layers) and c (number of outputs).





Number of weights in between the 1^{st} input layer and the 1^{st} hidden layer:

$$(d+1) \times n_H^{(1)}.$$

Number of weights in between the h^{st} hidden layer and the $(h+1)^{th}$ hidden layer:

$$(n_H^{(h)} + 1) \times n_H^{(h+1)}, h = 1, 2, \ldots, L-1.$$

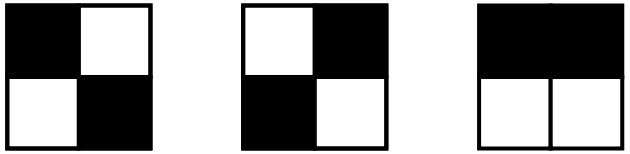
Number of weights in between the L^{th} hidden layer and the output layer:

$$(n_H^{(L)} + 1) \times c.$$

Total number of weights:

$$(d+1) \times n_H^{(1)} + \sum_{h=1}^{L-1} \left((n_H^{(h)} + 1) \times n_H^{(h+1)} \right) + (n_H^{(L)} + 1) \times c$$

Q3. Fig. 1 shows 3 patterns of a classification problem. A feedforward fully-connected neural network is employed to classify the patterns. How many inputs and outputs should be used for the neural network? List the input patterns and target output in a table.



(a) Pattern 1. (b) Pattern 2. (c) Pattern 3.

Figure 1: 3 patterns.

One possible input assignment is shown in Fig. 2.

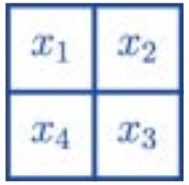
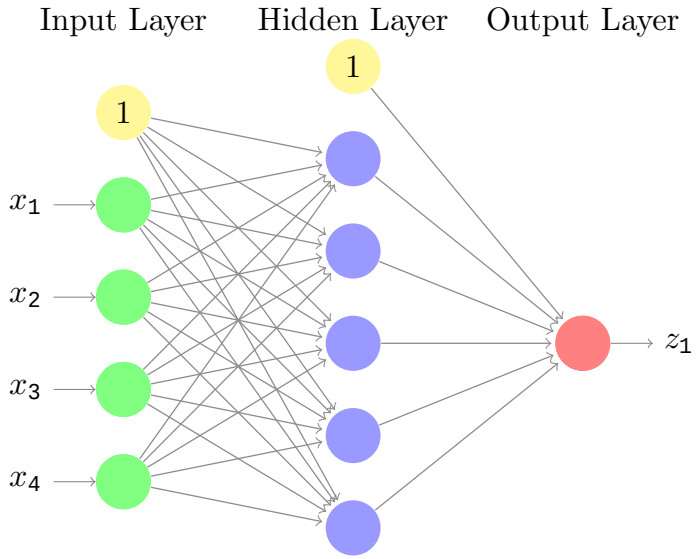


Figure 2: Assignment of input $\mathbf{x}$.



A 4-input-1-output feedforward fully-connected neural network can used for this classification problem.

0: Unfilled
1: Filled

x_1	x_2	x_3	x_4	Target output
1	0	1	0	1
0	1	0	1	2
1	1	0	0	3
Otherwise				0

- What activation function should be used in the output node?
- Or which activation function should NOT be used in the output node?
- How to determine the configuration of neural network?

Q4. Fig. 2 shows a 4-3-2 neural network, which is used to classify the patterns shown in Fig. 1. The assignment of input $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T$ is shown in Fig. 3 where 0 represents an unfilled grid and 1 represents a filled grid. The activation functions in the input layer, hidden layer and output layer are linear function, symmetric tangent sigmoid function and logarithmic sigmoid function, respectively. Given that the weight and bias matrices as:

$$\mathbf{W}_{ji} = \begin{bmatrix} -0.7057 & 1.9061 & 2.6605 & -1.1359 \\ 0.4900 & 1.9324 & -0.4269 & -5.1570 \\ 0.9438 & -5.4160 & -0.3431 & -0.2931 \end{bmatrix}, \mathbf{W}_{j0} = \begin{bmatrix} 4.8432 \\ 0.3973 \\ 2.1761 \end{bmatrix},$$

$$\mathbf{W}_{kj} = \begin{bmatrix} -1.1444 & 0.3115 & -9.9812 \\ 0.0106 & 11.5477 & 2.6479 \end{bmatrix}, \mathbf{W}_{k0} = \begin{bmatrix} 2.5230 \\ 2.6463 \end{bmatrix},$$

determine the output patterns representing the 3 classes.

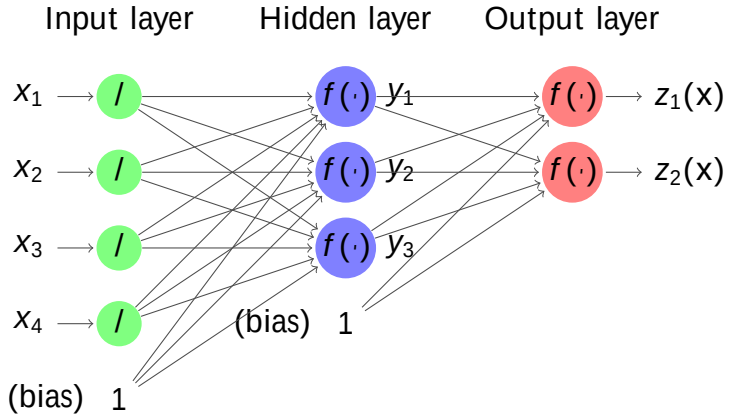


Figure 2: A diagram of 4-3-2 feedforward fully-connected neural network.

x_1	x_2
x_4	x_3

Figure 3: Assignment of input x.

- How to get the values of the above matrices?

Q4. Fig. 2 shows a 4-3-2 neural network, which is used to classify the patterns shown in Fig. 1. The assignment of input $x = [x_1 \ x_2 \ x_3 \ x_4]^T$ is shown in Fig. 3 where 0 represents an unfilled grid and 1 represents a filled grid. The activation functions in the input layer, hidden layer and output layer are linear function, symmetric tangent sigmoid function and logarithmic sigmoid function, respectively. Given that the weight and bias matrices as:

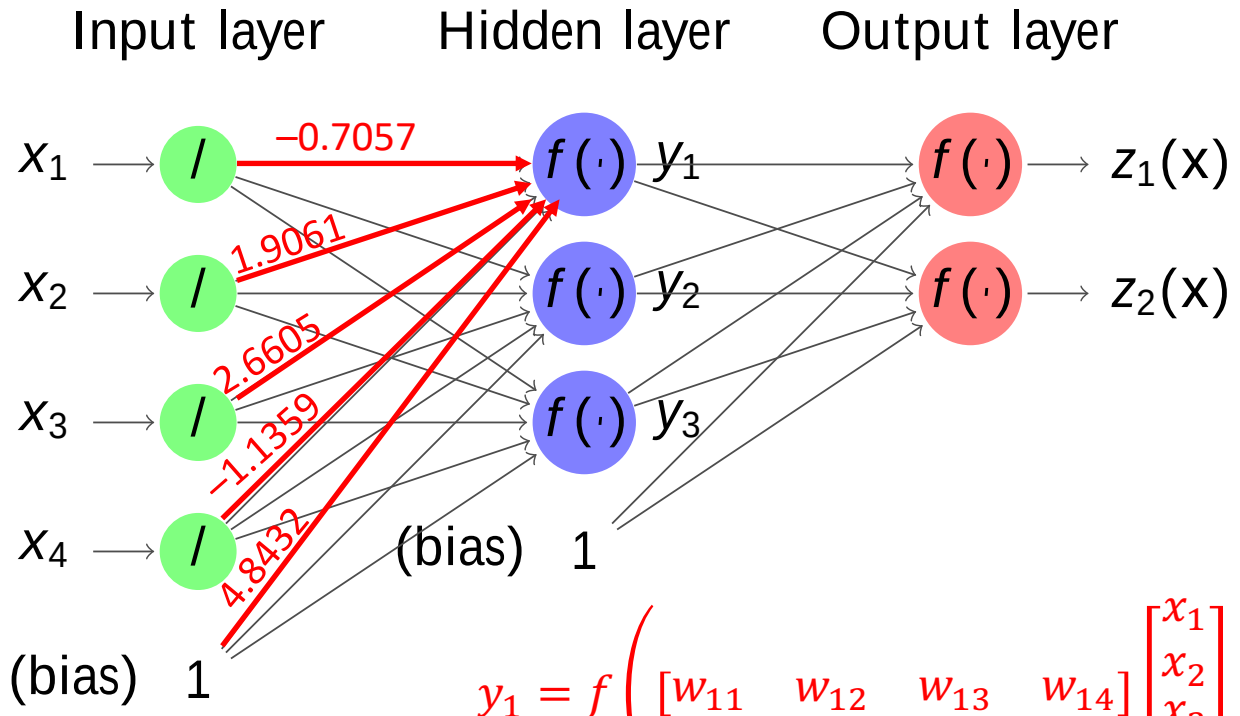
x_1	x_2
x_4	x_3

Figure 3: Assignment of input x.

$$W_{ji} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ -0.7057 & 1.9061 & 2.6605 & -1.1359 \\ 0.4900 & 1.9324 & -0.4269 & -5.1570 \\ 0.9438 & -5.4160 & -0.3431 & -0.2931 \end{bmatrix}, W_{j0} = \begin{bmatrix} w_{10} \\ 4.8432 \\ 0.3973 \\ 2.1761 \end{bmatrix},$$

$$W_{kj} = \begin{bmatrix} -1.1444 & 0.3115 & -9.9812 \\ 0.0106 & 11.5477 & 2.6479 \end{bmatrix}, W_{k0} = \begin{bmatrix} 2.5230 \\ 2.6463 \end{bmatrix},$$

determine the output patterns representing the 3 classes.



$$y_1 = f\left([w_{11} \quad w_{12} \quad w_{13} \quad w_{14}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + w_{10} \right)$$

Q4. Fig. 2 shows a 4-3-2 neural network, which is used to classify the patterns shown in Fig. 1. The assignment of input $x = [x_1 \ x_2 \ x_3 \ x_4]^T$ is shown in Fig. 3 where 0 represents an unfilled grid and 1 represents a filled grid. The activation functions in the input layer, hidden layer and output layer are linear function, symmetric tangent sigmoid function and logarithmic sigmoid function, respectively. Given that the weight and bias matrices as:

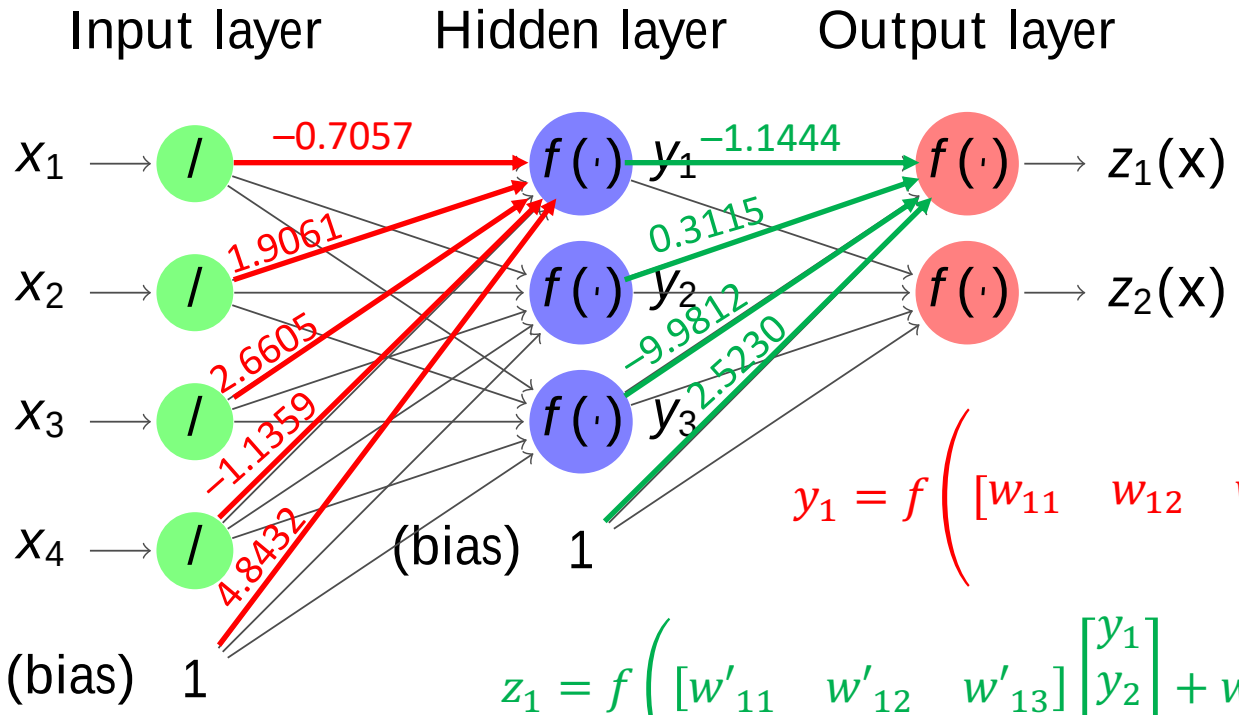
x_1	x_2
x_4	x_3

Figure 3: Assignment of input x.

$$W_{ji} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ -0.7057 & 1.9061 & 2.6605 & -1.1359 \\ 0.4900 & 1.9324 & -0.4269 & -5.1570 \\ 0.9438 & -5.4160 & -0.3431 & -0.2931 \end{bmatrix}, W_{j0} = \begin{bmatrix} w_{10} \\ 4.8432 \\ 0.3973 \\ 2.1761 \end{bmatrix},$$

$$W_{kj} = \begin{bmatrix} w'_{11} & w'_{12} & w'_{13} \\ -1.1444 & 0.3115 & -9.9812 \\ 0.0106 & 11.5477 & 2.6479 \end{bmatrix}, W_{k0} = \begin{bmatrix} w'_{10} \\ 2.5230 \\ 2.6463 \end{bmatrix},$$

determine the output patterns representing the 3 classes.



$$y_1 = f\left([w_{11} \quad w_{12} \quad w_{13} \quad w_{14}]\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + w_{10}\right)$$

$$z_1 = f\left([w'_{11} \quad w'_{12} \quad w'_{13}]\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} + w'_{10}\right)$$

Q4. Fig. 2 shows a 4-3-2 neural network, which is used to classify the patterns shown in Fig. 1. The assignment of input $\mathbf{x} = [\, x_1 \;\; x_2 \;\; x_3 \;\; x_4 \,]^T$ is shown in Fig. 3 where 0 represents an unfilled grid and 1 represents a filled grid. The activation functions in the input layer, hidden layer and output layer are linear function, symmetric tangent sigmoid function and logarithmic sigmoid function, respectively. Given that the weight and bias matrices as:

$$\mathbf{W}_{ji} = \begin{bmatrix} -0.7057 & 1.9061 & 2.6605 & -1.1359 \\ 0.4900 & 1.9324 & -0.4269 & -5.1570 \\ 0.9438 & -5.4160 & -0.3431 & -0.2931 \end{bmatrix}, \mathbf{W}_{j0} = \begin{bmatrix} 4.8432 \\ 0.3973 \\ 2.1761 \end{bmatrix},$$

$$\mathbf{W}_{kj} = \begin{bmatrix} -1.1444 & 0.3115 & -9.9812 \\ 0.0106 & 11.5477 & 2.6479 \end{bmatrix}, \mathbf{W}_{k0} = \begin{bmatrix} 2.5230 \\ 2.6463 \end{bmatrix},$$

determine the output patterns representing the 3 classes.

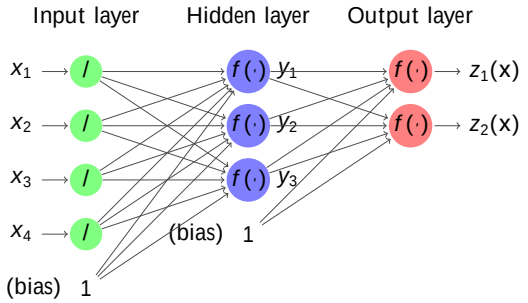


Figure 2: A diagram of 4-3-2 feedforward fully-connected neural network.

Figure 3: Assignment of input $\mathbf{x}$.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = f(\mathbf{W}_{ji}\mathbf{x} + \mathbf{W}_{j0}),\ \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = f(\mathbf{W}_{kj}\mathbf{y} + \mathbf{W}_{k0})$$

Sigmoid activation

For pattern 1, $\mathbf{x} = [\, 1 \;\; 0 \;\; 1 \;\; 0 \,]^T$, we have $\mathbf{y} = \begin{bmatrix} 1.0000 \\ 0.4304 \\ 0.9923 \end{bmatrix}$, $\mathbf{z} = \begin{bmatrix} 0.0002 \\ 1.0000 \end{bmatrix}$.

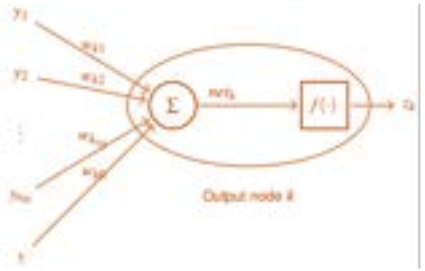
For pattern 2, $\mathbf{x} = [\, 0 \;\; 1 \;\; 0 \;\; 1 \,]^T$, we have $\mathbf{y} = \begin{bmatrix} 1.0000 \\ -0.9930 \\ -0.9983 \end{bmatrix}$, $\mathbf{z} = \begin{bmatrix} 1.0000 \\ 0.0000 \end{bmatrix}$.

For pattern 3, $\mathbf{x} = [\, 1 \;\; 1 \;\; 0 \;\; 0 \,]^T$, we have $\mathbf{y} = \begin{bmatrix} 1.0000 \\ 0.9929 \\ -0.9799 \end{bmatrix}$, $\mathbf{z} = \begin{bmatrix} 1.0000 \\ 1.0000 \end{bmatrix}$.

So, pattern 1 is represented by $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$, pattern 2 is represented by $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and pattern 3 is represented by $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

- Are there any other options to represent the output classes?
- After knowing the class label, if an input sample gives z = [0.8; 0.6], which class should it be classified to?

Q5. Fig. 4 shows a 3-layer partially connected neural network. Linear function is used as the activation function in the input units, symmetric tangent sigmoid function is used as the activation function in all hidden and output units. Stochastic backpropagation is employed to train the neural network using the cost $J = \frac{1}{2}\,\mathrm{t} - z_1\,^2$ where z_1 is the network output corresponding to input pattern x selected from the training set and t is its corresponding target output.

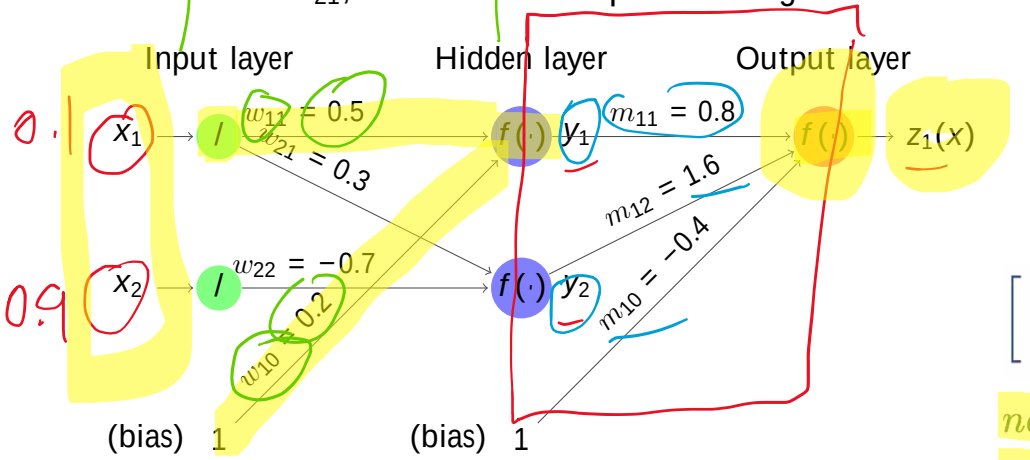


a. Considering $\mathrm{x} = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$, determine y_1, y_2 and z_1.

b. Derive the backpropagation update rules for m_{10} and w_{21}.

c. Given that the learning rate $\eta = 0.25$, $\mathrm{x} = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$, $\mathrm{t} = 0.5$ and the weights in Fig. 4, determine the updated values of m_{10} and w_{21} for the next iteration based on the update rules obtained in Q5.*b*.

d. Assuming that the backpropagation algorithm only updates the weights m_{10} and w_{21}, show that the updated weights are valid.



$$y_1 = f(0.5 \times x_1 + 0 \times x_2 + 1 \times 0.2)$$
$$y_2 = f(0.3 \times x_1 - 0.7 \times x_2)$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = f\left(\begin{bmatrix} 0.5 & 0 \\ 0.3 & -0.7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} \right)$$

$$net = 0.8y_1 + 1.6y_2 - 0.4$$
$$z_1 = f(net).$$

Figure 4: A diagram of 3-layer feedforward neural network.

Q5. Fig. 4 shows a 3-layer partially connected neural network. Linear function is used as the activation function in the input units, symmetric tangent sigmoid function is used as the activation function in all hidden and output units. Stochastic backpropagation is employed to train the neural network using the cost $J = \frac{1}{2} t - z_1^{\ 2}$ where z_1 is the network output corresponding to input pattern x selected from the training set and t is its corresponding target output.

a. Considering $x = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$, determine y_1, y_2 and z_1.

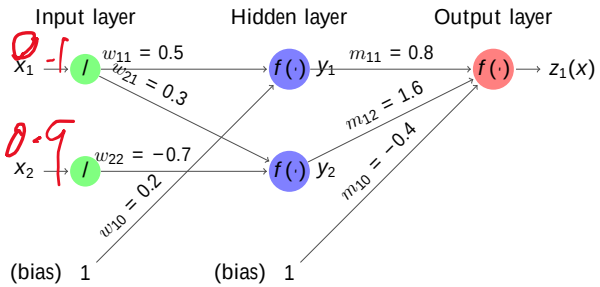


Figure 4: A diagram of 3-layer feedforward neural network.

Q5. Fig. 4 shows a 3-layer partially connected neural network. Linear function is used as the activation function in the input units, symmetric tangent sigmoid function is used as the activation function in all hidden and output units. Stochastic backpropagation is employed to train the neural network using the cost $J = \frac{1}{2} t - z_1^{\ 2}$ where z_1 is the network output corresponding to input pattern x selected from the training set and t is its corresponding target output.

b. Derive the backpropagation update rules for m_{10} and w_{21}.

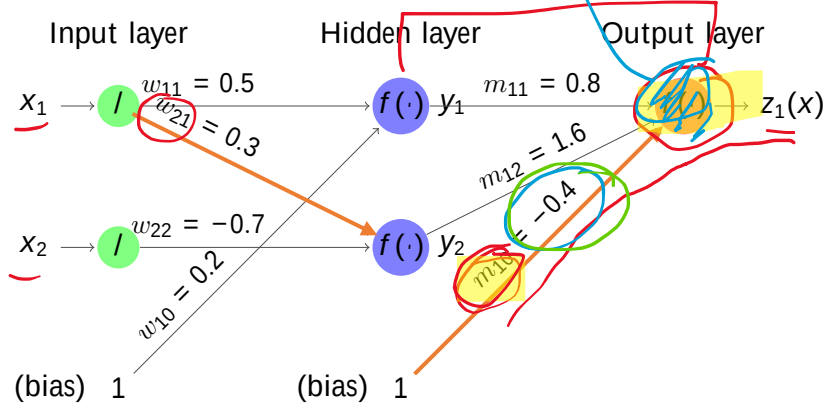


Figure 4: A diagram of 3-layer feedforward neural network.

Q5. Fig. 4 shows a 3-layer partially connected neural network. Linear function is used as the activation function in the input units, symmetric tangent sigmoid function is used as the activation function in all hidden and output units. Stochastic backpropagation is employed to train the neural network using the cost $J = \frac{1}{2}(t - z_1)^2$ where z_1 is the network output corresponding to input pattern x selected from the training set and t is its corresponding target output.

b. Derive the backpropagation update rules for m_{10} and w_{21}.

From the network, we have

- $y_2 = f(a)$
- $a = w_{21}x_1 + w_{22}x_2 = 0.3x_1 - 0.7x_2$
- When $\mathbf{x} = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$, $a = 0.3x_1 - 0.7x_2 = -0.6$ and $f(a) = -0.5370$.

Considering the weight w_{21}, we have

$$\Delta w_{21} = -\eta \frac{\partial J}{\partial w_{21}}$$

$$= -\eta \frac{\partial J}{\partial f(net)} \frac{\partial f(net)}{\partial net} \frac{\partial net}{\partial y_2} \frac{\partial y_2}{\partial a} \frac{\partial a}{\partial w_{21}}$$

$$= -\eta(z_1 - \mathbf{t}) \frac{4e^{-2net}}{(1+e^{-2net})^2} m_{12} \frac{4e^{-2a}}{(1+e^{-2a})^2} x_1$$

Update rule:

$$w_{21} \leftarrow w_{21} + \Delta w_{21} \Leftarrow \Delta w_{21} \leftarrow w_{21} - \eta(z_1 - \mathbf{t}) \frac{4e^{-2net}}{(1+e^{-2net})^2} m_{12} \frac{4e^{-2a}}{(1+e^{-2a})^2} x_1.$$

When $\mathbf{x} = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$, we have

$$w_{21} \leftarrow 0.3 - 0.25 \times (-0.7869 - 0.5) \frac{4e^{-2\times -1.0633}}{(1+e^{-2\times -1.0633})^2} \times 1.6 \times \frac{4e^{-2\times -0.6}}{(1+e^{-2\times -0.6})^2} \times 0.1 = 0.3140.$$

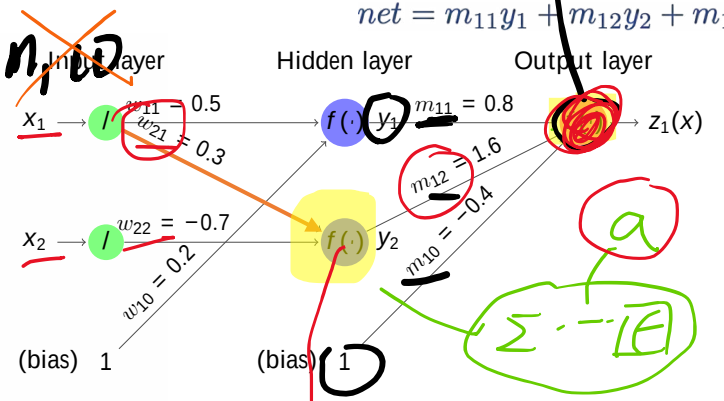


Figure 4: A diagram of 3-layer feedforward neural network.

Q5. Fig. 4 shows a 3-layer partially connected neural network. Linear function is used as the activation function in the input units, symmetric tangent sigmoid function is used as the activation function in all hidden and output units. Stochastic backpropagation is employed to train the neural network using the cost $J = \frac{1}{2} \mathrm{t} - \mathrm{z}_1 \ ^2$ where z_1 is the network output corresponding to input pattern x selected from the training set and t is its corresponding target output.

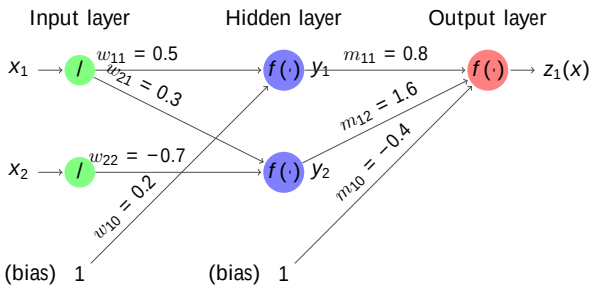


Figure 4: A diagram of 3-layer feedforward neural network.

d. Assuming that the backpropagation algorithm only updates the weights m_{10} and w_{21}, show that the updated weights are valid.

With the updated m_{10} and w_{21},

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = f\left(\begin{bmatrix} 0.5 & 0 \\ 0.3140 & -0.7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} \right).$$

When $\mathbf{x} = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$,

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0.2449 \\ -0.5361 \end{bmatrix}.$$

The network output z_1:

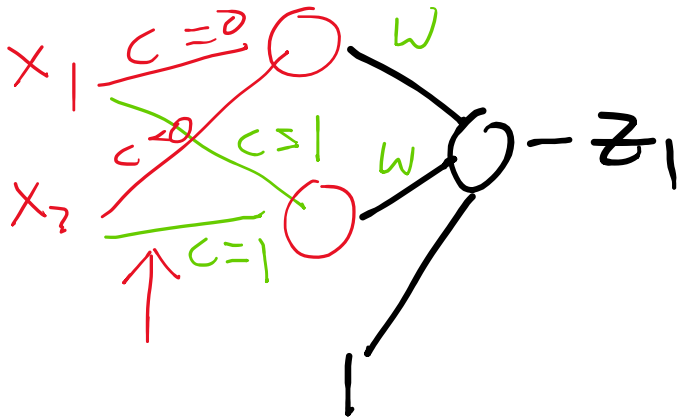
$net = 0.8y_1 + 1.6y_2 - 0.2775 = -0.9393,$

$z_1 = f(net) = f(-0.9393) = -0.7349.$

The value of J is reduced from $J = \frac{1}{2}(-0.7869 - 0.5)^2 = 0.8281$ (before the update) to $J = \frac{1}{2}(-0.7349 - 0.5)^2 = 0.7625$ (after the update).

The update of the weights brings the output z_1 closer to the target value.

Q6. Consider an XOR problem given in Table 1. Design a radial basis function (RBF) neural network with bias to solve the problem. Choose input pattern 1 and pattern 4 as the centres; Gaussian function in the hidden units (with $\sigma_j = \frac{\rho_{\max}}{\sqrt{2n_H}}$) and linear function in the output units.



a. Draw a diagram showing the structure of an RBF neural network solving the XOR problem.

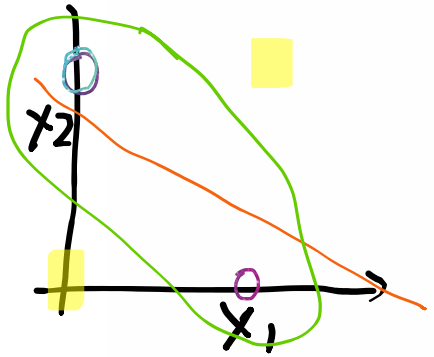
b. Compute the outputs at the hidden units for all input patterns and list them in a table. Show that the patterns at the hidden units are linearly separable.

c. Determine the output weights using the least squares method. Compute the outputs at the output unit(s) and list them in a table. Show that the XOR problem is solved.

d. Determine the classes for the input patterns given in Table 2.

p	x_1	x_2	t
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Table 1: XOR problem.



p	x_1	x_2
1	0.5	−0.1
2	−0.2	1.2
3	0.8	0.3
4	1.8	0.6

Table 2: Input patterns for the XOR classifier.

Q6. Consider an XOR problem given in Table 1. Design a radial basis function (RBF) neural network with bias to solve the problem. Choose input pattern 1 and pattern 4 as the centres; Gaussian function in the hidden units (with $\sigma_j = \frac{\rho_{\max}}{\sqrt{2n_H}}$) and linear function in the output units.

Rho max

a. Draw a diagram showing the structure of an RBF neural network solving the XOR problem.

p	x_1	x_2	t
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Table 1: XOR problem.

$$h(a) = e^{\frac{-\|a-c\|}{2\sigma^2}}$$

A 2-2-1 RBF is employed for the XOR problem.

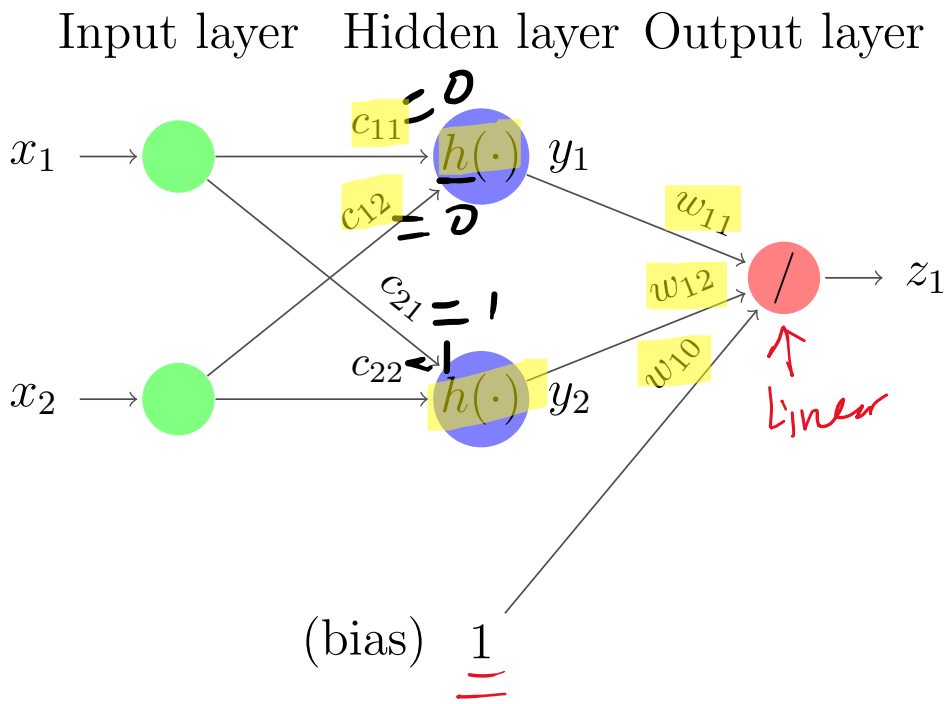
p	x_1	x_2
1	0.5	−0.1
2	−0.2	1.2
3	0.8	0.3
4	1.8	0.6

Table 2: Input patterns for the XOR classifier.

$$\sigma = \frac{\rho_{max}}{\sqrt{2 \times 2}}$$

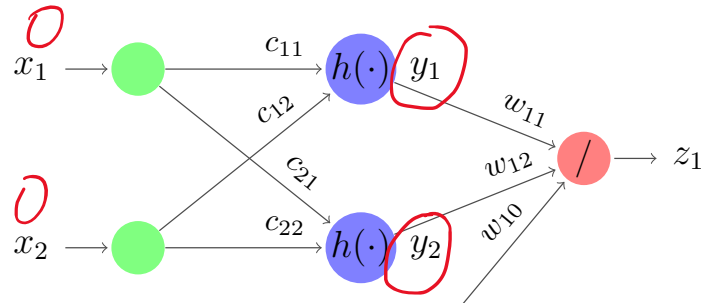
n_H

$$\rho_{max} = \|[1\ 1] - [0\ 0]\|$$



$$z_1 = y_1 w_{11} + y_2 w_{12} + w_{10}$$

Q6. Consider an XOR problem given in Table 1. Design a radial basis function (RBF) neural network with bias to solve the problem. Choose input pattern 1 and pattern 4 as the centres; Gaussian function in the hidden units (with $\sigma_j = \frac{\rho_{\max}}{\sqrt{2n_H}}$) and linear function in the output units.



b. Compute the outputs at the hidden units for all input patterns and list them in a table. Show that the patterns at the hidden units are linearly separable.

Sample 1

$\rho_{\max} = \|\mathbf{c}_1 - \mathbf{c}_2\|$, $n_H = 2$

Sample 4

$\mathbf{c}_1 = [c_{11} \quad c_{12}] = [0 \quad 0]$, $\mathbf{c}_2 = [c_{21} \quad c_{22}] = [1 \quad 1]$

$\sigma_1 = \sigma_2 = \frac{\rho_{\max}}{\sqrt{2n_H}} = \frac{\sqrt{(0-1)^2+(0-1)^2}}{\sqrt{2\times 2}} = \frac{1}{\sqrt{2}}$.

$y_1(\mathbf{x}_p) = e^{-\frac{\|\mathbf{x}_p - \mathbf{c}_1\|^2}{2\sigma_1^2}}$, $y_2(\mathbf{x}_p) = e^{-\frac{\|\mathbf{x}_p - \mathbf{c}_2\|^2}{2\sigma_2^2}}$, $p = 1, 2, 3, 4$.

Gaussian Function

p	x_1	x_2	t
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Table 1: XOR problem.

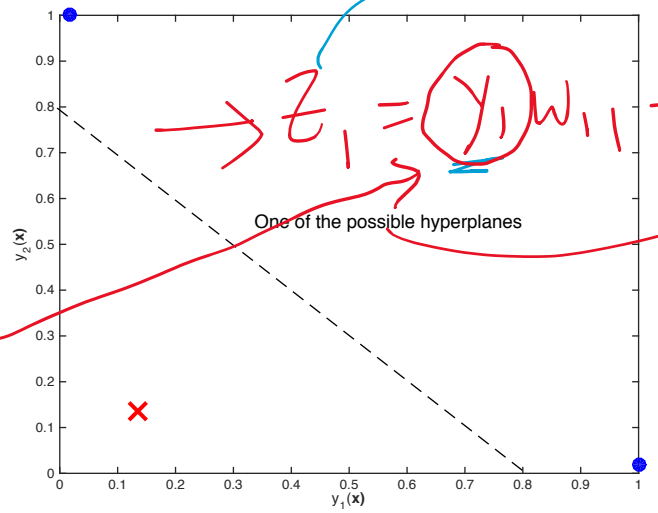
$x_1 = 0, x_2 = 0$ (bias) 1

$0 = 1 w_{11} + w_{12} + w_{10}$

0.1353

$1 = 0.3679 w_{11} + 0.3679 w_{12} + w_{10}$

p	x_1	x_2	$y_1(\mathbf{x})$	$y_2(\mathbf{x})$
1	0	0	1.0000	0.1353
2	0	1	0.3679	0.3679
3	1	0	0.3679	0.3679
4	1	1	0.1353	1.0000



$z_1 = y_1 w_{11} + y_2 w_{12} + w_{10}$

Q6. Consider an XOR problem given in Table 1. Design a radial basis function (RBF) neural network with bias to solve the problem. Choose input pattern 1 and pattern 4 as the centres; Gaussian function in the hidden units (with $\sigma_j = \frac{\rho_{\max}}{\sqrt{2n_H}}$) and linear function in the output units.

c. Determine the output weights using the least squares method. Compute the outputs at the output unit(s) and list them in a table. Show that the XOR problem is solved.

p	x_1	x_2	t
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Table 1: XOR problem.

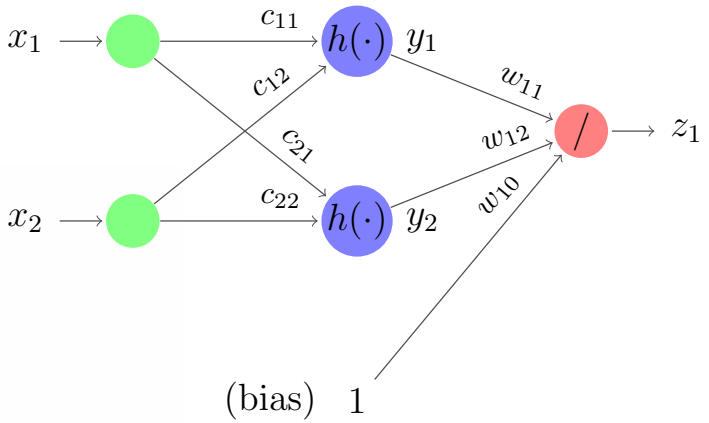
p	x_1	x_2
1	0.5	−0.1
2	−0.2	1.2
3	0.8	0.3
4	1.8	0.6

Table 2: Input patterns for the XOR classifier.

$\phi(x) w_1 = t_1$

$\phi^T \phi w_1 = \phi^T t_1$

$w_1 = (\phi^T \phi)^{-1} \phi^T t$



Activation function in the output units: *linear function*

Output equation of the RBF neural network is:

$$z_1(\mathbf{x}) = \sum_{j=0}^{2} w_{1j} y_j(\mathbf{x}) = w_{11} y_1(\mathbf{x}) + w_{12} y_2(\mathbf{x}) + w_{10}.$$

There are 3 weights to be determined, i.e., w_{11}, w_{12} and w_{10} (bias).

$y_1(\mathbf{x})$ $y_2(\mathbf{x})$ bias

$$\begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.1353 & 1 & 1 \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{12} \\ w_{10} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$\Phi(\mathbf{x})$ $\mathbf{w}_1$ $\mathbf{t}_1$ $z_1(\mathbf{x})$

p	x_1	x_2	$y_1(\mathbf{x})$	$y_2(\mathbf{x})$
1	0	0	1.0000	0.1353
2	0	1	0.3679	0.3679
3	1	0	0.3679	0.3679
4	1	1	0.1353	1.0000

Q6. Consider an XOR problem given in Table 1. Design a radial basis function (RBF) neural network with bias to solve the problem. Choose input pattern 1 and pattern 4 as the centres; Gaussian function in the hidden units (with $\sigma_j = \frac{\rho_{\max}}{\sqrt{2n_H}}$) and linear function in the output units.

p	x_1	x_2	t
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Table 1: XOR problem.

c. Determine the output weights using the least squares method. Compute the outputs at the output unit(s) and list them in a table. Show that the XOR problem is solved.

$$\underbrace{\begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.3679 & 0.3679 & 1 \\ 0.1353 & 1 & 1 \end{bmatrix}}_{\Phi(\mathbf{x})} \underbrace{\begin{bmatrix} w_{11} \\ w_{12} \\ w_{10} \end{bmatrix}}_{\mathbf{w}_1} = \underbrace{\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}}_{\mathbf{t}_1}$$

$$\mathbf{W}_1 = \left(\Phi(\mathbf{x})^T \Phi(\mathbf{x})\right)^{-1} \Phi(\mathbf{x})^T \mathbf{t}_1$$

Applying least squares method: $w_{11} = w_{12} = -2.5027$ and $w_{01} = 2.8413$.

Output equation of RBF neural network:

$$z_1(\mathbf{x}) = -2.5027 y_1(\mathbf{x}) - 2.5027 y_2(\mathbf{x}) + 2.8413$$
$$= -2.5027 e^{\frac{-\|\mathbf{x}-\mathbf{c}_1\|^2}{2\sigma_1^2}} - 2.5027 e^{\frac{-\|\mathbf{x}-\mathbf{c}_2\|^2}{2\sigma_2^2}} + 2.8413$$

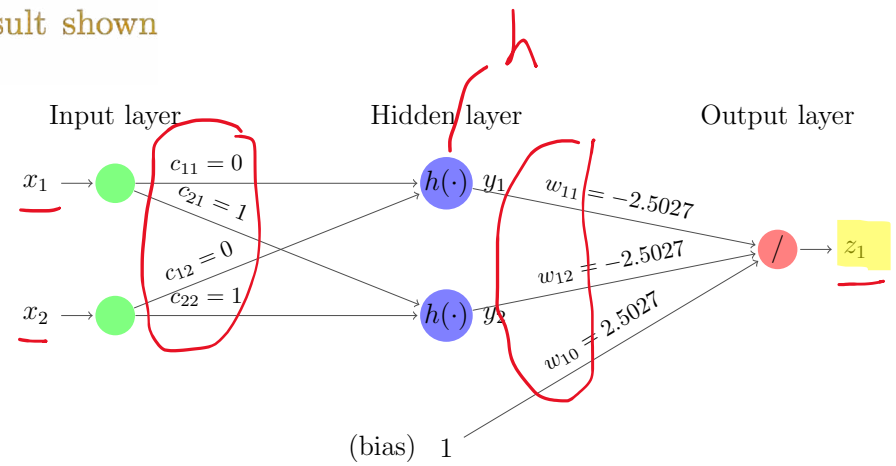
p	x_1	x_2
1	0.5	−0.1
2	−0.2	1.2
3	0.8	0.3
4	1.8	0.6

Table 2: Input patterns for the XOR classifier.

Applying the input patterns $\mathbf{x}$ to the equation, we have the result shown below, which shows that the classification can be done successfully.

$$\frac{\text{sgn}(z_1 - 0.5) + 1}{2} = \begin{cases} 1 & \text{if } z_1 > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

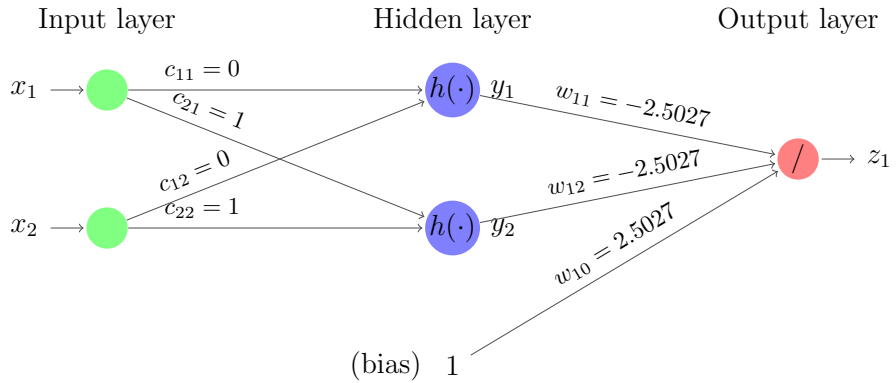
p	x_1	x_2	$y_1(\mathbf{x})$	$y_2(\mathbf{x})$	$z_1(\mathbf{x})$	Class: $\frac{\text{sgn}(z_1(\mathbf{x}) - 0.5) + 1}{2}$
1	0	0	1.0000	0.1353	−0.0001	0
2	0	1	0.3679	0.3679	0.9999	1
3	1	0	0.3679	0.3679	0.9999	1
4	1	1	0.1353	1.0000	−0.0001	0



$$\frac{\text{sgn}(z_1 - 0.5) + 1}{2}$$

Q6. Consider an XOR problem given in Table 1. Design a radial basis function (RBF) neural network with bias to solve the problem. Choose input pattern 1 and pattern 4 as the centres; Gaussian function in the hidden units (with $\sigma_j = \frac{\rho_{\max}}{\sqrt{2n_H}}$) and linear function in the output units.

d. Determine the classes for the input patterns given in Table 2.



p	x_1	x_2
1	0.5	−0.1
2	−0.2	1.2
3	0.8	0.3
4	1.8	0.6

Table 2: Input patterns for the XOR classifier.

p	x_1	x_2	$y_1(\mathbf{x})$	$y_2(\mathbf{x})$	$z_1(\mathbf{x})$	Class: $\frac{\text{sgn}(z_1(\mathbf{x})-0.5)+1}{2}$
1	0.5	−0.1	0.7711	0.2322	0.3304	0
2	−0.2	1.2	0.2276	0.2276	1.7019	1
3	0.8	0.3	0.4819	0.5886	0.1621	0
4	1.8	0.6	0.0273	0.4493	1.6484	1

Q6. Consider an XOR problem given in Table 1. Design a radial basis function (RBF) neural network with bias to solve the problem. Choose input pattern 1 and pattern 4 as the centres; Gaussian function in the hidden units (with $\sigma_j = \frac{\rho_{\max}}{\sqrt{2n_H}}$) and linear function in the output units.

Training phase (design phase):

Training samples

p	x_1	x_2	t
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Table 1: XOR problem.

→ **Determine:**
- **c**
- σ
- Activation functions

→ **Computing y(x) using the training samples**

→ **Determining the weights at the output layer using y(x) by least squares method**

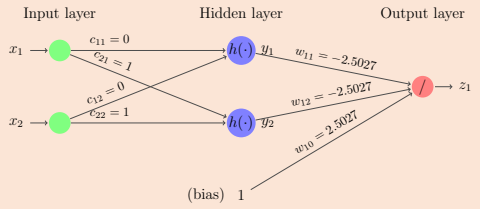
Test phase (classification phase):

Test samples (or any input sample)

p	x_1	x_2
1	0.5	−0.1
2	−0.2	1.2
3	0.8	0.3
4	1.8	0.6

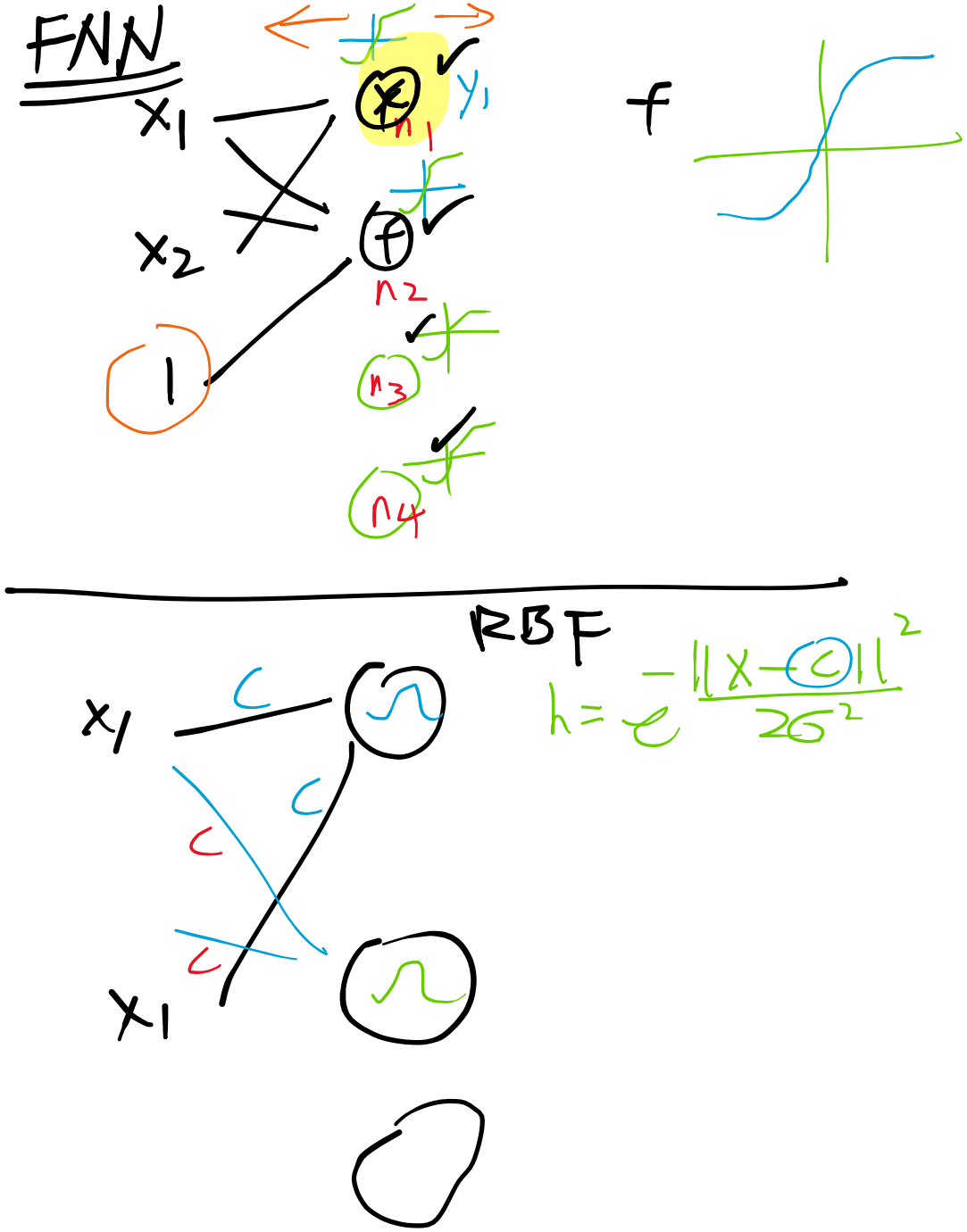
Table 2: Input patterns for the XOR classifier.

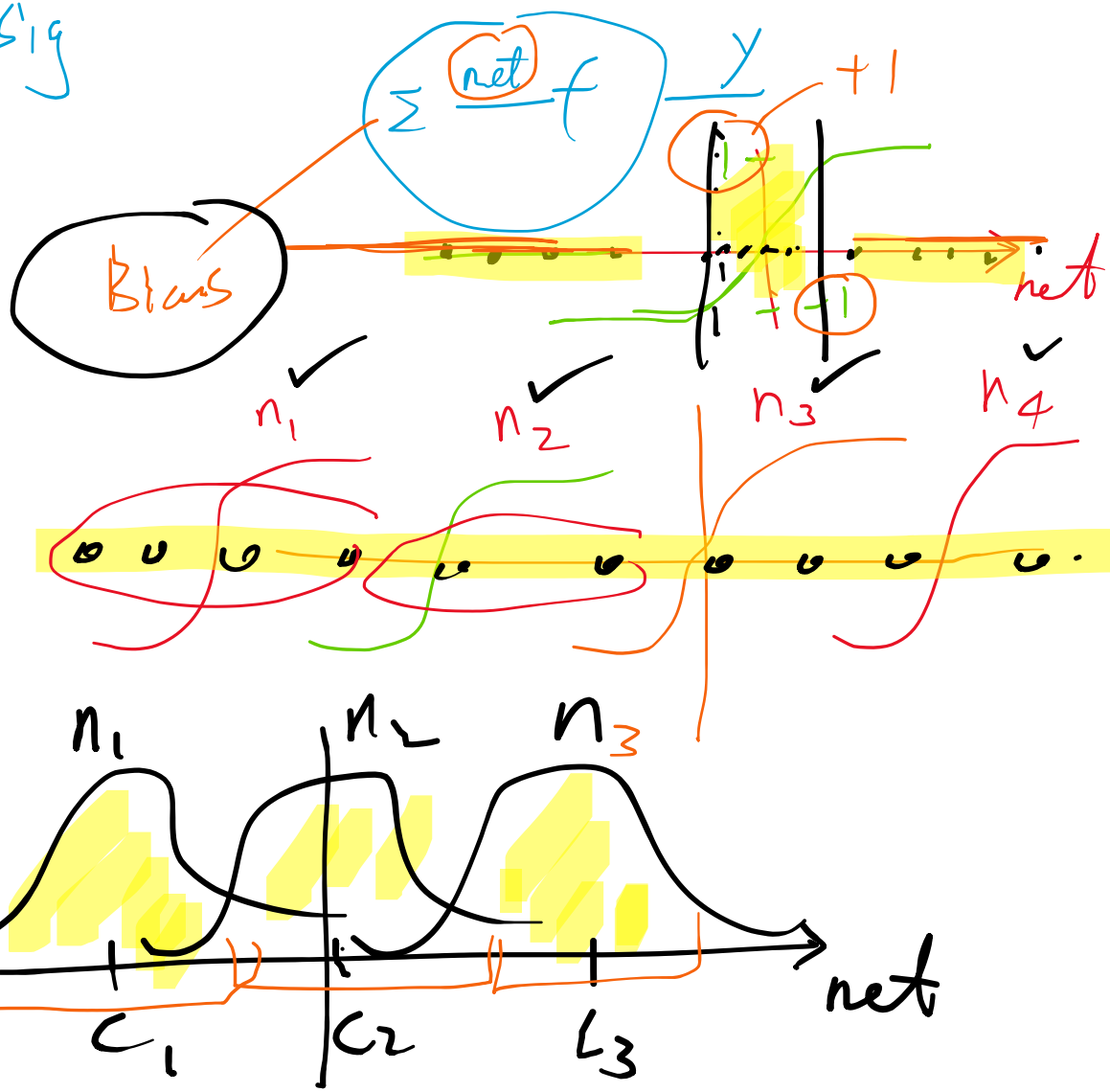
→ **Feeding samples to the designed RBF NN**



→ **Classifying output z according to some rules**

$$\frac{\text{sgn}(z_1 - 0.5) + 1}{2} = \begin{cases} 1 & \text{if } z_1 > 0.5 \\ 0 & \text{otherwise} \end{cases}$$


FNN
x1
x2
1
n1
n2
n3
n4
y1
f
tansig
RBF
h = e^(-||X-C||^2 / 2σ^2)
C
x1
x1


Σ
net
f
y
+1
Bias
net
n1
n2
n3
n4

n1
n2
n3
net
c1
c2
c3

Q7. Consider a classifier which is able to separate two classes, i.e.,

$$\begin{array}{ll} \text{class 1} & g(x) \geq \frac{x}{2} \\ \text{class 2} & g(x) < \frac{x}{2} \end{array}$$

where $g(x)$ is an unknown function. We have a limited information about the function $g(x)$. Table 3 lists the input-output mappings of the function $g(x)$.

a. Sketch the function $g(x)$ and determine the possible class the following points x = 0.1, 0.35, 0.55, 0.75, 0.9 belong to.

b. An RBF neural network is employed to implement the classifier. Determine its number of inputs and outputs.

c. An RBF neural network with three hidden units is employed to implement the classifier. Gaussian function with σ = 0.1 is used in all hidden units and linear function is used in all output units. Given that c_1 = 0.2, c_2 = 0.6 c_3 = 0.9, determine the output weights of the RBF neural network using the least squares method. Give the equation of the designed RBF neural networks.

d. Given clusters $S_1 = \{x_1, x_2, x_3\}$, $S_2 = \{x_4, x_5\}$, $S_3 = \{x_6, x_7, x_8, x_9\}$ and $S_4 = \{x_{10}, x_{11}, x_{12}\}$, find the centres for an RBF neural network implementing the classifier. Gaussian function with $\sigma = 2\rho_{avg}$ is used in all hidden units and linear function is used in all output units. Based on the found centres, determine the output weights of the RBF neural network using the least squares method. Draw the diagram of the designed RBF neural network and show the weights.

e. Which of the 2 RBF neural networks perform better? Justify your answer.

p	x	z
1	0.0500	0.0863
2	0.2000	0.2662
3	0.2500	0.2362
4	0.3000	0.1687
5	0.4000	0.1260
6	0.4300	0.1756
7	0.4800	0.3290
8	0.6000	0.6694
9	0.7000	0.4573
10	0.8000	0.3320
11	0.9000	0.4063
12	0.9500	0.3535

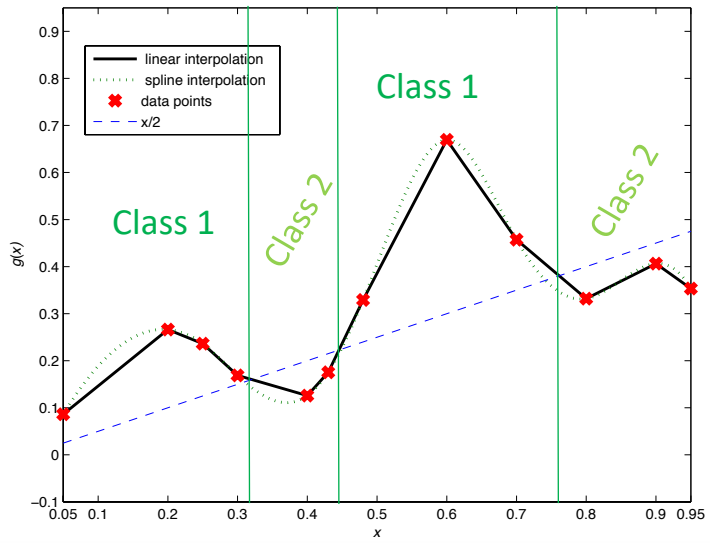
Table 3: Input-output mappings for function $g(x)$.

Q7. Consider a classifier which is able to separate two classes, i.e.,

$$\begin{array}{ll} \text{class 1} & g(x) \geq \frac{x}{2} \\ \text{class 2} & g(x) < \frac{x}{2} \end{array}$$

where $g(x)$ is an unknown function. We have a limited information about the function $g(x)$. Table 3 lists the input-output mappings of the function $g(x)$.

a. Sketch the function $g(x)$ and determine the possible class the following points x = 0.1, 0.35, 0.55, 0.75, 0.9 belong to.



p	x	z
1	0.0500	0.0863
2	0.2000	0.2662
3	0.2500	0.2362
4	0.3000	0.1687
5	0.4000	0.1260
6	0.4300	0.1756
7	0.4800	0.3290
8	0.6000	0.6694
9	0.7000	0.4573
10	0.8000	0.3320
11	0.9000	0.4063
12	0.9500	0.3535

Table 3: Input-output mappings for function $g(x)$.

x	Class
0.1	1
0.35	2
0.55	1
0.75	1 (hard to tell)
0.9	2

Class for input patterns based on the RBF classifier.

Q7. Consider a classifier which is able to separate two classes, i.e.,

$$\begin{array}{ll} \text{class 1} & g(x) \geq \frac{x}{2} \\ \text{class 2} & g(x) < \frac{x}{2} \end{array}$$

where $g(x)$ is an unknown function. We have a limited information about the function $g(x)$. Table 3 lists the input-output mappings of the function $g(x)$.

b. An RBF neural network is employed to implement the classifier. Determine its number of inputs and outputs.

p	x	z
1	0.0500	0.0863
2	0.2000	0.2662
3	0.2500	0.2362
4	0.3000	0.1687
5	0.4000	0.1260
6	0.4300	0.1756
7	0.4800	0.3290
8	0.6000	0.6694
9	0.7000	0.4573
10	0.8000	0.3320
11	0.9000	0.4063
12	0.9500	0.3535

Table 3: Input-output mappings for function $g(x)$.

A single-input-single-output RBF neural network which takes x as an input and z as output for this classification problem.

x as input
z as output

Q7. Consider a classifier which is able to separate two classes, i.e.,

$$\begin{array}{ll} \text{class 1} & g(x) \geq \frac{x}{2} \\ \text{class 2} & g(x) < \frac{x}{2} \end{array}$$

where $g(x)$ is an unknown function. We have a limited information about the function $g(x)$. Table 3 lists the input-output mappings of the function $g(x)$.

c. An RBF neural network with three hidden units is employed to implement the classifier. Gaussian function with σ = 0.1 is used in all hidden units and linear function is used in all output units. Given that c_1 = 0.2, c_2 = 0.6 c_3 = 0.9, determine the output weights of the RBF neural network using the least squares method. Give the equation of the designed RBF neural networks.

$$z_1(x) = w_{11}y_1(x) + w_{12}y_2(x) + w_{13}y_3(x) + w_{10}$$

where

$$y_1(x) = e^{-\frac{(x-c_1)^2}{2\times 0.1^2}},$$

$$y_2(x) = e^{-\frac{(x-c_2)^2}{2\times 0.1^2}},$$

$$y_3(x) = e^{-\frac{(x-c_3)^2}{2\times 0.1^2}}.$$

p	x	$y_1(x)$	$y_2(x)$	$y_3(x)$	Target: z
1	0.0500	0.3247	0.0000	0.0000	0.0863
2	0.2000	1.0000	0.0003	0.0000	0.2662
3	0.2500	0.8825	0.0022	0.0000	0.2362
4	0.3000	0.6065	0.0111	0.0000	0.1687
5	0.4000	0.1353	0.1353	0.0000	0.1260
6	0.4300	0.0710	0.2357	0.0000	0.1756
7	0.4800	0.0198	0.4868	0.0001	0.3290
8	0.6000	0.0003	1.0000	0.0111	0.6694
9	0.7000	0.0000	0.6065	0.1353	0.4573
10	0.8000	0.0000	0.1353	0.6065	0.3320
11	0.9000	0.0000	0.0111	1.0000	0.4063
12	0.9500	0.0000	0.0022	0.8825	0.3535

p	x	z
1	0.0500	0.0863
2	0.2000	0.2662
3	0.2500	0.2362
4	0.3000	0.1687
5	0.4000	0.1260
6	0.4300	0.1756
7	0.4800	0.3290
8	0.6000	0.6694
9	0.7000	0.4573
10	0.8000	0.3320
11	0.9000	0.4063
12	0.9500	0.3535

Table 3: Input-output mappings for function $g(x)$.

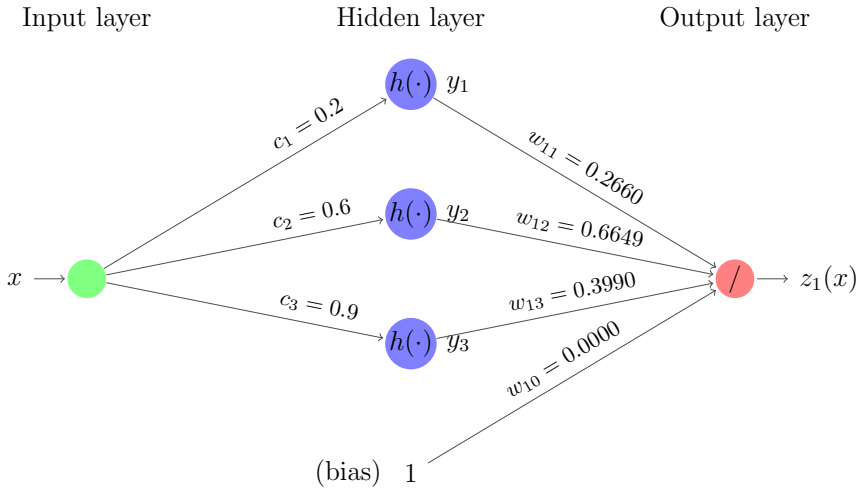


Figure 6: A diagram of 1-3-1 RBF neural network.

Q7. Consider a classifier which is able to separate two classes, i.e.,

class 1 $g(x) \geq \frac{x}{2}$

class 2 $g(x) < \frac{x}{2}$

where $g(x)$ is an unknown function. We have a limited information about the function $g(x)$. Table 3 lists the input-output mappings of the function $g(x)$.

p	x	z
1	0.0500	0.0863
2	0.2000	0.2662
3	0.2500	0.2362
4	0.3000	0.1687
5	0.4000	0.1260
6	0.4300	0.1756
7	0.4800	0.3290
8	0.6000	0.6694
9	0.7000	0.4573
10	0.8000	0.3320
11	0.9000	0.4063
12	0.9500	0.3535

Table 3: Input-output mappings for function $g(x)$.

c. An RBF neural network with three hidden units is employed to implement the classifier. Gaussian function with $\sigma = 0.1$ is used in all hidden units and linear function is used in all output units. Given that $c_1 = 0.2$, $c_2 = 0.6$ $c_3 = 0.9$, determine the output weights of the RBF neural network using the least squares method. Give the equation of the designed RBF neural networks.

$$z_1(x) = w_{11}y_1(x) + w_{12}y_2(x) + w_{13}y_3(x) + w_{10}$$

$$y_1(x) = e^{-\frac{(x-c_1)^2}{2\times 0.1^2}},$$
$$y_2(x) = e^{-\frac{(x-c_2)^2}{2\times 0.1^2}},$$
$$y_3(x) = e^{-\frac{(x-c_3)^2}{2\times 0.1^2}}.$$

p	x	$y_1(x)$	$y_2(x)$	$y_3(x)$	Target: z
1	0.0500	0.3247	0.0000	0.0000	0.0863
2	0.2000	1.0000	0.0003	0.0000	0.2662
3	0.2500	0.8825	0.0022	0.0000	0.2362
4	0.3000	0.6065	0.0111	0.0000	0.1687
5	0.4000	0.1353	0.1353	0.0000	0.1260
6	0.4300	0.0710	0.2357	0.0000	0.1756
7	0.4800	0.0198	0.4868	0.0001	0.3290
8	0.6000	0.0003	1.0000	0.0111	0.6694
9	0.7000	0.0000	0.6065	0.1353	0.4573
10	0.8000	0.0000	0.1353	0.6065	0.3320
11	0.9000	0.0000	0.0111	1.0000	0.4063
12	0.9500	0.0000	0.0022	0.8825	0.3535

$$\underbrace{\begin{bmatrix} 0.3247 & 0.0000 & 0.0000 & 1 \\ 1.0000 & 0.0003 & 0.0000 & 1 \\ 0.8825 & 0.0022 & 0.0000 & 1 \\ 0.6065 & 0.0111 & 0.0000 & 1 \\ 0.1353 & 0.1353 & 0.0000 & 1 \\ 0.0710 & 0.2357 & 0.0000 & 1 \\ 0.0198 & 0.4868 & 0.0001 & 1 \\ 0.0003 & 1.0000 & 0.0111 & 1 \\ 0.0000 & 0.6065 & 0.1353 & 1 \\ 0.0000 & 0.1353 & 0.6065 & 1 \\ 0.0000 & 0.0111 & 1.0000 & 1 \\ 0.0000 & 0.0022 & 0.8825 & 1 \end{bmatrix}}_{\Phi(\mathbf{x})} \underbrace{\begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{10} \end{bmatrix}}_{\mathbf{w}_1} = \underbrace{\begin{bmatrix} 0.0863 \\ 0.2662 \\ 0.2362 \\ 0.1687 \\ 0.1260 \\ 0.1756 \\ 0.3290 \\ 0.6694 \\ 0.4573 \\ 0.3320 \\ 0.4063 \\ 0.3535 \end{bmatrix}}_{\mathbf{t}_1}$$

Applying least squares method:

$w_{11} = 0.2660,\ w_{12} = 0.6649,\ w_{13} = 0.3990,\ w_{10} = 0.0000.$

Output equation of RBF neural networt:

$$z_1(x) = 0.2660y_1(x) + 0.6649y_2(x) + 0.3990y_3(x)$$
$$= 0.2660e^{-\frac{(x-c_1)^2}{2\times 0.1^2}} + 0.6649e^{-\frac{(x-c_2)^2}{2\times 0.1^2}} + 0.3990e^{-\frac{(x-c_3)^2}{2\times 0.1^2}}.$$

Q7. Consider a classifier which is able to separate two classes, i.e.,

$$\begin{array}{ll} \text{class 1} & g(x) \geq \frac{x}{2} \\ \text{class 2} & g(x) < \frac{x}{2} \end{array}$$

where $g(x)$ is an unknown function. We have a limited information about the function $g(x)$. Table 3 lists the input-output mappings of the function $g(x)$.

c. An RBF neural network with three hidden units is employed to implement the classifier. Gaussian function with $\sigma = 0.1$ is used in all hidden units and linear function is used in all output units. Given that $c_1 = 0.2$, $c_2 = 0.6$ $c_3 = 0.9$, determine the output weights of the RBF neural network using the least squares method. Give the equation of the designed RBF neural networks.

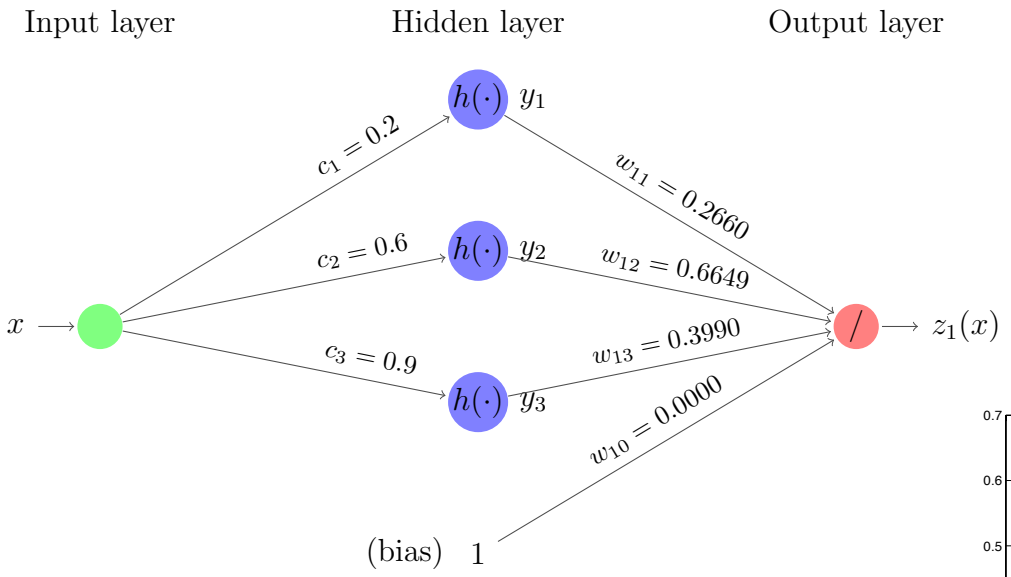
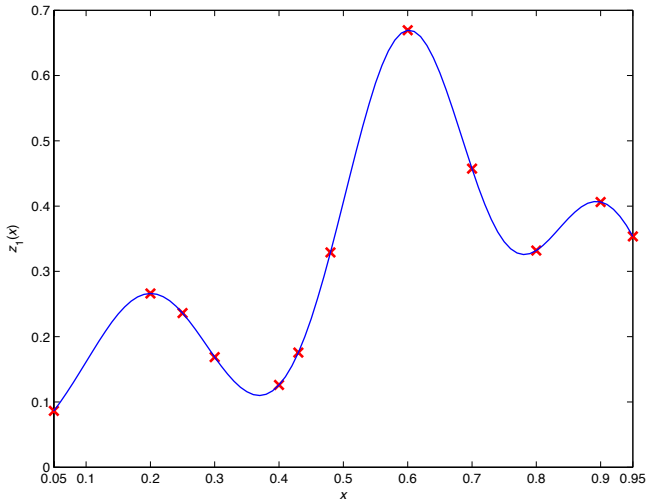


Figure 6: A diagram of 1-3-1 RBF neural network.

Remark: The design of RBF NN is done by 1) choosing **c** and σ (manually or randomly) and 2) determining the weights at the output layer using the training dataset.

p	x	z
1	0.0500	0.0863
2	0.2000	0.2662
3	0.2500	0.2362
4	0.3000	0.1687
5	0.4000	0.1260
6	0.4300	0.1756
7	0.4800	0.3290
8	0.6000	0.6694
9	0.7000	0.4573
10	0.8000	0.3320
11	0.9000	0.4063
12	0.9500	0.3535

Table 3: Input-output mappings for function $g(x)$.



Q7. Consider a classifier which is able to separate two classes, i.e.,

$$\begin{array}{ll} \text{class 1} & g(x) \geq \frac{x}{2} \\ \text{class 2} & g(x) < \frac{x}{2} \end{array}$$

where $g(x)$ is an unknown function. We have a limited information about the function $g(x)$. Table 3 lists the input-output mappings of the function $g(x)$.

d. Given clusters $S_1 = \{x_1, x_2, x_3\}$, $S_2 = \{x_4, x_5\}$, $S_3 = \{x_6, x_7, x_8, x_9\}$ and $S_4 = \{x_{10}, x_{11}, x_{12}\}$, find the centres for an RBF neural network implementing the classifier. Gaussian function with $\sigma = 2\rho_{avg}$ is used in all hidden units and linear function is used in all output units. Based on the found centres, determine the output weights of the RBF neural network using the least squares method. Draw the diagram of the designed RBF neural network and show the weights.

p	x	z
1	0.0500	0.0863
2	0.2000	0.2662
3	0.2500	0.2362
4	0.3000	0.1687
5	0.4000	0.1260
6	0.4300	0.1756
7	0.4800	0.3290
8	0.6000	0.6694
9	0.7000	0.4573
10	0.8000	0.3320
11	0.9000	0.4063
12	0.9500	0.3535

Table 3: Input-output mappings for function $g(x)$.

$$S_1 = \{x_1, x_2, x_3\} \Rightarrow c_1 = \frac{0.05+0.2+0.25}{3} = 0.1667,$$
$$S_2 = \{x_4, x_5\} \Rightarrow c_2 = \frac{0.3+0.4}{2} = 0.35,$$
$$S_3 = \{x_6, x_7, x_8, x_9\} \Rightarrow c_3 = \frac{0.43+0.48+0.6+0.7}{4} = 0.5525,$$
$$S_4 = \{x_{10}, x_{11}, x_{12}\} \Rightarrow c_4 = \frac{0.8+0.9+0.95}{3} = 0.8833.$$

Output equation of RBF neural network:

$$z_1(x) = w_{11}y_1(x) + w_{12}y_2(x) + w_{13}y_3(x) + w_{14}y_4(x) + w_{10}$$

where

$$y_1(x) = e^{-\frac{(x-c_1)^2}{2\times\sigma^2}},$$
$$y_2(x) = e^{-\frac{(x-c_2)^2}{2\times\sigma^2}},$$
$$y_3(x) = e^{-\frac{(x-c_3)^2}{2\times\sigma^2}},$$
$$y_4(x) = e^{-\frac{(x-c_4)^2}{2\times\sigma^2}}.$$

Remark: The design of RBF NN is done by 1) determining **c** and σ using clustering method and 2) determining the weights at the output layer using the training dataset.

$$\rho_{avg} = \frac{\|c_1-c_2\|+\|c_1-c_3\|+\|c_1-c_4\|+\|c_2-c_3\|+\|c_2-c_4\|+\|c_3-c_4\|}{6} = 0.3504$$
$$\Rightarrow \sigma = 2 \times 0.3921 = 0.7842.$$

Q7. Consider a classifier which is able to separate two classes, i.e.,

$$\begin{array}{ll} \text{class 1} & g(x) \geq \frac{x}{2} \\ \text{class 2} & g(x) < \frac{x}{2} \end{array}$$

where $g(x)$ is an unknown function. We have a limited information about the function $g(x)$. Table 3 lists the input-output mappings of the function $g(x)$.

d. Given clusters $S_1 = \{x_1, x_2, x_3\}$, $S_2 = \{x_4, x_5\}$, $S_3 = \{x_6, x_7, x_8, x_9\}$ and $S_4 = \{x_{10}, x_{11}, x_{12}\}$, find the centres for an RBF neural network implementing the classifier. Gaussian function with $\sigma = 2\rho_{avg}$ is used in all hidden units and linear function is used in all output units. Based on the found centres, determine the output weights of the RBF neural network using the least squares method. Draw the diagram of the designed RBF neural network and show the weights.

$$y_1(x) = e^{-\frac{(x-c_1)^2}{2\times\sigma^2}},$$
$$y_2(x) = e^{-\frac{(x-c_2)^2}{2\times\sigma^2}},$$
$$y_3(x) = e^{-\frac{(x-c_3)^2}{2\times\sigma^2}},$$
$$y_4(x) = e^{-\frac{(x-c_4)^2}{2\times\sigma^2}}.$$

p	x	z
1	0.0500	0.0863
2	0.2000	0.2662
3	0.2500	0.2362
4	0.3000	0.1687
5	0.4000	0.1260
6	0.4300	0.1756
7	0.4800	0.3290
8	0.6000	0.6694
9	0.7000	0.4573
10	0.8000	0.3320
11	0.9000	0.4063
12	0.9500	0.3535

Table 3: Input-output mappings for function $g(x)$.

p	x	$y_1(x)$	$y_2(x)$	$y_3(x)$	$y_4(x)$	Target: z
1	0.0500	0.9890	0.9294	0.8144	0.5686	0.0863
2	0.2000	0.9991	0.9819	0.9039	0.6841	0.2662
3	0.2500	0.9944	0.9919	0.9283	0.7217	0.2362
4	0.3000	0.9857	0.9980	0.9495	0.7583	0.1687
5	0.4000	0.9567	0.9980	0.9813	0.8270	0.1260
6	0.4300	0.9452	0.9948	0.9879	0.8461	0.1756
7	0.4800	0.9233	0.9864	0.9957	0.8761	0.3290
8	0.6000	0.8584	0.9505	0.9982	0.9368	0.6694
9	0.7000	0.7936	0.9052	0.9825	0.9731	0.4573
10	0.8000	0.7217	0.8482	0.9514	0.9944	0.3320
11	0.9000	0.6458	0.7820	0.9065	0.9998	0.4063
12	0.9500	0.6072	0.7462	0.8794	0.9964	0.3535

$y_1(x)$ to $y_4(x)$ at the hidden units.

Least squares methods

$$\underbrace{\begin{bmatrix} 0.9890 & 0.9294 & 0.8144 & 0.5686 & 1 \\ 0.9991 & 0.9819 & 0.9039 & 0.6841 & 1 \\ 0.9944 & 0.9919 & 0.9283 & 0.7217 & 1 \\ 0.9857 & 0.9980 & 0.9495 & 0.7583 & 1 \\ 0.9567 & 0.9980 & 0.9813 & 0.8270 & 1 \\ 0.9452 & 0.9948 & 0.9879 & 0.8461 & 1 \\ 0.9233 & 0.9864 & 0.9957 & 0.8761 & 1 \\ 0.8584 & 0.9505 & 0.9982 & 0.9368 & 1 \\ 0.7936 & 0.9052 & 0.9825 & 0.9731 & 1 \\ 0.7217 & 0.8482 & 0.9514 & 0.9944 & 1 \\ 0.6458 & 0.7820 & 0.9065 & 0.9998 & 1 \\ 0.6072 & 0.7462 & 0.8794 & 0.9964 & 1 \end{bmatrix}}_{\Phi(x)} \underbrace{\begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ w_{10} \end{bmatrix}}_{\mathbf{w}_1} = \underbrace{\begin{bmatrix} 0.0863 \\ 0.2662 \\ 0.2362 \\ 0.1687 \\ 0.1260 \\ 0.1756 \\ 0.3290 \\ 0.6694 \\ 0.4573 \\ 0.3320 \\ 0.4063 \\ 0.3535 \end{bmatrix}}_{\mathbf{t}_1}$$

Q7. Consider a classifier which is able to separate two classes, i.e.,

$$\begin{array}{ll} \text{class 1} & g(x) \geq \frac{x}{2} \\ \text{class 2} & g(x) < \frac{x}{2} \end{array}$$

where $g(x)$ is an unknown function. We have a limited information about the function $g(x)$. Table 3 lists the input-output mappings of the function $g(x)$.

e. Which of the 2 RBF neural networks perform better? Justify your answer.

$$\underbrace{\begin{bmatrix} y_1(\mathbf{x}) & y_2(\mathbf{x}) & y_3(\mathbf{x}) & y_4(\mathbf{x}) & \text{bias} \\ 0.9890 & 0.9294 & 0.8144 & 0.5686 & 1 \\ 0.9991 & 0.9819 & 0.9039 & 0.6841 & 1 \\ 0.9944 & 0.9919 & 0.9283 & 0.7217 & 1 \\ 0.9857 & 0.9980 & 0.9495 & 0.7583 & 1 \\ 0.9567 & 0.9980 & 0.9813 & 0.8270 & 1 \\ 0.9452 & 0.9948 & 0.9879 & 0.8461 & 1 \\ 0.9233 & 0.9864 & 0.9957 & 0.8761 & 1 \\ 0.8584 & 0.9505 & 0.9982 & 0.9368 & 1 \\ 0.7936 & 0.9052 & 0.9825 & 0.9731 & 1 \\ 0.7217 & 0.8482 & 0.9514 & 0.9944 & 1 \\ 0.6458 & 0.7820 & 0.9065 & 0.9998 & 1 \\ 0.6072 & 0.7462 & 0.8794 & 0.9964 & 1 \end{bmatrix}}_{\Phi(\mathbf{x})} \underbrace{\begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ w_{10} \end{bmatrix}}_{\mathbf{w}_1} = \underbrace{\begin{bmatrix} z_1(\mathbf{x}) \\ 0.0863 \\ 0.2662 \\ 0.2362 \\ 0.1687 \\ 0.1260 \\ 0.1756 \\ 0.3290 \\ 0.6694 \\ 0.4573 \\ 0.3320 \\ 0.4063 \\ 0.3535 \end{bmatrix}}_{\mathbf{t}_1}$$

Apply the pseudo-inverse method, we have $w_{11} = 129.3011$, $w_{12} = -235.3659$, $w_{13} = 129.7278$, $w_{14} = -4.6769$, $w_{10} = -11.9851$.

Output equation of RBF neural network:

$$z_1(x) = 129.3011y_1(x) - 235.3659y_2(x) + 129.7278y_3(x) - 4.6769y_4(x) - 11.9851$$

$$= 129.3011e^{-\frac{(x-c_1)^2}{2\times 0.7842^2}} - 235.3659e^{-\frac{(x-c_2)^2}{2\times 0.7842^2}} + 129.7278e^{-\frac{(x-c_3)^2}{2\times 0.7842^2}} - 4.6769e^{-\frac{(x-c_4)^2}{2\times 0.7842^2}} - 11.9851.$$

p	x	z
1	0.0500	0.0863
2	0.2000	0.2662
3	0.2500	0.2362
4	0.3000	0.1687
5	0.4000	0.1260
6	0.4300	0.1756
7	0.4800	0.3290
8	0.6000	0.6694
9	0.7000	0.4573
10	0.8000	0.3320
11	0.9000	0.4063
12	0.9500	0.3535

Table 3: Input-output mappings for function $g(x)$.

p	x	$y_1(x)$	$y_2(x)$	$y_3(x)$	$y_4(x)$	Target: z
1	0.0500	0.9890	0.9294	0.8144	0.5686	0.0863
2	0.2000	0.9991	0.9819	0.9039	0.6841	0.2662
3	0.2500	0.9944	0.9919	0.9283	0.7217	0.2362
4	0.3000	0.9857	0.9980	0.9495	0.7583	0.1687
5	0.4000	0.9567	0.9980	0.9813	0.8270	0.1260
6	0.4300	0.9452	0.9948	0.9879	0.8461	0.1756
7	0.4800	0.9233	0.9864	0.9957	0.8761	0.3290
8	0.6000	0.8584	0.9505	0.9982	0.9368	0.6694
9	0.7000	0.7936	0.9052	0.9825	0.9731	0.4573
10	0.8000	0.7217	0.8482	0.9514	0.9944	0.3320
11	0.9000	0.6458	0.7820	0.9065	0.9998	0.4063
12	0.9500	0.6072	0.7462	0.8794	0.9964	0.3535

Q7. Consider a classifier which is able to separate two classes, i.e.,

$$\begin{array}{ll} \text{class 1} & g(x) \geq \frac{x}{2} \\ \text{class 2} & g(x) < \frac{x}{2} \end{array}$$

where $g(x)$ is an unknown function. We have a limited information about the function $g(x)$. Table 3 lists the input-output mappings of the function $g(x)$.

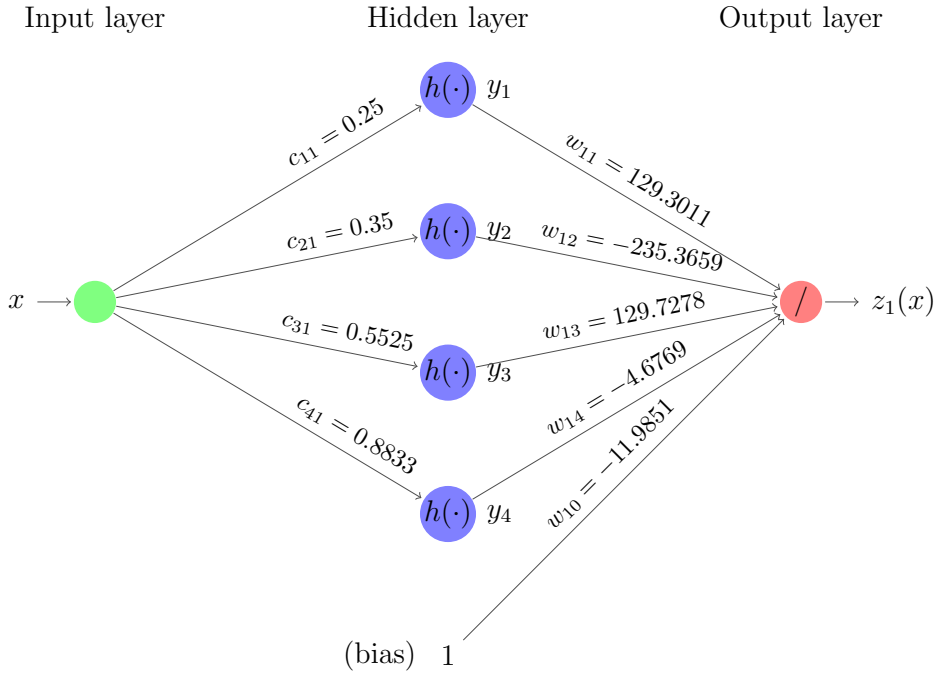
e. Which of the 2 RBF neural networks perform better? Justify your answer.

Apply the pseudo-inverse method, we have $w_{11} = 129.3011$, $w_{12} = -235.3659$, $w_{13} = 129.7278$, $w_{14} = -4.6769$, $w_{10} = -11.9851$.

Output equation of RBF neural network:

$$z_1(x) = 129.3011y_1(x) - 235.3659y_2(x) + 129.7278y_3(x) - 4.6769y_4(x) - 11.9851$$

$$= 129.3011e^{-\frac{(x-c_1)^2}{2\times 0.7842^2}} - 235.3659e^{-\frac{(x-c_2)^2}{2\times 0.7842^2}} + 129.7278e^{-\frac{(x-c_3)^2}{2\times 0.7842^2}} - 4.6769e^{-\frac{(x-c_4)^2}{2\times 0.7842^2}} - 11.9851.$$



A diagram of 1-4-1 RBF neural network.

p	x	z
1	0.0500	0.0863
2	0.2000	0.2662
3	0.2500	0.2362
4	0.3000	0.1687
5	0.4000	0.1260
6	0.4300	0.1756
7	0.4800	0.3290
8	0.6000	0.6694
9	0.7000	0.4573
10	0.8000	0.3320
11	0.9000	0.4063
12	0.9500	0.3535

Table 3: Input-output mappings for function $g(x)$.

p	x	$y_1(x)$	$y_2(x)$	$y_3(x)$	$y_4(x)$	Target: z
1	0.0500	0.9890	0.9294	0.8144	0.5686	0.0863
2	0.2000	0.9991	0.9819	0.9039	0.6841	0.2662
3	0.2500	0.9944	0.9919	0.9283	0.7217	0.2362
4	0.3000	0.9857	0.9980	0.9495	0.7583	0.1687
5	0.4000	0.9567	0.9980	0.9813	0.8270	0.1260
6	0.4300	0.9452	0.9948	0.9879	0.8461	0.1756
7	0.4800	0.9233	0.9864	0.9957	0.8761	0.3290
8	0.6000	0.8584	0.9505	0.9982	0.9368	0.6694
9	0.7000	0.7936	0.9052	0.9825	0.9731	0.4573
10	0.8000	0.7217	0.8482	0.9514	0.9944	0.3320
11	0.9000	0.6458	0.7820	0.9065	0.9998	0.4063
12	0.9500	0.6072	0.7462	0.8794	0.9964	0.3535

Q7. Consider a classifier which is able to separate two classes, i.e.,

$$\begin{array}{ll} \text{class 1} & g(x) \geq \frac{x}{2} \\ \text{class 2} & g(x) < \frac{x}{2} \end{array}$$

where $g(x)$ is an unknown function. We have a limited information about the function $g(x)$. Table 3 lists the input-output mappings of the function $g(x)$.

e. Which of the 2 RBF neural networks perform better? Justify your answer.

$$J = \sum_{p=1}^{12} (z_1(x_p) - t_p)^2 \text{ (sum of squared errors)}$$

			$1-3-1$ RBF network		$1-4-1$ RBF network	
p	x	Target $z_1(x)$	$z_1(x)$	$(z_1(x)-t_p)^2$	$z_1(x)$	$(z_1(x)-t_p)^2$
1	0.0500	0.0863	0.0864	0.0000	0.1254	0.0015
2	0.2000	0.2662	0.2662	0.0000	0.1627	0.0107
3	0.2500	0.2362	0.2362	0.0000	0.1795	0.0032
4	0.3000	0.1687	0.1687	0.0000	0.2012	0.0011
5	0.4000	0.1260	0.1260	0.0000	0.2607	0.0182
6	0.4300	0.1756	0.1756	0.0000	0.2823	0.0114
7	0.4800	0.3290	0.3290	0.0000	0.3210	0.0001
8	0.6000	0.6694	0.6694	0.0000	0.4151	0.0647
9	0.7000	0.4573	0.4573	0.0000	0.4708	0.0002
10	0.8000	0.3320	0.3320	0.0000	0.4747	0.0204
11	0.9000	0.4063	0.4064	0.0000	0.3960	0.0001
12	0.9500	0.3535	0.3536	0.0000	0.3168	0.0013

Squared errors for both RBF neural networks.

For the $1-3-1$ RBF neural network, $J = 1.009 \times 10^{-8} \approx 0$.

For the $1-4-1$ RBF neural network, $J = 0.0664$.

As the sum of squared errors from the $1-3-1$ RBF neural network is smaller, it is concluded that the $1-3-1$ RBF neural network performs better than the $1-4-1$ RBF neural network.

Any another criteria for comparison?

p	x	z
1	0.0500	0.0863
2	0.2000	0.2662
3	0.2500	0.2362
4	0.3000	0.1687
5	0.4000	0.1260
6	0.4300	0.1756
7	0.4800	0.3290
8	0.6000	0.6694
9	0.7000	0.4573
10	0.8000	0.3320
11	0.9000	0.4063
12	0.9500	0.3535

Table 3: Input-output mappings for function $g(x)$.

Q8. A nonlinear classifier with multiple inputs and single output is implemented by a feedforward neural network. All inputs are in the range of -1 and 1. Propose a method to replace the feedforward neural network with an RBF neural network.

The following procedure is employed to redesign the classifier using an RBF neural network.

Step 1: Generate random input patterns $\mathbf{x}_p \in \Re^{d \times p}$, $p = 1, \ldots, n$, of which each entry is in the range of -1 and 1. Apply the input patterns to the feedback forward neural network and collect the output as $\mathbf{t}_p \in \Re^p$.

Step 2: Form the dataset with the obtained input patterns and $\mathbf{x}_p$ and output classes $\mathbf{t}_p$

Step 3: Divide the dataset into training, test and validation sets, say, in a ration of 70%:15%:15%.

Step 4: Choose the activation functions for the hidden and output units.

Step 5: Determine the number of hidden units and centres by some algorithms.

Step 6: If linear function is used as the activation functions in the output units, the whole dataset is employed to determine the output weights using the pseudo inverse method. If nonlinear function is used as the activation functions in the output units, stochastic or batch backpropagation algorithm is employed to learn the parameters (output weights and parameters of activation functions, and even the centres) using the training set. One of the stopping criteria is that the training stop when the MSE for the validation set starts increasing.

Step 7: Evaluate the performance of the trained RBF network using the test set. If the performance is acceptable, stop the procedure, otherwise, go to step 5.

Pattern Recognition, Neural Networks and Deep Learning (7CCSMPNN)

Tutorial 8: Solutions

Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

a. Determine the best kernel function for this classification problem.

b. Design an SVM classifier to classify all given points.

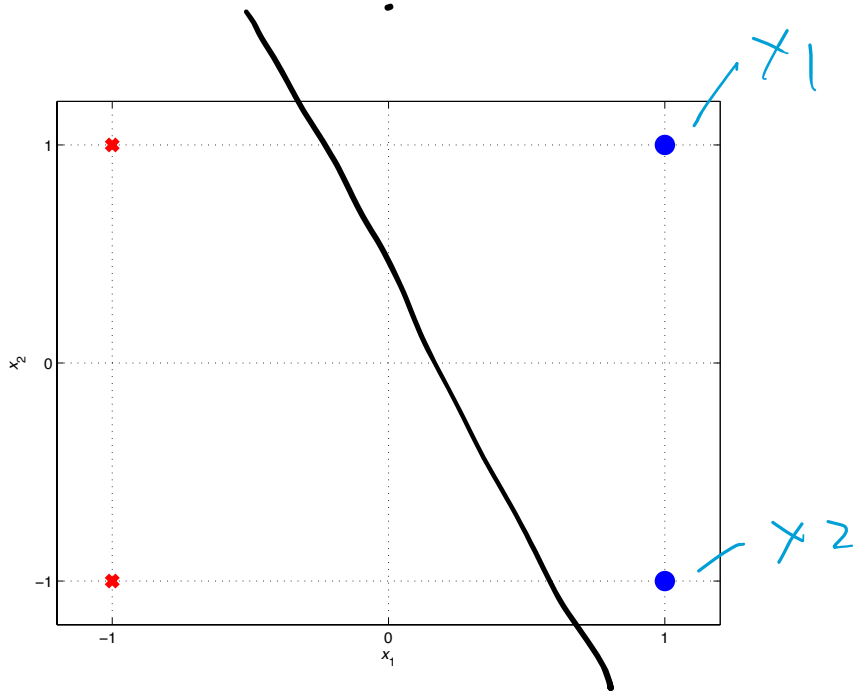
c. Identify the support vectors.

Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$
$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

a. Determine the best kernel function for this classification problem.

From the figure, the two classes can be separated by a straight line. It is thus linearly separable, which suggests that the best kernel is a linear kernel function.



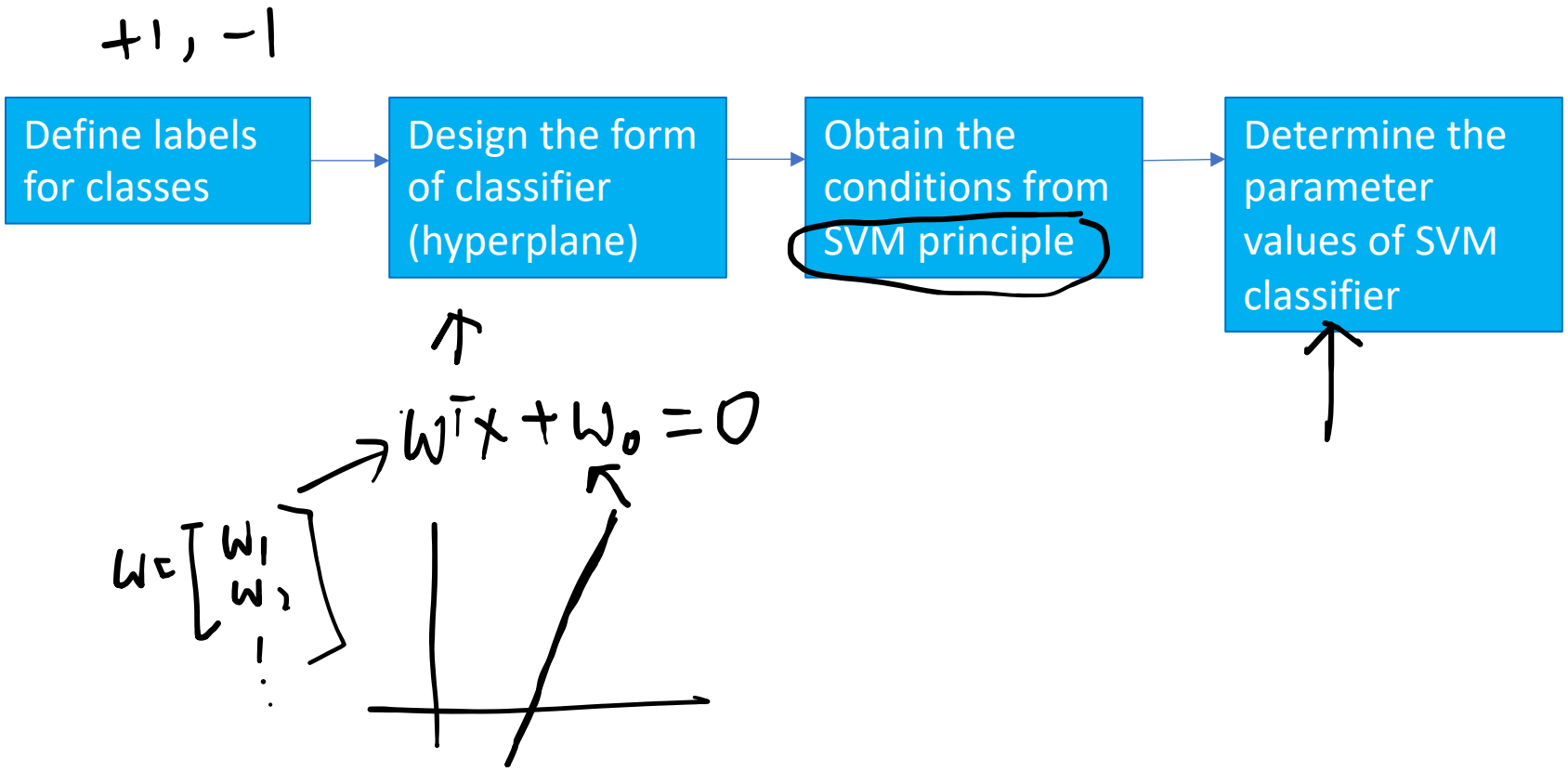
Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}. \qquad +1$$

$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}. \qquad -1$$

a. Determine the best kernel function for this classification problem.

+1, -1



Q1. An SVM classifier is employed to classify the following points:

Class 1: $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

Class 2: $\mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$, $\mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$.

b. Design an SVM classifier to classify all given points.

Define the label for Class 1 as $+1$ and Class 2 as -1 so we have $y_1 = y_2 = 1$ and $y_3 = y_4 = -1$. ← Defined by the designer

The hyperplane is: $\mathbf{w}^T\mathbf{x} + w_0 = w_1 x_1 + w_2 x_2 + w_0 = 0$. ← Data in 2 dimensional space

Primal problem:

Y_i is the out for input

$$\mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{4} \lambda_i (y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1)$$

$$= \frac{w_1^2 + w_2^2}{2} - \lambda_1(w_1 + w_2 + w_0 - 1) - \lambda_2(w_1 - w_2 + w_0 - 1)$$

$$- \lambda_3(w_1 - w_2 - w_0 - 1) - \lambda_4(w_1 + w_2 - w_0 - 1)$$

For x_1 | For x_2 | For x_3 | For x_4

W1 and w2 is because we have two input vectors in x1 altogether we have four inputs and w sign reflects with a vectors input

Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$
$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

b. Design an SVM classifier to classify all given points.

Consider the term: $\lambda_i(y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1)$

Take $i = 1$ as an example, we have $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $y_1 = 1$ and $\lambda_1(y_1(\mathbf{w}^T\mathbf{x}_1 + w_0) - 1)$.

After expending, with $\mathbf{w}^T = \begin{bmatrix} w_1 & w_2 \end{bmatrix}$ we have

$$\lambda_1(1 \times (\begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + w_0) - 1) = \lambda_1(w_1 + w_2 + w_0 - 1)$$

Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

From p. 16 of lecture notes:

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$

$$\lambda_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \right) = 0, \quad i = 1, 2, \ldots, N$$

b. Design an SVM classifier to classify all given points.

$$\begin{aligned} \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda}) &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1} \lambda_i (y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1) \\ &= \frac{w_1^2 + w_2^2}{2} - \lambda_1(w_1 + w_2 + w_0 - 1) - \lambda_2(w_1 - w_2 + w_0 - 1) \\ &\quad - \lambda_3(w_1 - w_2 - w_0 - 1) - \lambda_4(w_1 + w_2 - w_0 - 1) \end{aligned}$$

Step 1: $\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_1} = 0 \Rightarrow w_1 = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4$

$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_2} = 0 \Rightarrow w_2 = \lambda_1 - \lambda_2 - \lambda_3 + \lambda_4$

Step 2: $\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \lambda_1 + \lambda_2 - \lambda_3 - \lambda_4 = 0$

Minus here from mainly because of class

Step 3: $\lambda_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \right) = 0$ leading to:

$$\lambda_1(w_1 + w_2 + w_0 - 1) = 0 \quad (1)$$

$$\lambda_2(w_1 - w_2 + w_0 - 1) = 0 \quad (2)$$

$$\lambda_3(w_1 - w_2 - w_0 - 1) = 0 \quad (3)$$

$$\lambda_4(w_1 + w_2 - w_0 - 1) = 0 \quad (4)$$

w_1^2 w_2^2

Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$
$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

b. Design an SVM classifier to classify all given points.

From p. 16 of lecture notes:

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$
$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$
$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$
$$\lambda_i (y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1) = 0, \quad i = 1, 2, \ldots, N$$

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \sum_{i=1}^{4} \lambda_i y_i \mathbf{x}_i$$
$$= \lambda_1 y_1 \mathbf{x}_1 + \lambda_2 y_2 \mathbf{x}_2 + \lambda_1 y_3 \mathbf{x}_3 + \lambda_4 y_4 \mathbf{x}_4$$
$$= \lambda_1 \times 1 \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \lambda_2 \times 1 \times \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \lambda_3 \times -1 \times \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \lambda_4 \times -1 \times \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$
$$= \begin{bmatrix} \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \\ \lambda_1 - \lambda_2 - \lambda_3 + \lambda_4 \end{bmatrix}$$

$$\sum_{i=1}^{4} \lambda_i y_i = \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 + \lambda_4 y_4$$
$$= \lambda_1 \times 1 + \lambda_2 \times 1 + \lambda_3 \times -1 + \lambda_4 \times -1$$
$$= \lambda_1 + \lambda_2 - \lambda_3 - \lambda_4 = 0$$

Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$
$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

b. Design an SVM classifier to classify all given points.

From p. 16 of lecture notes:

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$
$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$
$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$
$$\lambda_i (y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1) = 0, \quad i = 1, 2, \ldots, N$$

Step 4: $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$ How many variables we need to find?

(1) + (2) + (3) + (4): $w_1^2 + w_2^2 - w_1 = w_1(w_1 - 1) + w_2^2 = 0$

Also consider the condition of $y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \geq 0$ leading to

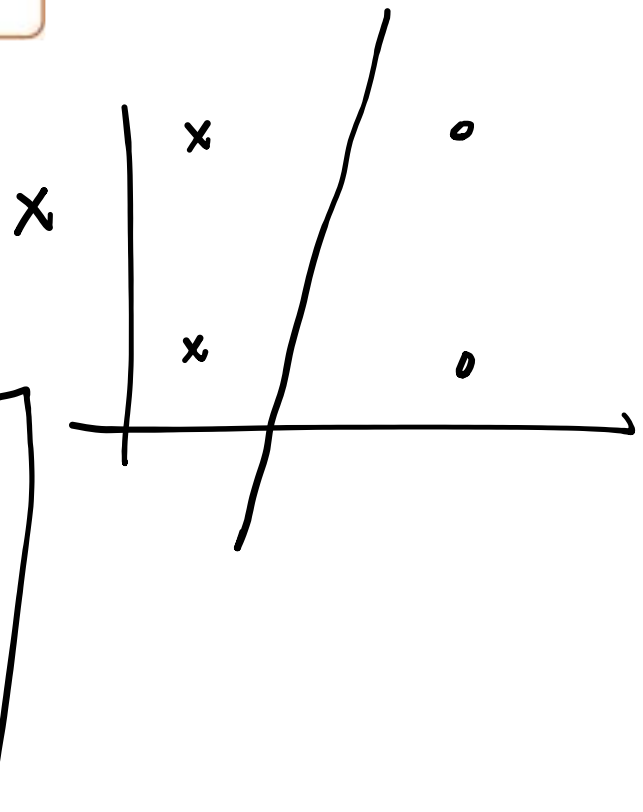
x_1 $w_1 + w_2 + w_0 - 1 \geq 0$ (5)

x_2 $w_1 - w_2 + w_0 - 1 \geq 0$ (6)

x_3 $w_1 - w_2 - w_0 - 1 \geq 0$ (7)

x_4 $w_1 + w_2 - w_0 - 1 \geq 0$ (8)

The sum of these for inequalities gives $w_1 \geq 1$.



Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$
$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

b. Design an SVM classifier to classify all given points.

From p. 16 of lecture notes:

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$
$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$
$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$
$$\lambda_i (y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1) = 0, \quad i = 1, 2, \ldots, N$$

$$w_1(w_1 - 1) + w_2^2 = 0$$

$$w_1 \geq 1$$

$w_1 > 1$, $w_1 = 1.01$

$w_2^2 = -1.01(1.01 - 1)$ ← X

$$\rightarrow w_2^2 = -w_1(w_1 - 1)$$

$w_1 = 1$

$w_2^2 = -1(1 - 1) = 0 \Rightarrow w_2 = 0$

Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

b. Design an SVM classifier to classify all given points.

From p. 16 of lecture notes:

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$

$$\lambda_i (y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1) = 0, \quad i = 1, 2, \ldots, N$$

$$w_1(w_1 - 1) + w_2^2 = 0$$

$$w_1 \geq 1$$

Looking at $w_1(w_1 - 1) + w_2^2 = 0$, if $w_1 > 1$, $w_2^2 = -w_1(w_1 - 1)$ which is not possible.

The only solution is $w_1 = 1$, then $w_2^2 = -w_1(w_1 - 1) = 0 \Rightarrow w_2 = 0$.

w_0

Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$
$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

b. Design an SVM classifier to classify all given points.

From p. 16 of lecture notes:

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$
$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$
$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$
$$\lambda_i (y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1) = 0, \quad i = 1, 2, \ldots, N$$

Substituting $w_1 = 1$ and $w_2 = 0$ to (1) to (4) and considering (1) + (2) − (3) − (4), we have:

$$\lambda_1(1 + w_0 - 1) + \lambda_2(1 + w_0 - 1) - \lambda_3(1 - w_0 - 1) - \lambda_4(1 - w_0 - 1) = w_0 = 0.$$

$w_1 = 1$
$w_2 = 0$
$w_0 = 0$

The linear SVM classifier is: $\text{sgn}(w_1 x_1 + w_2 x_2 + w_0) = \text{sgn}(x_1)$ (hard classifier)

$$\lambda_1(w_1 + w_2 + w_0 - 1) = 0 \quad (1)$$
$$\lambda_2(w_1 - w_2 + w_0 - 1) = 0 \quad (2)$$
$$\lambda_3(w_1 - w_2 - w_0 - 1) = 0 \quad (3)$$
$$\lambda_4(w_1 + w_2 - w_0 - 1) = 0 \quad (4)$$

Step 1: $\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_1} = 0 \Rightarrow w_1 = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4$

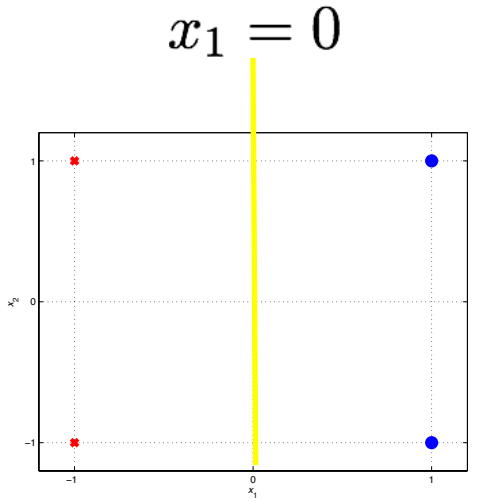
$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_2} = 0 \Rightarrow w_2 = \lambda_1 - \lambda_2 - \lambda_3 + \lambda_4$

Step 2: $\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \lambda_1 + \lambda_2 - \lambda_3 - \lambda_4 = 0$

Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$



$w_1 x_1 + w_2 x_2 + w_0 = 0$

$x_1 = 0$

c. Identify the support vectors.

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$

$$\lambda_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \right) = 0, \quad i = 1, 2, \ldots, N$$

$$\begin{aligned} \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} &= \sum_{i=1}^{4} \lambda_i y_i \mathbf{x}_i \\ &= \lambda_1 y_1 \mathbf{x}_1 + \lambda_2 y_2 \mathbf{x}_2 + \lambda_1 y_3 \mathbf{x}_3 + \lambda_4 y_4 \mathbf{x}_4 \\ &= \lambda_1 \times 1 \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \lambda_2 \times 1 \times \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \lambda_3 \times -1 \times \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \lambda_4 \times -1 \times \begin{bmatrix} -1 \\ -1 \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \\ \lambda_1 - \lambda_2 - \lambda_3 + \lambda_4 \end{bmatrix} \end{aligned}$$

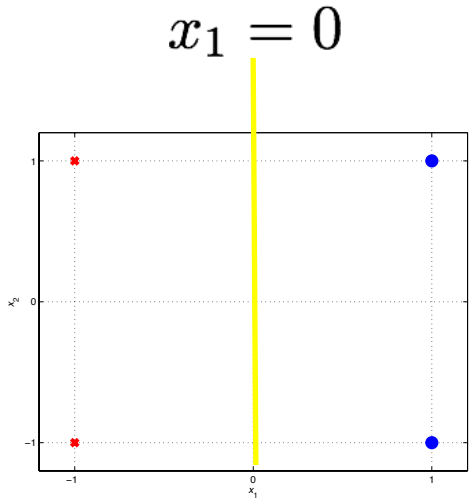
$$\begin{aligned} \sum_{i=1}^{4} \lambda_i y_i &= \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 + \lambda_4 y_4 \\ &= \lambda_1 \times 1 + \lambda_2 \times 1 + \lambda_3 \times -1 + \lambda_4 \times -1 \\ &= \lambda_1 + \lambda_2 - \lambda_3 - \lambda_4 = 0 \end{aligned}$$

Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

c. Identify the support vectors.



Step 1: $\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_1} = 0 \Rightarrow w_1 = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4$

$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_2} = 0 \Rightarrow w_2 = \lambda_1 - \lambda_2 - \lambda_3 + \lambda_4$

Step 2: $\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \lambda_1 + \lambda_2 - \lambda_3 - \lambda_4 = 0$

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$$

$$\lambda_1 - \lambda_2 - \lambda_3 + \lambda_4 = 0$$

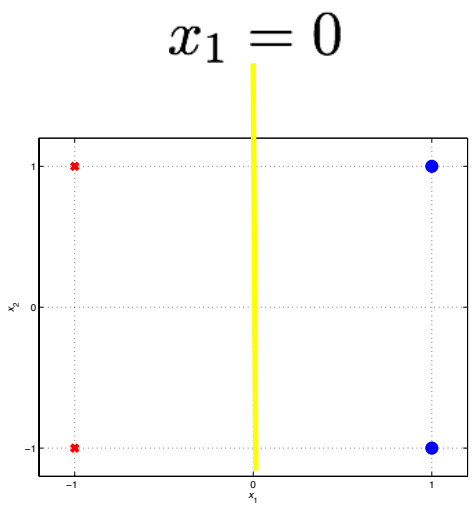
$$\lambda_1 + \lambda_2 - \lambda_3 - \lambda_4 = 0$$

Q1. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

c. Identify the support vectors.



As the hyperplane is $x_1 = 0$, it is a vertical line separating two classes. By shifting the hyperplane to the left and to the right, it will touch all four points. It implies that $\mathbf{x}_1$ to $\mathbf{x}_4$ are the support vectors. It can be verified by finding the value of λ_i. From the conditions of $w_1 = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$, $w_2 = \lambda_1 - \lambda_2 - \lambda_3 + \lambda_4 = 0$ and $\lambda_1 + \lambda_2 - \lambda_3 - \lambda_4 = 0$, it gives $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0.25$. The $\mathbf{x}_i$ with non-zero λ_i is a support vector.

Recalling that $\mathbf{w} = \sum_{i \in SVs} \lambda_i y_i \mathbf{x}_i = 0.25 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0.25 \begin{bmatrix} 1 \\ -1 \end{bmatrix} - 0.25 \begin{bmatrix} -1 \\ 1 \end{bmatrix} - 0.25 \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ which produces the same result that $w_1 = 1$ and $w_2 = 0$.

Recalling that the hyperplane is $\mathbf{w}^T \mathbf{x} + w_0 = w_1 x_1 + w_2 x_2 + w_0 = x_1 + w_0$. When $\mathbf{x}$ is a support vector, the value of $x_1 + w_0$ is 1 (if $\mathbf{x}$ belongs to the class '+1') or -1 (if $\mathbf{x}$ belongs to the class '-1'). So, for example, using $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and its class is '+1', we have $x_1 + w_0 = 1 + w_0 = 1$ which gives $w_0 = 0$.

Q2. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 7 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} 8 \\ 0 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}.$$

a. Determine if the dataset is linearly separable.

b. Identify the support vectors by inspection.

c. Design an SVM classifier to classify all given points. What is the margin?

d. Classify the point $\mathbf{x} = \begin{bmatrix} 2.5 \\ 0.5 \end{bmatrix}$ using the designed SVM classifier.

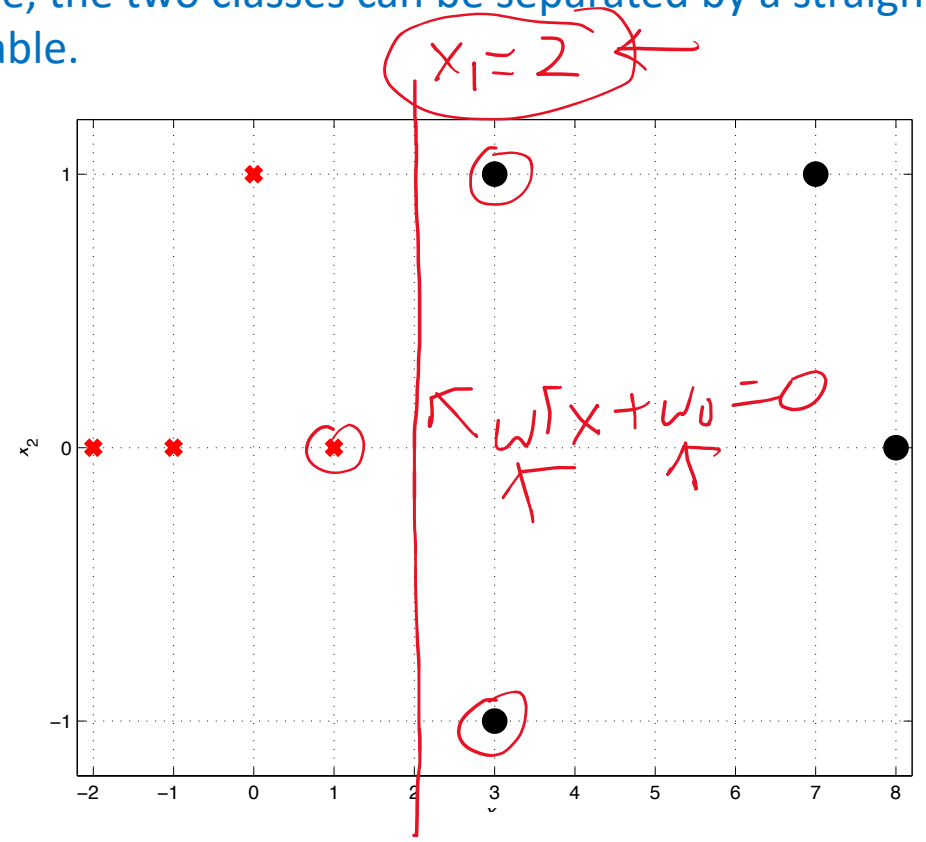
Q2. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 7 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} 8 \\ 0 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}.$$

a. Determine if the dataset is linearly separable.

From the figure, the two classes can be separated by a straight line. It is thus linearly separable.



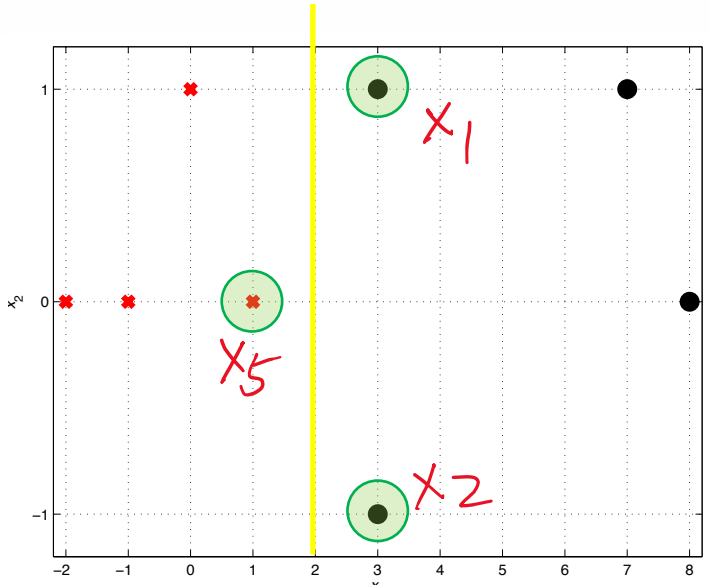
Q2. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 7 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} 8 \\ 0 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}.$$

b. Identify the support vectors by inspection.

By inspection, the two classes can be separated by constructing a hyperplane in the region of $1 \leq x_1 \leq 3$. The support vectors are: $\mathbf{x}_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$, $\mathbf{x}_2 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$, $\mathbf{x}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.



Q2. An SVM classifier is employed to classify the following points:

Class 1: $\mathbf{x}_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 7 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} 8 \\ 0 \end{bmatrix}$. ← Class label: 1

Class 2: $\mathbf{x}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$. ← Class label: – 1

c. Design an SVM classifier to classify all given points. What is the margin?

Define the label for Class 1 as +1 and Class 2 as −1:

$y_1 = y_2 = y_3 = y_4 = 1,$

$y_5 = y_6 = y_7 = y_8 = -1.$

Hyperplane: $\mathbf{w}^T\mathbf{x} + w_0 = 0$

$$\mathbf{w} = \lambda_1 y_1 \mathbf{x}_1 + \lambda_2 y_2 \mathbf{x}_2 + \lambda_5 y_5 \mathbf{x}_5$$

$$= \lambda_1 \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \lambda_2 \begin{bmatrix} 3 \\ -1 \end{bmatrix} - \lambda_5 \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$

$$\lambda_i (y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1) = 0, \quad i = 1, 2, \ldots, N$$

Remark: Except λ_1, λ_2 and λ_5 are non zero (support vectors), the rest λ are all zero.

Q2. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 7 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} 8 \\ 0 \end{bmatrix}. \quad \leftarrow \text{Class label: 1}$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}. \quad \leftarrow \text{Class label: } -1$$

c. Design an SVM classifier to classify all given points. What is the margin?

Define the label for Class 1 as +1 and Class 2 as −1:
$y_1 = y_2 = y_3 = y_4 = 1$,
$y_5 = y_6 = y_7 = y_8 = -1$.

$$\mathbf{w} = \lambda_1 y_1 \mathbf{x}_1 + \lambda_2 y_2 \mathbf{x}_2 + \lambda_5 y_5 \mathbf{x}_5 = \lambda_1 \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \lambda_2 \begin{bmatrix} 3 \\ -1 \end{bmatrix} - \lambda_5 \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Recall that $y_i(\mathbf{w}^T\mathbf{x} + w_0) = 1$ when $\mathbf{x}$ is a support vector.

$\mathbf{x} = \mathbf{x}_1$, $y_1 = 1$:

$$y_1(\mathbf{w}^T\mathbf{x}_1 + w_0) = 1 \times \left(\left(\lambda_1 \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \lambda_2 \begin{bmatrix} 3 \\ -1 \end{bmatrix} - \lambda_5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)^T \begin{bmatrix} 3 \\ 1 \end{bmatrix} + w_0 \right) = 1$$

$$\Rightarrow 10\lambda_1 + 8\lambda_2 - 3\lambda_5 + w_0 = 1.$$

$\mathbf{x} = \mathbf{x}_2$, $y_2 = 1$:

$$y_2(\mathbf{w}^T\mathbf{x}_2 + w_0) = 1 \times \left(\left(\lambda_1 \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \lambda_2 \begin{bmatrix} 3 \\ -1 \end{bmatrix} - \lambda_5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)^T \begin{bmatrix} 3 \\ -1 \end{bmatrix} + w_0 \right) = 1$$

$$\Rightarrow 8\lambda_1 + 10\lambda_2 - 3\lambda_5 + w_0 = 1.$$

$\mathbf{x} = \mathbf{x}_5$, $y_5 = -1$:

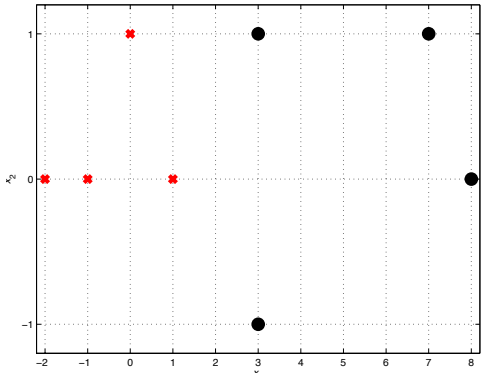
$$y_5(\mathbf{w}^T\mathbf{x}_5 + w_0) = -1 \times \left(\left(\lambda_1 \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \lambda_2 \begin{bmatrix} 3 \\ -1 \end{bmatrix} - \lambda_5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)^T \begin{bmatrix} 1 \\ 0 \end{bmatrix} + w_0 \right) = 1$$

$$\Rightarrow 3\lambda_1 + 3\lambda_2 - \lambda_5 + w_0 = -1.$$

Q2. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 7 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} 8 \\ 0 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}.$$



c. Design an SVM classifier to classify all given points. What is the margin?

$$10\lambda_1 + 8\lambda_2 - 3\lambda_5 + w_0 = 1.$$
$$8\lambda_1 + 10\lambda_2 - 3\lambda_5 + w_0 = 1.$$
$$3\lambda_1 + 3\lambda_2 - \lambda_5 + w_0 = -1.$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$

$$\lambda_i (y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1) = 0, \quad i = 1, 2, \ldots, N$$

$$\sum_{i=1}^{8} \lambda_i y_i = 0 \Rightarrow \lambda_1 + \lambda_2 - \lambda_5 = 0.$$

$$\begin{bmatrix} 10 & 8 & -3 & 1 \\ 8 & 10 & -3 & 1 \\ 3 & 3 & -1 & 1 \\ 1 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_5 \\ w_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 0 \end{bmatrix}.$$

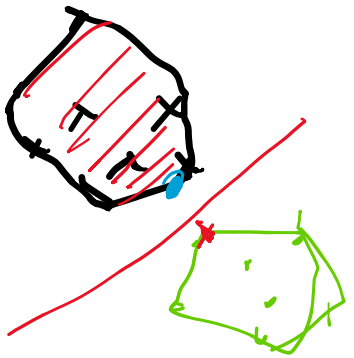
Lambda they are not support vectors and they have to be zero

$$\lambda_1 = \lambda_2 = 0.25, \lambda_5 = 0.5 \text{ and } w_0 = -2.$$

$$\mathbf{w} = 0.25 \begin{bmatrix} 3 \\ 1 \end{bmatrix} + 0.25 \begin{bmatrix} 3 \\ -1 \end{bmatrix} - 0.5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$w_1 = 1$

$w_2 = 0$



Q2. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 7 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} 8 \\ 0 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}.$$

c. Design an SVM classifier to classify all given points. What is the margin?

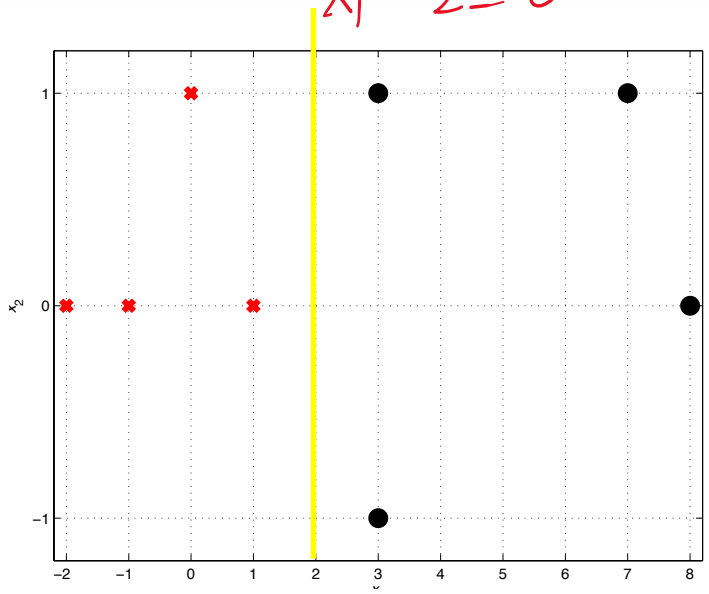
$$\mathbf{w} = 0.25 \begin{bmatrix} 3 \\ 1 \end{bmatrix} + 0.25 \begin{bmatrix} 3 \\ -1 \end{bmatrix} - 0.5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The hyperplane is $\mathbf{w}^T\mathbf{x} + w_0 = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 2 = x_1 - 2 = 0.$

The margin is 2.

On p. 2 of lecture note:

Margin: $$\frac{2}{\|\mathbf{w}\|} = \frac{2}{\sqrt{w_1^2 + w_2^2}} = \frac{2}{\sqrt{1^2 + 0^2}} = 2$$



Q2. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 7 \\ 1 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} 8 \\ 0 \end{bmatrix}.$$

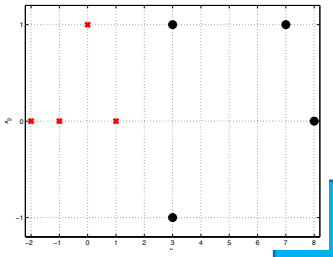
$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}.$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$

$$\lambda_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \right) = 0, \quad i = 1, 2, \ldots, N$$



Assign class labels, plot data and identify support vectors by observation

→ Represent **w** using the identified support vectors as:

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

→ Construct linear equations in terms of λ using the support vectors:

$$y_i(\mathbf{w}^T \mathbf{x} + w_0) = 1$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

↓

Solve the linear equations (in matrix form) to find λ and w_0

→ Construct w using with the found λ as:

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

→ Construct the hyperplane as:

$$\mathbf{w}^T \mathbf{x} + w_0 = 0$$

Q3. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

a. Determine if the dataset is linearly separable.

b. Apply the feature mapping function $\Phi(\mathbf{x}) = \begin{cases} \begin{bmatrix} 4 - \frac{x_2}{2} + |x_1 - x_2| \\ 4 - \frac{x_1}{2} + |x_1 - x_2| \end{bmatrix} & \text{if } \|\mathbf{x}\| > 2 \\ \begin{bmatrix} x_1 - 2 \\ x_2 - 3 \end{bmatrix} & \text{Otherwise} \end{cases}$

to the dataset where $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Determine if the dataset is linearly separable in the new feature space.

c. Identify the support vectors by inspection.

d. Design a nonlinear SVM classifier to classify all given points.

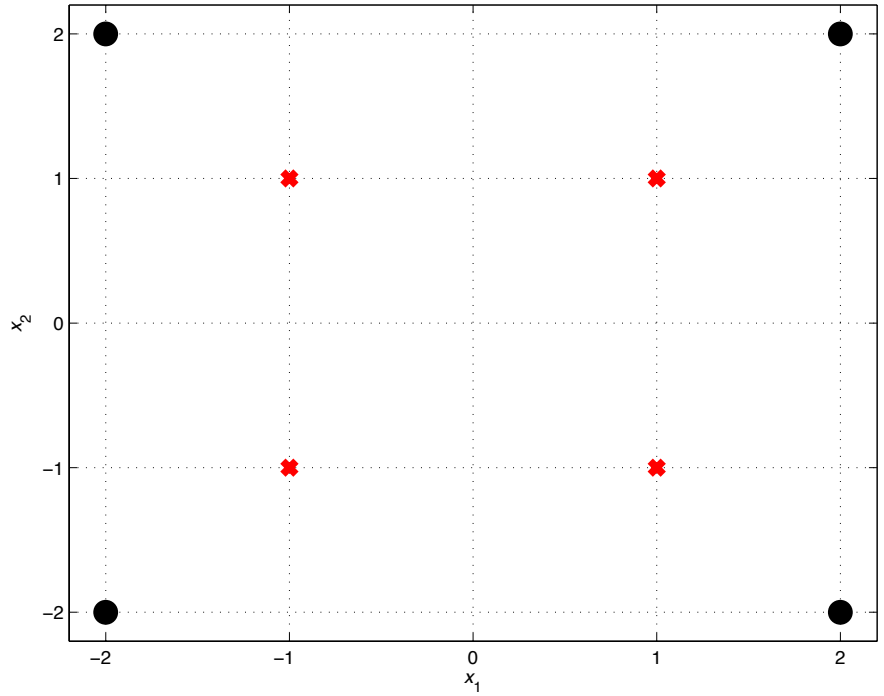
Q3. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

a. Determine if the dataset is linearly separable.

The two classes cannot be separated by a straight line. It is thus non-linearly separable.



What can be done to the data?

Q3. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

b. Apply the feature mapping function $\Phi(\mathbf{x}) = \begin{cases} \begin{bmatrix} 4 - \frac{x_2}{2} + |x_1 - x_2| \\ 4 - \frac{x_1}{2} + |x_1 - x_2| \end{bmatrix} & \text{if } \|\mathbf{x}\| > 2 \\ \begin{bmatrix} x_1 - 2 \\ x_2 - 3 \end{bmatrix} & \text{Otherwise} \end{cases}$

to the dataset where $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Determine if the dataset is linearly separable in the new feature space.

Take $\mathbf{x}_2$ as an example: $x_1 = 2$ and $x_2 = -2$

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2} = \sqrt{2^2 + (-2)^2} = \sqrt{8} > 2$$

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 4 - \frac{x_2}{2} + |x_1 - x_2| \\ 4 - \frac{x_1}{2} + |x_1 - x_2| \end{bmatrix} = \begin{bmatrix} 4 - \frac{-2}{2} + |2 - (-2)| \\ 4 - \frac{2}{2} + |2 - (-2)| \end{bmatrix} = \begin{bmatrix} 9 \\ 7 \end{bmatrix}$$

Take $\mathbf{x}_6$ as an example: $x_1 = 1$ and $x_2 = -1$

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2} = \sqrt{1^2 + (-1)^2} = \sqrt{2} < 2$$

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 - 2 \\ x_2 - 3 \end{bmatrix} = \begin{bmatrix} 1 - 2 \\ -1 - 3 \end{bmatrix} = \begin{bmatrix} -1 \\ -4 \end{bmatrix}$$

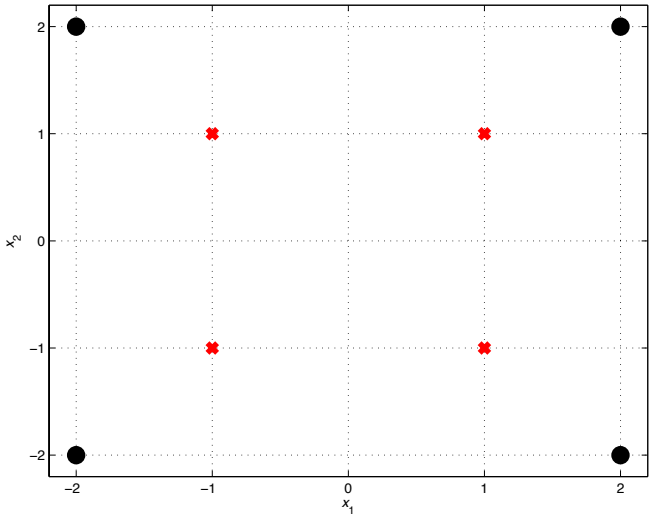
Q3. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

b. Apply the feature mapping function $\Phi(\mathbf{x}) = \begin{cases} \begin{bmatrix} 4 - \frac{x_2}{2} + |x_1 - x_2| \\ 4 - \frac{x_1}{2} + |x_1 - x_2| \end{bmatrix} & \text{if } \|\mathbf{x}\| > 2 \\ \begin{bmatrix} x_1 - 2 \\ x_2 - 3 \end{bmatrix} & \text{Otherwise} \end{cases}$

to the dataset where $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Determine if the dataset is linearly separable in the new feature space.

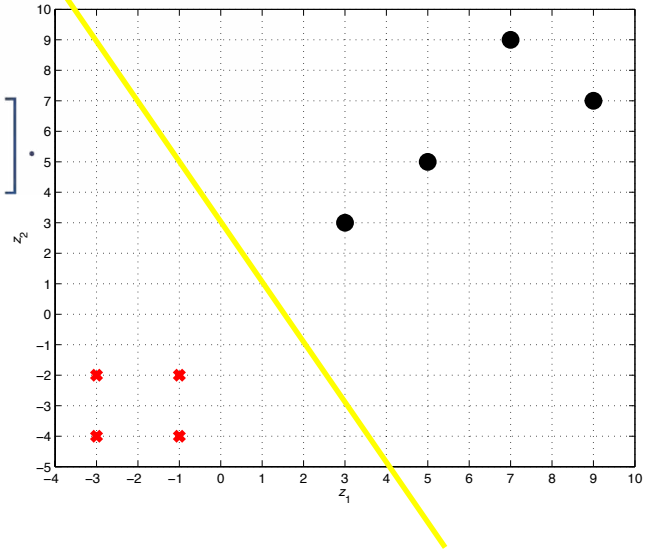


Applying the feature mapping function to $\mathbf{x}$, we obtain dataset in a new feature space, i.e., $\mathbf{z} = \Phi(\mathbf{x})$:

These vectors here are the obtained once we applied mapping like it's previous slide

$$\text{Class 1: } \mathbf{z}_1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}, \mathbf{z}_2 = \begin{bmatrix} 9 \\ 7 \end{bmatrix}, \mathbf{z}_3 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \mathbf{z}_4 = \begin{bmatrix} 7 \\ 9 \end{bmatrix};$$

$$\text{Class 2: } \mathbf{z}_5 = \begin{bmatrix} -1 \\ -2 \end{bmatrix}, \mathbf{z}_6 = \begin{bmatrix} -1 \\ -4 \end{bmatrix}, \mathbf{z}_7 = \begin{bmatrix} -3 \\ -4 \end{bmatrix}, \mathbf{z}_8 = \begin{bmatrix} -3 \\ -2 \end{bmatrix}.$$



The dataset in feature space **z** is linearly separable as a straight line (linear classifier) can separate the samples correctly into two classes.

How to find the mapping function?

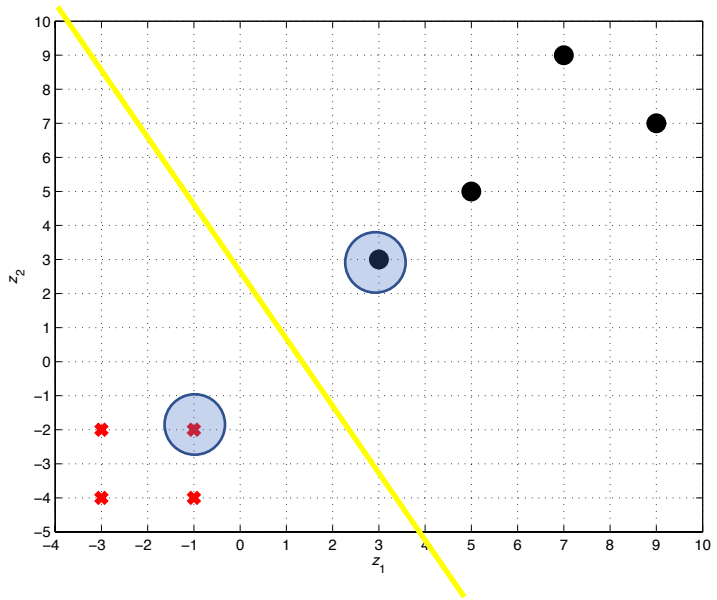
Q3. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}.$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

c. Identify the support vectors by inspection.

By inspection, the support vectors are $\mathbf{z}_1 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$ and $\mathbf{z}_5 = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$



Q3. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}. \quad \leftarrow \text{Class label: +1}$$

$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}. \quad \leftarrow \text{Class label: –1}$$

d. Design a nonlinear SVM classifier to classify all given points. What is the margin? Which data points can be removed without changing the size of the margin?

Define the label for Class 1 as $+1$ and Class 2 as -1 so we have $y_1 = y_2 = y_3 = y_4 = 1$ and $y_5 = y_6 = y_7 = y_8 = -1$.

The hyperplane is $\mathbf{w}^T\mathbf{z} + w_0 = \mathbf{w}^T\Phi(\mathbf{x}) + w_0 = 0$ where $\mathbf{w} = \lambda_1 y_1 \mathbf{z}_1 + \lambda_5 y_5 \mathbf{z}_5 = \lambda_1 y_1 \Phi(\mathbf{x}_1) + \lambda_5 y_5 \Phi(\mathbf{x}_5) = \lambda_1 \begin{bmatrix} 3 \\ 3 \end{bmatrix} - \lambda_5 \begin{bmatrix} -1 \\ -2 \end{bmatrix}$. Recalling that $y_i(\mathbf{w}^T\mathbf{z} + w_0) = 1$ when $\mathbf{z}$ is a support vector.

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$

$$\lambda_i (y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1) = 0, \quad i = 1, 2, \ldots, N$$

For $\mathbf{z} = \mathbf{z}_1$, $\left(\lambda_1 \begin{bmatrix} 3 \\ 3 \end{bmatrix} - \lambda_5 \begin{bmatrix} -1 \\ -2 \end{bmatrix}\right)^T \begin{bmatrix} 3 \\ 3 \end{bmatrix} + w_0 = 1$

$\Rightarrow 18\lambda_1 + 9\lambda_5 + w_0 = 1.$

For $\mathbf{z} = \mathbf{z}_5$, $-1 \times \left(\lambda_1 \begin{bmatrix} 3 \\ 3 \end{bmatrix} - \lambda_5 \begin{bmatrix} -1 \\ -2 \end{bmatrix}\right)^T \begin{bmatrix} -1 \\ -2 \end{bmatrix} + w_0 = 1$

$\Rightarrow -9\lambda_1 - 5\lambda_5 + w_0 = -1.$

Q3. An SVM classifier is employed to classify the following points:

$$\text{Class 1: } \mathbf{x}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \mathbf{x}_4 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}.$$
$$\text{Class 2: } \mathbf{x}_5 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{x}_6 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{x}_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \mathbf{x}_8 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

d. Design a nonlinear SVM classifier to classify all given points.

Combining with the condition $\sum_{i=1}^{8} \lambda_i y_i = 0 \Rightarrow \lambda_1 - \lambda_5 = 0$, we have

$$\begin{bmatrix} 18 & 9 & 1 \\ -9 & -5 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_5 \\ w_0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}.$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{0} \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$
$$\frac{\partial \mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \Rightarrow \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$
$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, N$$
$$\lambda_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \right) = 0, \quad i = 1, 2, \ldots, N$$

It gives $\lambda_1 = \lambda_5 = 0.0488$ and $w_0 = -0.3171$.

As a result, $\mathbf{w} = 0.0488 \begin{bmatrix} 3 \\ 3 \end{bmatrix} - 0.0488 \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 0.1952 \\ 0.2440 \end{bmatrix}$

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{z}_i = \lambda_1 y_1 \Phi(\mathbf{x}_1) + \lambda_5 y_5 \Phi(\mathbf{x}_5)$$

Hyperplane (in new feature space $\mathbf{z}$):
$0.1952 z_1 + 0.2440 z_2 - 0.3171 = 0.$

The hyperplane (in original feature space $\mathbf{x}$):

$$\mathbf{w}^T \Phi(\mathbf{x}) + w_0 = (\lambda_1 y_1 \Phi(\mathbf{x}_1) + \lambda_5 y_5 \Phi(\mathbf{x}_5))^T \Phi(\mathbf{x}) + w_0 = \begin{bmatrix} 0.1952 \\ 0.2440 \end{bmatrix}^T \Phi(\mathbf{x}) - 0.3171 = 0.$$

$\lambda_1 y_1 \Phi(\mathbf{x}_1) \Phi(\mathbf{x})$ $\quad$ $\lambda_5 y_5 \Phi(\mathbf{x}_5) \Phi(\mathbf{x})$

The hyperplane in the original feature space in kernel form:
$0.0488 K(\mathbf{x}_1, \mathbf{x}) - 0.0488 K(\mathbf{x}_5, \mathbf{x}) - 0.3171 = 0$ where $K(\mathbf{p}, \mathbf{x}) = \Phi(\mathbf{p})^T \Phi(\mathbf{x})$

Q4. Compare and discuss the four multi-class approaches (namely one against one, one against all, binary decision tree and binary coded approach) their advantages and disadvantages.

	One against one	One against all	Binary decision tree	Binary code
Number of SVMs required	$R(R-1)/2$	R	$R-1$	$\lceil \log_2 R \rceil$
Number of SVMs consulted	All	All	$\lceil \log_2 R \rceil$	All

One against one approach:

Advantages:

- Training is quicker for each SVM as the training dataset is smaller compared with the whole dataset.
- It is flexibly to add/remove classes. For adding classes, existing SVMs are not needed to be re-trained and only SVMs for new classes are needed to be trained. For removing classes, the SVMs of the corresponding classes can be simply removed.

Disadvantages:

- The number of SVMs is larger compared with other approaches.
- More SVMs have to be evaluated to make the final decision.
- Some domains in the feature space cannot be determined (can be alleviated by employing a soft SVM classifiers).

Q4. Compare and discuss the four multi-class approaches (namely one against one, one against all, binary decision tree and binary coded approach) their advantages and disadvantages.

One against all approach:

Advantages:

- Number of SVMs to be trained is relatively less.
- Number of SVMs to be evaluated making a decision is relatively less.

Disadvantages:

- Training time may be longer as the whole dataset is used for training for each SVM.
- When adding/removing classes, all SVMs have to be trained.
- Some domains in the feature space cannot be determined (can be alleviated by employing a real value SVM classifiers).

Q4. Compare and discuss the four multi-class approaches (namely one against one, one against all, binary decision tree and binary coded approach) their advantages and disadvantages.

Binary decision tree approach:

Advantages:

- Number of SVMs to be trained is relatively less.
- Number of SVMs to be evaluated making a decision is further reduced.
- When adding classes, only SVMs for new classes are needed to be trained. The existing SVMs can be re-used again.
- Training time is a bit quicker as not all SVM uses the whole training dataset. The lower is the level in the decision tree, the smaller the training set is required for the training of SVMs.

Disadvantages:

- When removing classes, all SVMs are needed to be re-trained.
- The classification performance depends heavily on the SVMs in the upper level. Classification error will propagate.

Q4. Compare and discuss the four multi-class approaches (namely one against one, one against all, binary decision tree and binary coded approach) their advantages and disadvantages.

Binary coded approach:

Advantages:

- Number of SVMs to be trained is further reduced.
- Number of SVMs to be evaluated making a decision is further reduced.

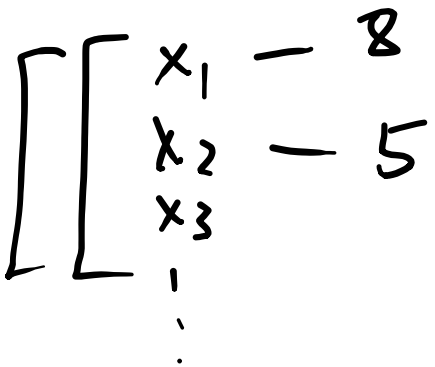
Disadvantages:

- Training time may be longer as the whole dataset is used for training for each SVM.
- When adding/removing classes, all SVMs have to be re-trained.
- It may lead to undefined class when the number of classes is not exactly $2^{\text{Number of SVMs}}$.

Q5. Given the class assignments in Table 1 for a multi-class SVM classifier using *binary coded approach*, list the binary codes for all classes in a table. What is the main disadvantage?

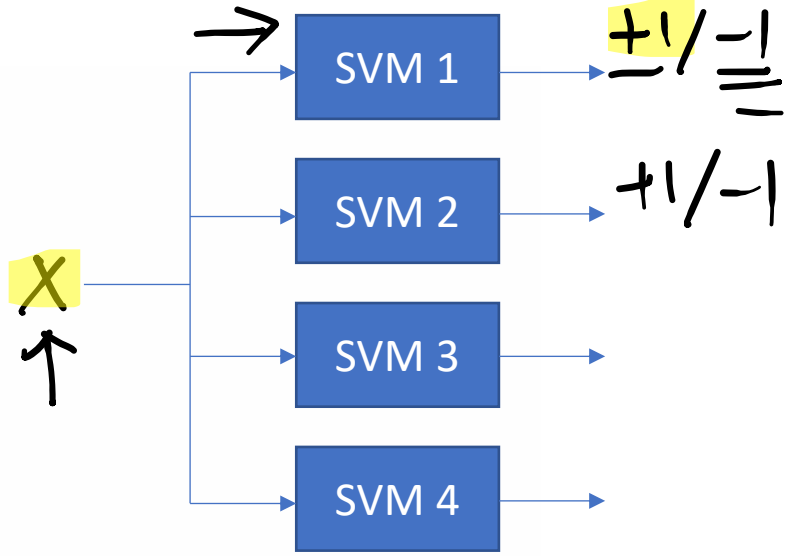
SVM Number	Class assignment $(+1 \mid -1)$
SVM 1	12345|6789
SVM 2	12567|3489
SVM 3	14578|2369
SVM 4	1368|24579

Table 1: Class assignments.



Code assignments:

Class	SVM 1	SVM 2	SVM 3	SVM 4
1	+1	+1	+1	+1
2	+1	+1	−1	−1
3	+1	−1	−1	+1
4	+1	−1	+1	−1
5	+1	+1	+1	−1
6	−1	+1	−1	+1
7	−1	+1	+1	−1
8	−1	−1	+1	+1
9	−1	−1	−1	−1



The main disadvantage is that the classifier has undefined classes, i.e., classes 10-16.

Pattern Recognition, Neural Networks and Deep Learning (7CCSMPNN)

Tutorial 9: Solutions

Q1. Consider the following dataset: $\{(\mathbf{x} = (1, 0), y = +1), (\mathbf{x} = (-1, 0), y = +1), (\mathbf{x} = (0, 1), y = -1), (\mathbf{x} = (0, -1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_7(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_8(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

H1 to h8 is 8 weak classifiers.

a. Plot the dataset in 2-dimensional space. Is the dataset linearly separable?

b. Determine the classification error for all weak classifiers.

c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.

Q1. Consider the following dataset: $\{(\mathbf{x} = (1, 0), y = +1), (\mathbf{x} = (-1, 0), y = +1), (\mathbf{x} = (0, 1), y = -1), (\mathbf{x} = (0, -1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

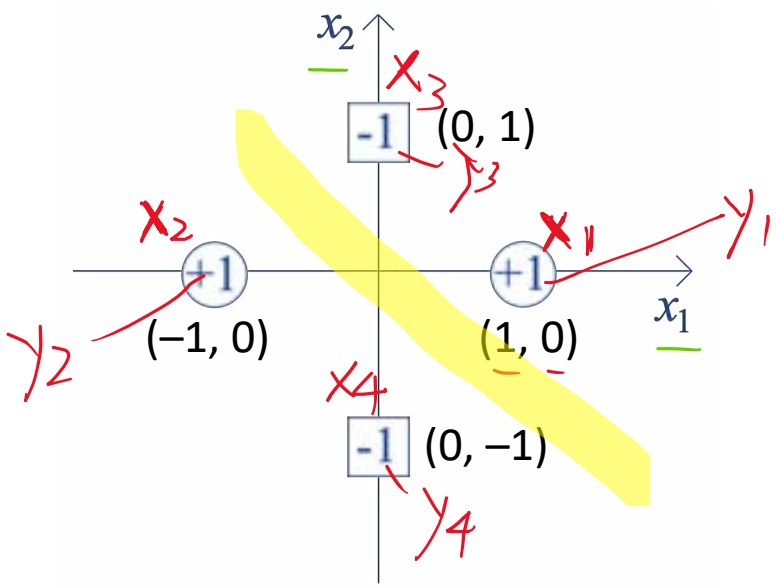
$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_7(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_8(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

a. Plot the dataset in 2-dimensional space. Is the dataset linearly separable?



It is not linearly separable as the two classes cannot be separated by a linear line.

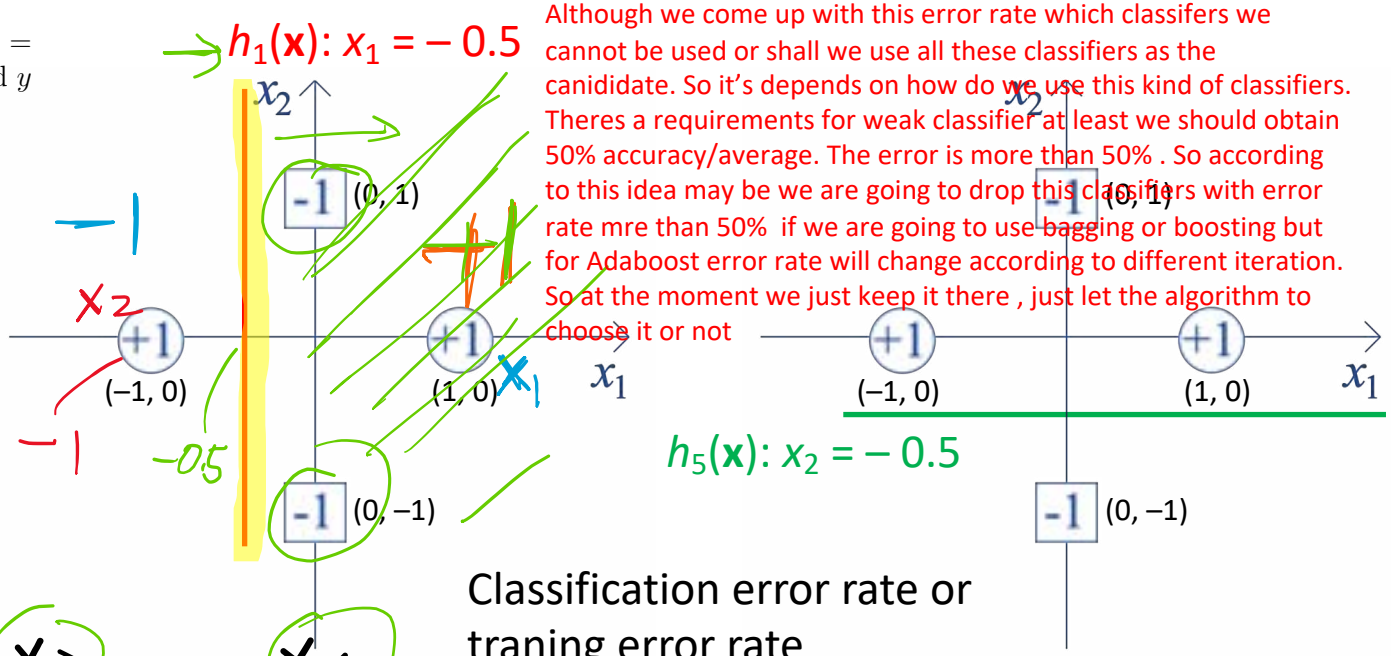
Q1. Consider the following dataset: $\{(\mathbf{x} = (1,0), y = +1), (\mathbf{x} = (-1,0), y = +1), (\mathbf{x} = (0,1), y = -1), (\mathbf{x} = (0,-1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_7(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > 0.5 \\ -1, & \text{otherwise} \end{cases} ; \quad h_8(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$



Although we come up with this error rate which classifers we cannot be used or shall we use all these classifiers as the canididate. So it's depends on how do we use this kind of classifiers. Theres a requirements for weak classifier at least we should obtain 50% accuracy/average. The error is more than 50% . So according to this idea may be we are going to drop this classifiers with error rate mre than 50% if we are going to use bagging or boosting but for Adaboost error rate will change according to different iteration. So at the moment we just keep it there , just let the algorithm to choose it or not

b. Determine the classification error for all weak classifiers.

Classification error rate or traning error rate

Classifier	$(1,0), +1$	$(-1,0), +1$	$(0,1), -1$	$(0,-1), -1$	Training Error
$h_1(\mathbf{x})$	$+1$ ✗	-1 ✗	$+1$ ✗	$+1$ ✗	0.75
$h_2(\mathbf{x})$	-1 ✗	$+1$ ✓	-1 ✓	-1 ✓	0.25
$h_3(\mathbf{x})$	$+1$	-1	-1	-1	0.25
$h_4(\mathbf{x})$	-1	$+1$	$+1$	$+1$	0.75
$h_5(\mathbf{x})$					
$h_6(\mathbf{x})$	-1	-1	-1	$+1$	0.75
$h_7(\mathbf{x})$	-1	-1	$+1$	-1	0.75
$h_8(\mathbf{x})$	$+1$	$+1$	-1	$+1$	0.25

Training error is calculated by total x points. In order to tell which points its belong to horzintal line is x1 axis and vertical is x2 so if the classifier is x1 than it;'s vertical line classifier if classier is x2 then is horizontal. And threshold is 0.5 is which middle. If the point is beyond that line is positive otherwise negative.

Discussion: Which classifiers cannot be used? Can we keep them there? How to design these classifiers? Shall we design them beforehand?

So we have 8 canidate so we have 8 kmax

Q1. Consider the following dataset: $\{(\mathbf{x} = (1, 0), y = +1), (\mathbf{x} = (-1, 0), y = +1), (\mathbf{x} = (0, 1), y = -1), (\mathbf{x} = (0, -1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

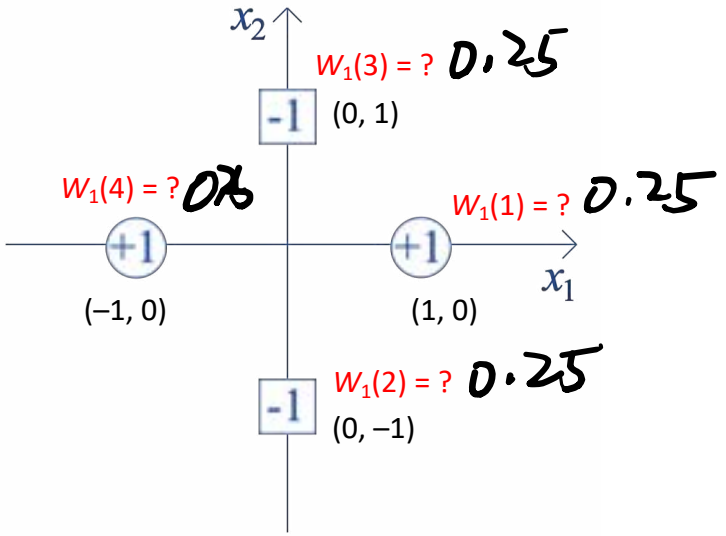
$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_7(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_8(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.



Algorithm: Adaboost

Given $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i = \{-1, 1\}, i = 1, \ldots, n$ ← How many samples, n = ? 4

begin initialise k_{max}, $W_1(i) = 1/n, i = 1, \ldots, n$ ← $W_1(i)$ = ? for i = 1, 2, 3, 4

$k \leftarrow 0$

do $k \leftarrow k + 1$

1. $\hat{h}_k = \arg\min_{h_j \in \mathscr{H}} E_j$ where $E_j = \sum_{i=1, y_i \neq h_j(\mathbf{x}_i)}^{n} W_k(i)$ (Weighted error rate)†
2. ε_k = overall weighted error rate of classifier $\hat{h}_k$ (i.e., the minimum E_j)

   if $\varepsilon_k > 0.5$

   $k_{max} = k - 1$

   Stop

   end
3. $\alpha_k = \frac{1}{2} \ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$
4. $W_{k+1}(i) = \frac{W_k(i) e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}}{Z_k}, i = 1, \ldots, n$ (Update $W_k(i)$)‡

until $k = k_{max}$

end

† Find the classifier $h_k \in \mathscr{H}$ that minimises the error ε_k with respect to the distribution $W_k(i)$.

‡ Z_k is a normalization factor chosen so that $W_{k+1}(i)$ is a normalised distribution.

k_{max} = ? → 8

$\frac{1}{4} = 0.25$

Kmax is maximum no of classifiers/weak and k <- 0 is a counter and it's go to k+1 = 1 so we are in the first iteration. We go to step 1 to 4 and then go back to top, increase the counter in order to determine the classifier/in order to determine how do we combine these classifier using alphaK

Initially the error rate we assign to each sample 1/n or ¼ which is 0.25 or weighted error. If you check w_1(i) part

Q1. Consider the following dataset: $\{(\mathbf{x} = (1,0), y = +1), (\mathbf{x} = (-1,0), y = +1), (\mathbf{x} = (0,1), y = -1), (\mathbf{x} = (0,-1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_7(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_8(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

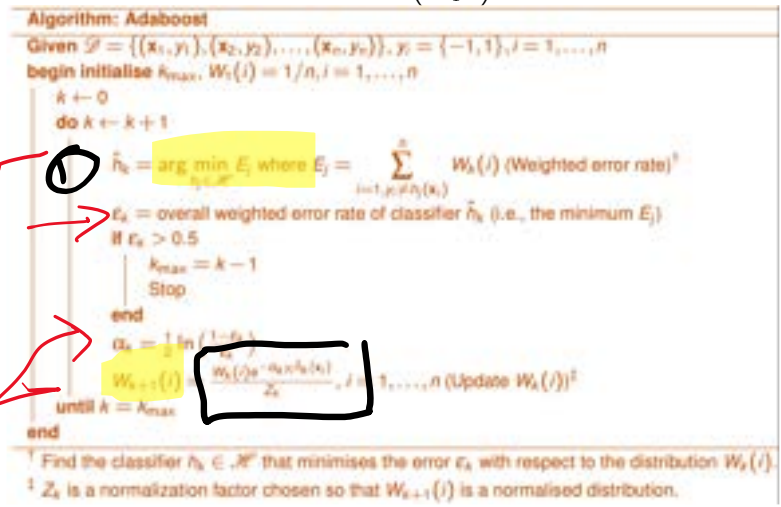
c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.

i	Class	$W_k(i)$
1	(1,0), +1	0.25
2	(−1,0), +1	0.25
3	(0,1), −1	0.25
4	(0,−1), −1	0.25

Classifier	(1,0), +1	(−1,0), +1	(0,1), −1	(0,−1), −1	Training Error
$h_1(\mathbf{x})$	+1	−1	+1	+1	0.75
$h_2(\mathbf{x})$	−1	+1	−1	−1	0.25
$h_3(\mathbf{x})$	+1	−1	−1	−1	0.25
$h_4(\mathbf{x})$	−1	+1	+1	+1	0.75
$h_5(\mathbf{x})$	+1	+1	+1	−1	0.25
$h_6(\mathbf{x})$	−1	−1	−1	+1	0.75
$h_7(\mathbf{x})$	−1	−1	+1	−1	0.75
$h_8(\mathbf{x})$	+1	+1	−1	+1	0.25

Table 1: Training error of 8 classifiers.

- **Initialise parameters:** Choose $k_{\max} = 8$; $n = 4$; $W_1(i) = 0.25$; $i = 1, 2, 3, 4$.
- **Compute the training error:** Classifier $h_i(\mathbf{x})$: $E_i = 0.75$, $i = 1, 4, 6, 7$.
- **Pick the best classifier:** As classifiers $h_2(\mathbf{x})$, $h_3(\mathbf{x})$ offer the same lowest training error, we randomly pick say, $\hat{h}_1(\mathbf{x}) = h_2(\mathbf{x})$.
- **Determine ε_1:** Choose $\varepsilon_1 = E_2 = 0.25$.
- **Determine α_1:** $\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) = \frac{1}{2}\ln\left(\frac{1-0.25}{0.25}\right) = 0.54$



- **Initialise parameters:** Choose $k_{\max} = 8$; $n = 4$; $W_1(i) = 0.25$; $i = 1, 2, 3, 4$.

this is round 1 which is k = 1
we are this point.

Round 1:

1 • **Compute the training error:** Classifier $h_i(\mathbf{x})$: $E_1 = 0.75$; $E_2 = 0.25$; $E_3 = 0.75$; $E_4 = 0.75$; $E_5 = 0.25$; $E_6 = 0.75$; $E_7 = 0.75$; $E_8 = 0.25$

It's a notation to indicate it's a classifier (h1hat) for the iteration 1

2 • **Pick the best classifier:** $\hat{h}_1(\mathbf{x}) = h_2(x)$

We choose the lowest error

3 • **Determine ε_1:** $\varepsilon_1 = E_2 = 0.25$

- **Determine α_1:** $\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) = 0.5493.$

this step it means that it's a weight asoociated to first classifier h1hat

- **Initialise parameters:** Choose $k_{\max} = 8$; $n = 4$; $W_1(i) = 1/n = 0.25$, $i = 1, 2, 3, 4$.

- **Compute the training error:** Classifier $h_i(\mathbf{x})$: $E_i = 0.25$, $i = 2, 3, 5, 8$. Classifier $h_i(\mathbf{x})$: $E_i = 0.75$, $i = 1, 4, 6, 7$.

- **Pick the best classifier:** As classifiers $h_2(\mathbf{x})$, $h_3(\mathbf{x})$, $h_5(\mathbf{x})$ and $h_8(\mathbf{x})$ offer the same lowest training error, we randomly pick one in this round, say, $\hat{h}_1(\mathbf{x}) = h_2(\mathbf{x})$.

- **Determine ε_1:** Choose $\varepsilon_1 = E_2 = 0.25$.
- **Determine α_1:** $\alpha_1 = \frac{1}{2} \ln \left(\frac{1-\varepsilon_1}{\varepsilon_1} \right) = \frac{1}{2} \ln \left(\frac{1-0.25}{0.25} \right) = 0.5493$.

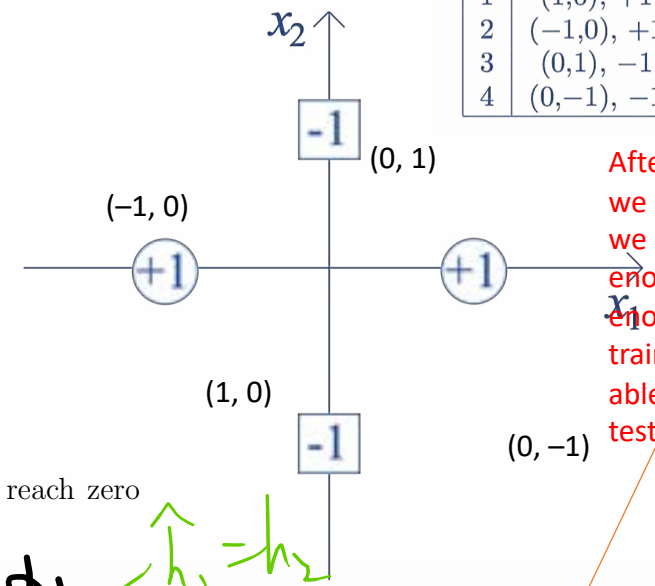
Q1. Consider the following dataset: $\{(\mathbf{x}=(1,0), y=+1), (\mathbf{x}=(-1,0), y=+1), (\mathbf{x}=(0,1), y=-1), (\mathbf{x}=(0,-1), y=-1)\}$ where $\mathbf{x}=(x_1,x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

$$h_1(\mathbf{x})=\begin{cases}+1, & \text{if } x_1>-0.5\\ -1, & \text{otherwise}\end{cases} \quad ; \quad h_2(\mathbf{x})=\begin{cases}-1, & \text{if } x_1>-0.5\\ +1, & \text{otherwise}\end{cases}$$

$$h_3(\mathbf{x})=\begin{cases}+1, & \text{if } x_1>0.5\\ -1, & \text{otherwise}\end{cases} \quad ; \quad h_4(\mathbf{x})=\begin{cases}-1, & \text{if } x_1>0.5\\ +1, & \text{otherwise}\end{cases}$$

$$h_5(\mathbf{x})=\begin{cases}+1, & \text{if } x_2>-0.5\\ -1, & \text{otherwise}\end{cases} \quad ; \quad h_6(\mathbf{x})=\begin{cases}-1, & \text{if } x_2>-0.5\\ +1, & \text{otherwise}\end{cases}$$

$$h_7(\mathbf{x})=\begin{cases}+1, & \text{if } x_2>0.5\\ -1, & \text{otherwise}\end{cases} \quad ; \quad h_8(\mathbf{x})=\begin{cases}-1, & \text{if } x_2>0.5\\ +1, & \text{otherwise}\end{cases}$$



c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.

We are at round 1: $k = 1$

- **Compute $W_2(i)$ for next round:**

i	Class	$W_k(i)$	$W_k(i)e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}$	$W_{k+1}(i)=W_k(i)e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}/Z_k$
1	(1,0), +1	0.25	$0.25e^{0.5493}=0.4330$	$0.4330/0.8659=0.5001$
2	(−1,0), +1	0.25	$0.25e^{-0.5493}=0.1443$	$0.1443/0.8659=0.1666$
3	(0,1), −1	0.25	$0.25e^{-0.5493}=0.1443$	$0.1443/0.8659=0.1666$
4	(0,−1), −1	0.25	$0.25e^{-0.5493}=0.1443$	$0.1443/0.8659=0.1666$

4

i column is the sample

When i is 1 it's correspond to first sample . It's a weighted error column

Alpha value found in previous slide

- $Z_1=\sum_{i=1}^{4} W_1(i)e^{-\alpha_1 y_i \hat{h}_1(\mathbf{x}_i)}=0.4330+0.1443\times 3=0.8659$

Zk is the normalization factor and it's a sum of all W-K column

Wki is the way to initialize error rate at the first place since here it's got 4 sample it's 1/4

After all these calculations, before we go to second iteration/round , we have to ask ourself whether we are going to stop or not. One thing we can determine that we ask ourself that is, is this classifier good enough? We have to determine what it means by good enough, good enough that is say one possibility or one criterion which give zero training error, it means that we are able to have a classifier which is able to correctly classify all the training examples because we do have a test sample so this is one thing we can use

- **Compute $W_2(i)$ for next round:**

i	Class	$W_k(i)$	$W_k(i)e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}$	$W_{k+1}(i)=W_k(i)e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}/Z_k$
1	(1,0), +1	0.25	$0.25e^{0.5493}=0.4330$	$0.4330/0.8659=0.5001$
2	(−1,0), +1	0.25	$0.25e^{-0.5493}=0.1443$	$0.1443/0.8659=0.1666$
3	(0,1), −1	0.25	$0.25e^{-0.5493}=0.1443$	$0.1443/0.8659=0.1666$
4	(0,−1), −1	0.25	$0.25e^{-0.5493}=0.1443$	$0.1443/0.8659=0.1666$

Q1. Consider the following dataset: $\{(\mathbf{x} = (1, 0), y = +1), (\mathbf{x} = (-1, 0), y = +1), (\mathbf{x} = (0, 1), y = -1), (\mathbf{x} = (0, -1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

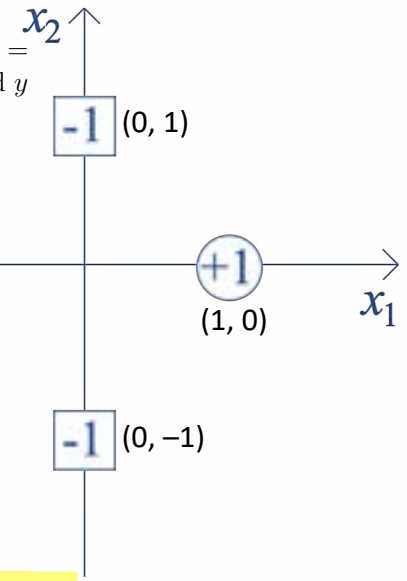
No we have to determine whether this classifier is good enough so after the first iteration we come up with this adaboost classifier.

$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_7(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_8(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$



After all these calculations, before we go to second iteration/round , we have to ask ourself whether we are going to stop or not. One thing we can determine that we ask ourself that is, is this classifier good enough? We have to determine what it means by good enough, good enough that is say one possibility or one criterion which give zero training error, it means that we are able to have a classifier which is able to correctly classify all the training examples because we do have a test sample so this is one thing we can use

c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.

- **AdaBoost Classifier in this round:** $h(\mathbf{x}) = \alpha_1 h_2(\mathbf{x}) = 0.5493 h_2(\mathbf{x})$
- Hard classifier is used for classification: $\text{sgn}(0.5493 h_2(\mathbf{x}))$.

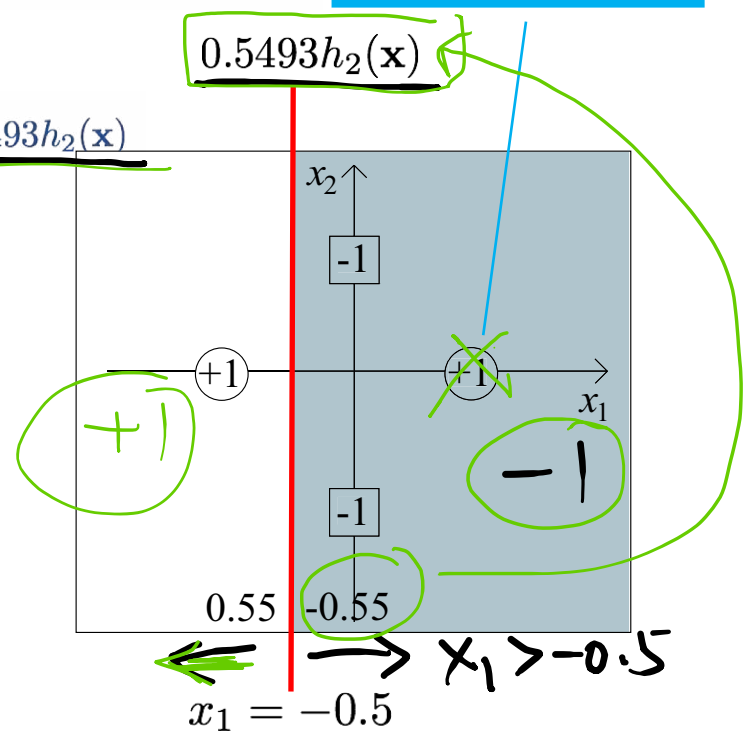
Atm we cannot use for the classification because it will give us -0.55 so we have to turn into hard classifier using the sign function. If you see (1, 0) it's classified on wrong side that means we still cannot reach 0 classification error. That it suggest that we are not going to stop iterations. We go to next step and start from the very beginning and go through again four steps. Check next slide

Does the Adaboost classifier improve the classification accuracy at $k = 1$? Shall we stop the iterations?

Classifier	(1,0), +1	(−1,0), +1	(0,1), −1	(0,−1), −1	Training Error
$h_1(\mathbf{x})$	+1	−1	+1	+1	0.75
$h_2(\mathbf{x})$	−1	+1	−1	−1	0.25
$h_3(\mathbf{x})$	+1	−1	−1	−1	0.25
$h_4(\mathbf{x})$	−1	+1	+1	+1	0.75
$h_5(\mathbf{x})$	+1	+1	+1	−1	0.25
$h_6(\mathbf{x})$	−1	−1	−1	+1	0.75
$h_7(\mathbf{x})$	−1	−1	+1	−1	0.75
$h_8(\mathbf{x})$	+1	+1	−1	+1	0.25

Table 1: Training error of 8 classifiers.

(1, 0) is wrongly classified



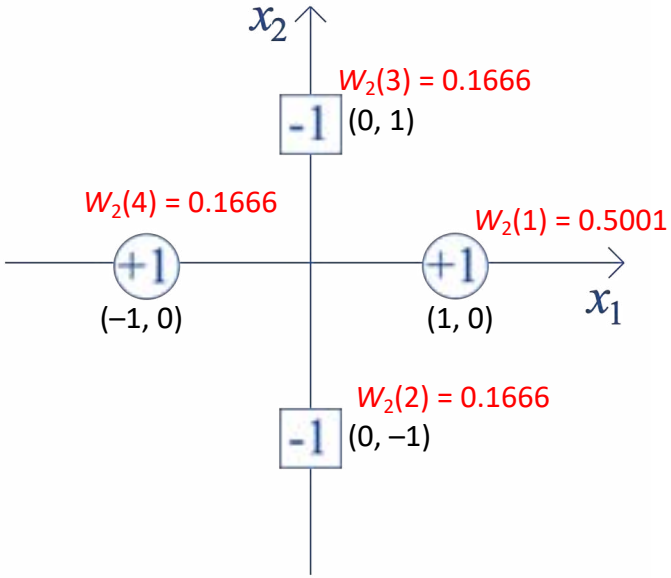
Q1. Consider the following dataset: $\{(\mathbf{x} = (1,0), y = +1), (\mathbf{x} = (-1,0), y = +1), (\mathbf{x} = (0,1), y = -1), (\mathbf{x} = (0,-1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_7(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_8(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$



c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.

Algorithm: Adaboost

Given $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i = \{-1, 1\}, i = 1, \ldots, n$

begin initialise k_{max}, $W_1(i) = 1/n, i = 1, \ldots, n$

$k \leftarrow 0$

do $k \leftarrow k + 1$ ← $k = 2$

$\hat{h}_k = \arg\min_{h_j \in \mathcal{H}} E_j$ where $E_j = \sum_{i=1, y_i \neq h_j(\mathbf{x}_i)}^{n} W_k(i)$ (Weighted error rate)†

$\varepsilon_k =$ overall weighted error rate of classifier $\hat{h}_k$ (i.e., the minimum E_j)

if $\varepsilon_k > 0.5$

$k_{max} = k - 1$

Stop

end

$\alpha_k = \frac{1}{2} \ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$

$W_{k+1}(i) = \frac{W_k(i) e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}}{Z_k}, i = 1, \ldots, n$ (Update $W_k(i)$)‡

until $k = k_{max}$

end

† Find the classifier $h_k \in \mathcal{H}$ that minimises the error ε_k with respect to the distribution $W_k(i)$.

‡ Z_k is a normalization factor chosen so that $W_{k+1}(i)$ is a normalised distribution.

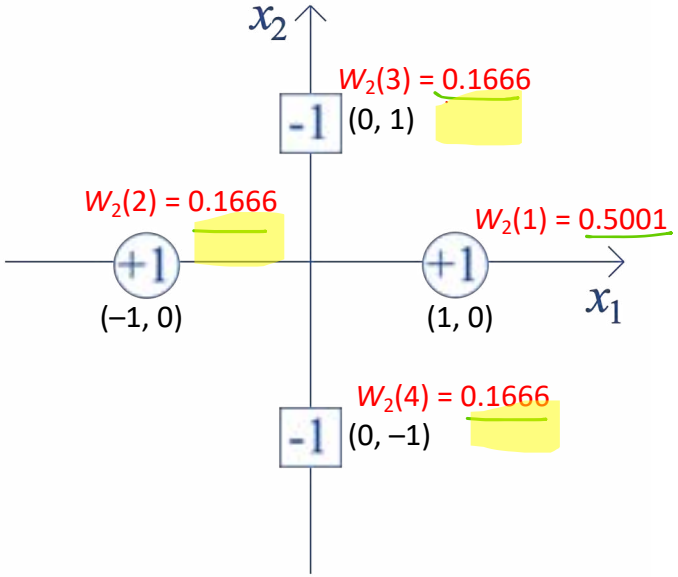
Q1. Consider the following dataset: $\{(\mathbf{x} = (1,0), y = +1), (\mathbf{x} = (-1,0), y = +1), (\mathbf{x} = (0,1), y = -1), (\mathbf{x} = (0,-1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_7(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_8(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$



c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.

Round 2: $k = 2$

Weighted error times depends on classifier errors

- **Compute the training error:**

Classifier	(1,0), +1	(−1,0), +1	(0,1), −1	(0,−1), −1	Weighted Training Error
$h_1(\mathbf{x})$	+1	−1	+1	+1	$0.1666 \times 3 = 0.4998$
$h_2(\mathbf{x})$	−1	+1	−1	−1	0.5001
$h_3(\mathbf{x})$	+1	−1	−1	−1	0.1666
$h_4(\mathbf{x})$	−1	+1	+1	+1	$0.5001 + 0.1666 \times 2 = 0.8333$
$h_5(\mathbf{x})$	+1	+1	+1	−1	0.1666
$h_6(\mathbf{x})$	−1	−1	−1	+1	$0.5001 + 0.1666 \times 2 = 0.8333$
$h_7(\mathbf{x})$	−1	−1	+1	−1	$0.5001 + 0.1666 \times 2 = 0.8333$
$h_8(\mathbf{x})$	+1	+1	−1	+1	0.1666

here it also depends on the which x points(sample) errors and use relative to that. Check previous slides for each sample erros

Pick the best classifier: $\hat{h}_2(\mathbf{x}) =$

Q1. Consider the following dataset: $\{(\mathbf{x} = (1,0), y = +1), (\mathbf{x} = (-1,0), y = +1), (\mathbf{x} = (0,1), y = -1), (\mathbf{x} = (0,-1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

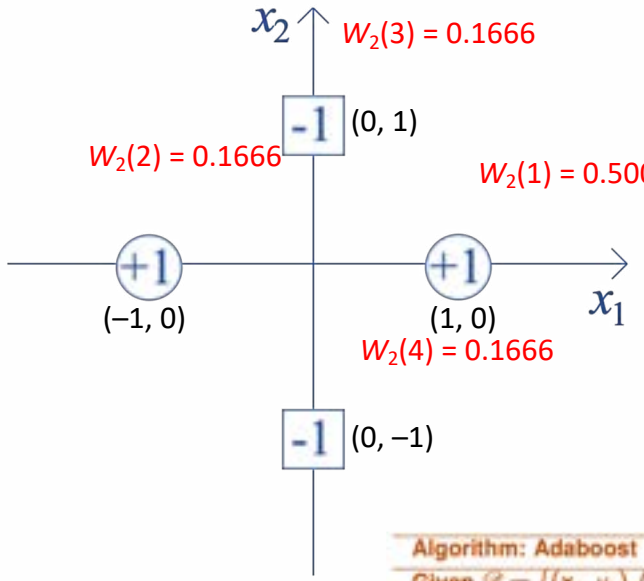
$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_7(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_8(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.



1 • **Pick the best classifier:** $\hat{h}_2(\mathbf{x}) = h_3$

2 • **Determine ε_2:** $\varepsilon_2 = 0.1666$

3 • **Determine α_2:** $\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_2}{\varepsilon_2}\right) = 0.8050$

Classifier	(1,0), +1	(−1,0), +1	(0,1), −1	(0,−1), −1	Weighted Training Error
$h_1(\mathbf{x})$	+1	−1	+1	+1	$0.1666 \times 3 = 0.4998$
$h_2(\mathbf{x})$	−1	+1	−1	−1	0.5001
$h_3(\mathbf{x})$	+1	−1	−1	−1	0.1666
$h_4(\mathbf{x})$	−1	+1	+1	+1	$0.5001 + 0.1666 \times 2 = 0.8333$
$h_5(\mathbf{x})$	+1	+1	+1	−1	0.1666
$h_6(\mathbf{x})$	−1	−1	−1	+1	$0.5001 + 0.1666 \times 2 = 0.8333$
$h_7(\mathbf{x})$	−1	−1	+1	−1	$0.5001 + 0.1666 \times 2 = 0.8333$
$h_8(\mathbf{x})$	+1	+1	−1	+1	0.1666

Algorithm: Adaboost

Given $\mathscr{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i = \{-1, 1\}, i = 1, \ldots, n$

begin initialise k_{max}, $W_1(i) = 1/n, i = 1, \ldots, n$

$k \leftarrow 0$

do $k \leftarrow k + 1$

$\hat{h}_k = \arg\min_{h_j \in \mathscr{H}} E_j$ where $E_j = \sum_{i=1, y_i \neq h_j(\mathbf{x}_i)}^{n} W_k(i)$ (Weighted error rate)†

ε_k = overall weighted error rate of classifier $\hat{h}_k$ (i.e., the minimum E_j)

if $\varepsilon_k > 0.5$

$k_{max} = k - 1$

Stop

end

$\alpha_k = \frac{1}{2}\ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$

$W_{k+1}(i) = \frac{W_k(i) e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}}{Z_k}, i = 1, \ldots, n$ (Update $W_k(i)$)‡

until $k = k_{max}$

end

† Find the classifier $h_k \in \mathscr{H}$ that minimises the error ε_k with respect to the distribution $W_k(i)$.

‡ Z_k is a normalization factor chosen so that $W_{k+1}(i)$ is a normalised distribution.

- **Pick the best classifier:** As classifiers $h_3(\mathbf{x})$, $h_5(\mathbf{x})$ and $h_8(\mathbf{x})$ offer the same lowest training error, we randomly pick one in this round, say, $\hat{h}_2(\mathbf{x}) = h_3(\mathbf{x})$.

- **Determine ε_2:** Choose $\varepsilon_2 = E_3 = 0.1666$.
- **Determine α_2:** $\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_2}{\varepsilon_2}\right) = \frac{1}{2}\ln\left(\frac{1-0.1666}{0.1666}\right) = 0.8050$.

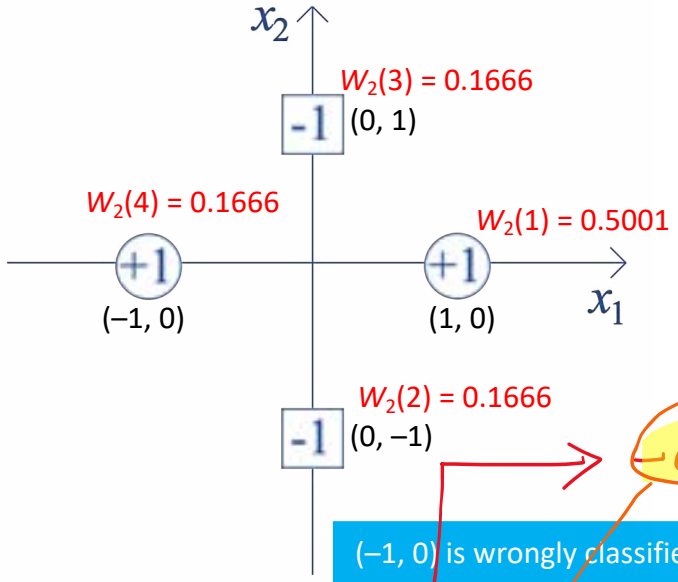
Q1. Consider the following dataset: $\{(\mathbf{x} = (1,0), y = +1), (\mathbf{x} = (-1,0), y = +1), (\mathbf{x} = (0,1), y = -1), (\mathbf{x} = (0,-1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_7(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_8(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$



c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.

4

- **Compute $W_2(i)$ for next round:**

$\hat{h}_2(\mathbf{x}) = h_3(\mathbf{x})$

i	Class	$W_k(i)$	$W_k(i)e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}$	$W_{k+1}(i) = W_k(i)e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}/Z_k$
1	(1,0), +1	0.5001	$0.5001e^{-0.8050} = 0.2236$	$0.2236/0.7452 = 0.3000$
2	(−1,0), +1	0.1666	$0.1666e^{0.8050} = 0.3726$	$0.3726/0.7452 = 0.5000$
3	(0,1), −1	0.1666	$0.1666e^{-0.8050} = 0.0745$	$0.0745/0.7452 = 0.1000$
4	(0,−1), −1	0.1666	$0.1666e^{-0.8050} = 0.0745$	$0.0745/0.7452 = 0.1000$

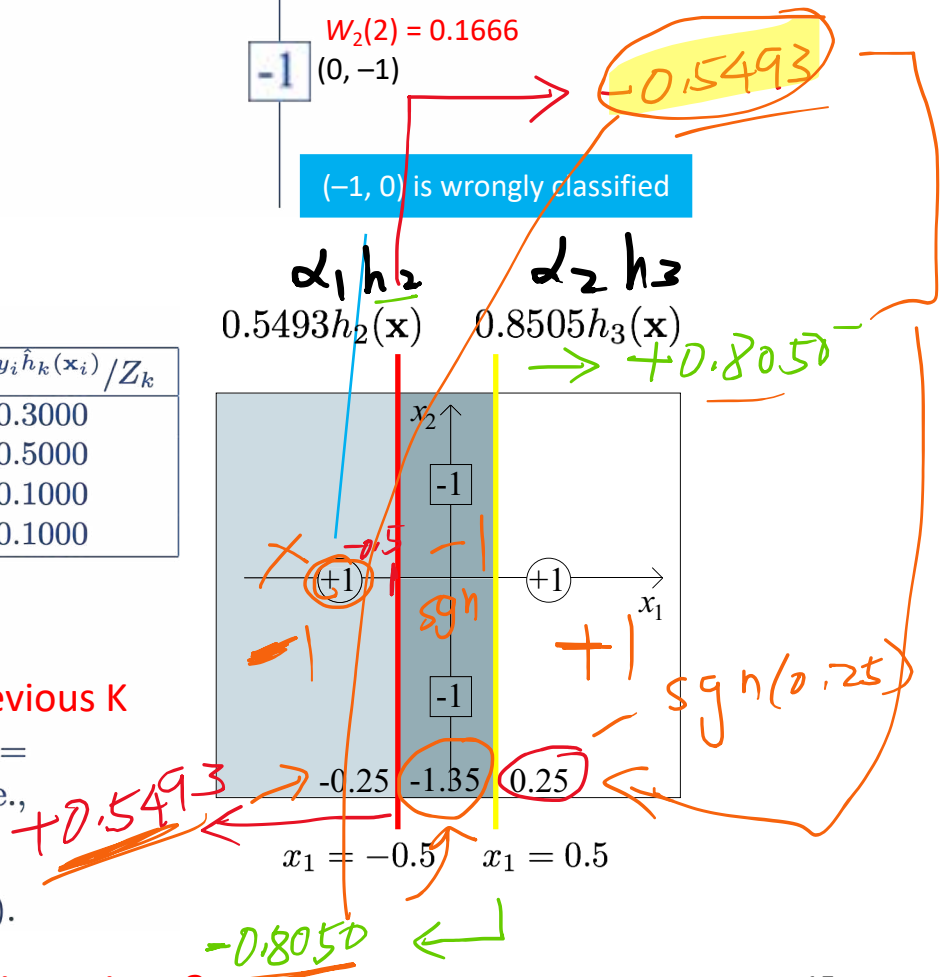
Wk(i) is a weighted error from previous step

- $Z_2 = \sum_{i=1}^{4} W_1(i)e^{-\alpha_2 y_i \hat{h}_2(\mathbf{x}_i)} = 0.2236 + 0.3726 + 0.0745 \times 2 = 0.7452$

Finding ln(logarithm) is alpha and alpha 1 is from the previous K

- **AdaBoost Classifier in this round:** $h(\mathbf{x}) = \alpha_1 h_2(\mathbf{x}) + \alpha_2 h_3(\mathbf{x}) = 0.5493h_2(\mathbf{x}) + 0.8050h_3(\mathbf{x})$ gives $\frac{1}{4} = 0.25$ (unweighed) training error, i.e., 1 out of 4 is wrongly classified.

- Hard classifier is used for classification: $\text{sgn}(0.5493h_2(\mathbf{x}) + 0.8050h_3(\mathbf{x}))$.



Is the Adaboost classifier good enough? Shall we stop the iterations?

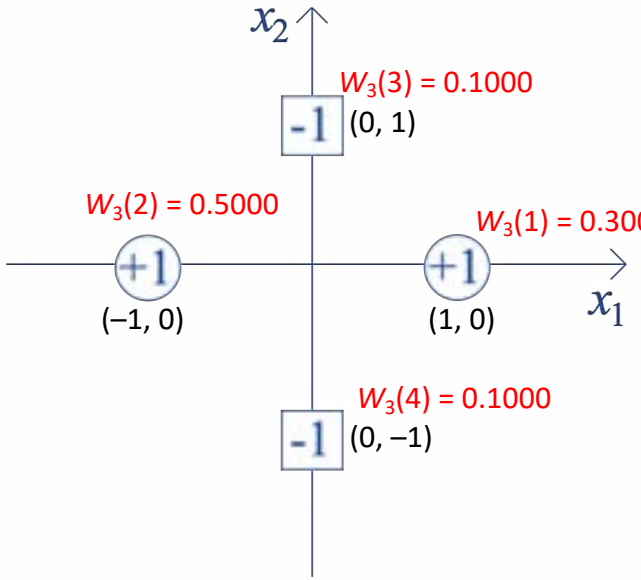
Q1. Consider the following dataset: $\{(\mathbf{x} = (1,0), y = +1), (\mathbf{x} = (-1,0), y = +1), (\mathbf{x} = (0,1), y = -1), (\mathbf{x} = (0,-1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

$$h_7(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_8(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$



This errors are from previous k=2

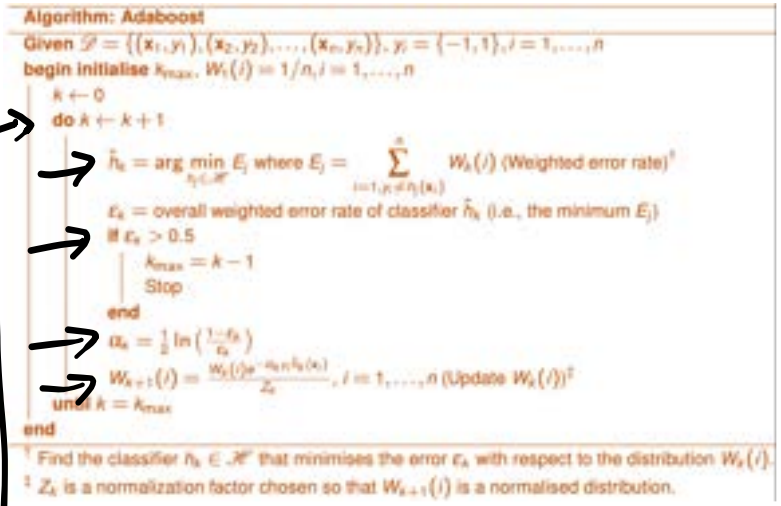
Updated error rate

c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.

Round 3: $k = 3$

- **Compute the training error:**

Classifier	(1,0), +1	(−1,0), +1	(0,1), −1	(0,−1), −1	Weighted Training Error
$h_1(\mathbf{x})$	+1	−1	+1	+1	0.5000 + 0.1000 × 2 = 0.7000
$h_2(\mathbf{x})$	−1	+1	−1	−1	0.3000
$h_3(\mathbf{x})$	+1	−1	−1	−1	0.5000
$h_4(\mathbf{x})$	−1	+1	+1	+1	0.3000 + 0.1000 × 2 = 0.5000
$h_5(\mathbf{x})$	+1	+1	+1	−1	0.1000
$h_6(\mathbf{x})$	−1	−1	−1	+1	0.3000 + 0.5000 + 0.1000 = 0.9000
$h_7(\mathbf{x})$	−1	−1	+1	−1	0.3000 + +0.5000 + 0.1000 = 0.9000
$h_8(\mathbf{x})$	+1	+1	−1	+1	0.1000



1 • **Pick the best classifier:** As classifiers $h_5(\mathbf{x})$ and $h_8(\mathbf{x})$ offer the same lowest training error, we randomly pick one in this round, say, $\hat{h}_3(\mathbf{x}) = h_5(\mathbf{x})$.

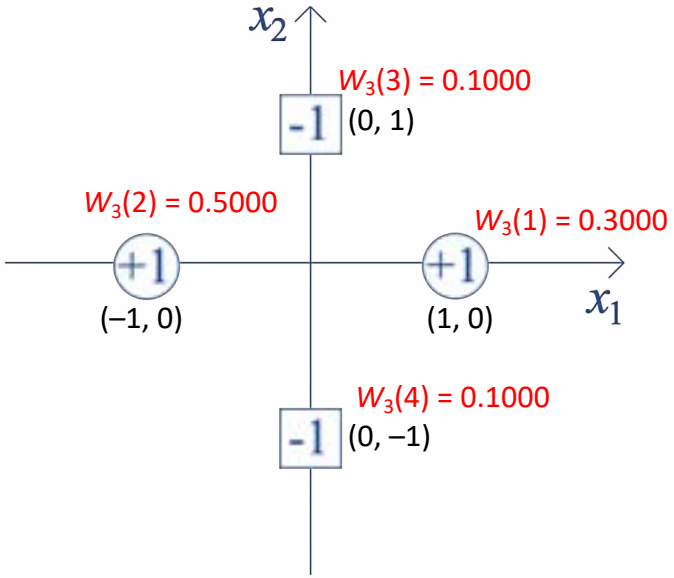
Q1. Consider the following dataset: $\{(\mathbf{x}=(1,0), y=+1), (\mathbf{x}=(-1,0), y=+1), (\mathbf{x}=(0,1), y=-1), (\mathbf{x}=(0,-1), y=-1)\}$ where $\mathbf{x}=(x_1,x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

$$h_1(\mathbf{x})=\begin{cases}+1, & \text{if } x_1>-0.5\\ -1, & \text{otherwise}\end{cases} \quad ; \quad h_2(\mathbf{x})=\begin{cases}-1, & \text{if } x_1>-0.5\\ +1, & \text{otherwise}\end{cases}$$

$$h_3(\mathbf{x})=\begin{cases}+1, & \text{if } x_1>0.5\\ -1, & \text{otherwise}\end{cases} \quad ; \quad h_4(\mathbf{x})=\begin{cases}-1, & \text{if } x_1>0.5\\ +1, & \text{otherwise}\end{cases}$$

$$h_5(\mathbf{x})=\begin{cases}+1, & \text{if } x_2>-0.5\\ -1, & \text{otherwise}\end{cases} \quad ; \quad h_6(\mathbf{x})=\begin{cases}-1, & \text{if } x_2>-0.5\\ +1, & \text{otherwise}\end{cases}$$

$$h_7(\mathbf{x})=\begin{cases}+1, & \text{if } x_2>0.5\\ -1, & \text{otherwise}\end{cases} \quad ; \quad h_8(\mathbf{x})=\begin{cases}-1, & \text{if } x_2>0.5\\ +1, & \text{otherwise}\end{cases}$$



c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.

2 • **Determine ε_3:** Choose $\varepsilon_3 = E_5 = 0.1000$.

3 • **Determine α_3:** $\alpha_3 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_3}{\varepsilon_3}\right) = \frac{1}{2}\ln\left(\frac{1-0.1000}{0.1000}\right) = 1.0986$.

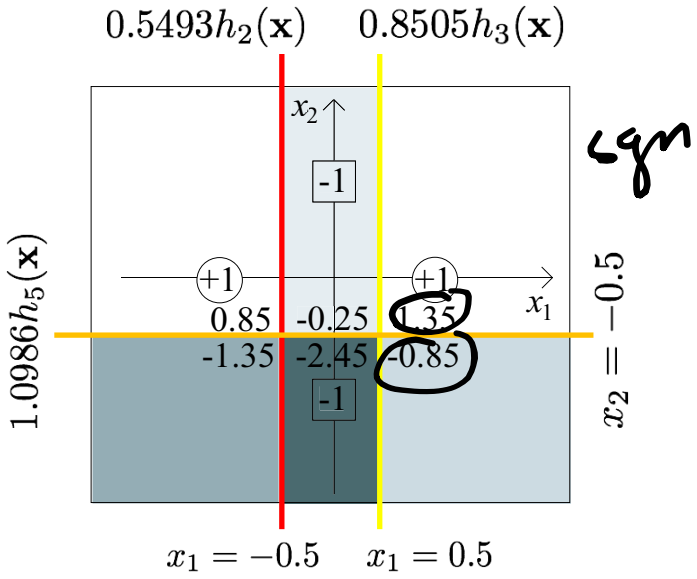
• **Compute $W_3(i)$ for next round:**

i	Class	$W_k(i)$	$W_k(i)e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}$	$W_{k+1}(i) = W_k(i)e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}/Z_k$
1	(1,0), +1	0.3000	$0.3000e^{-1.0986} = 0.1000$	$0.1000/0.6333 = 0.1579$
2	(−1,0), +1	0.5000	$0.5000e^{-1.0986} = 0.1667$	$0.1667/0.6333 = 0.2632$
3	(0,1), −1	0.1000	$0.1000e^{1.0986} = 0.3000$	$0.3000/0.6333 = 0.4737$
4	(0,−1), −1	0.1000	$0.1000e^{-1.0986} = 0.0333$	$0.0333/0.6333 = 0.0526$

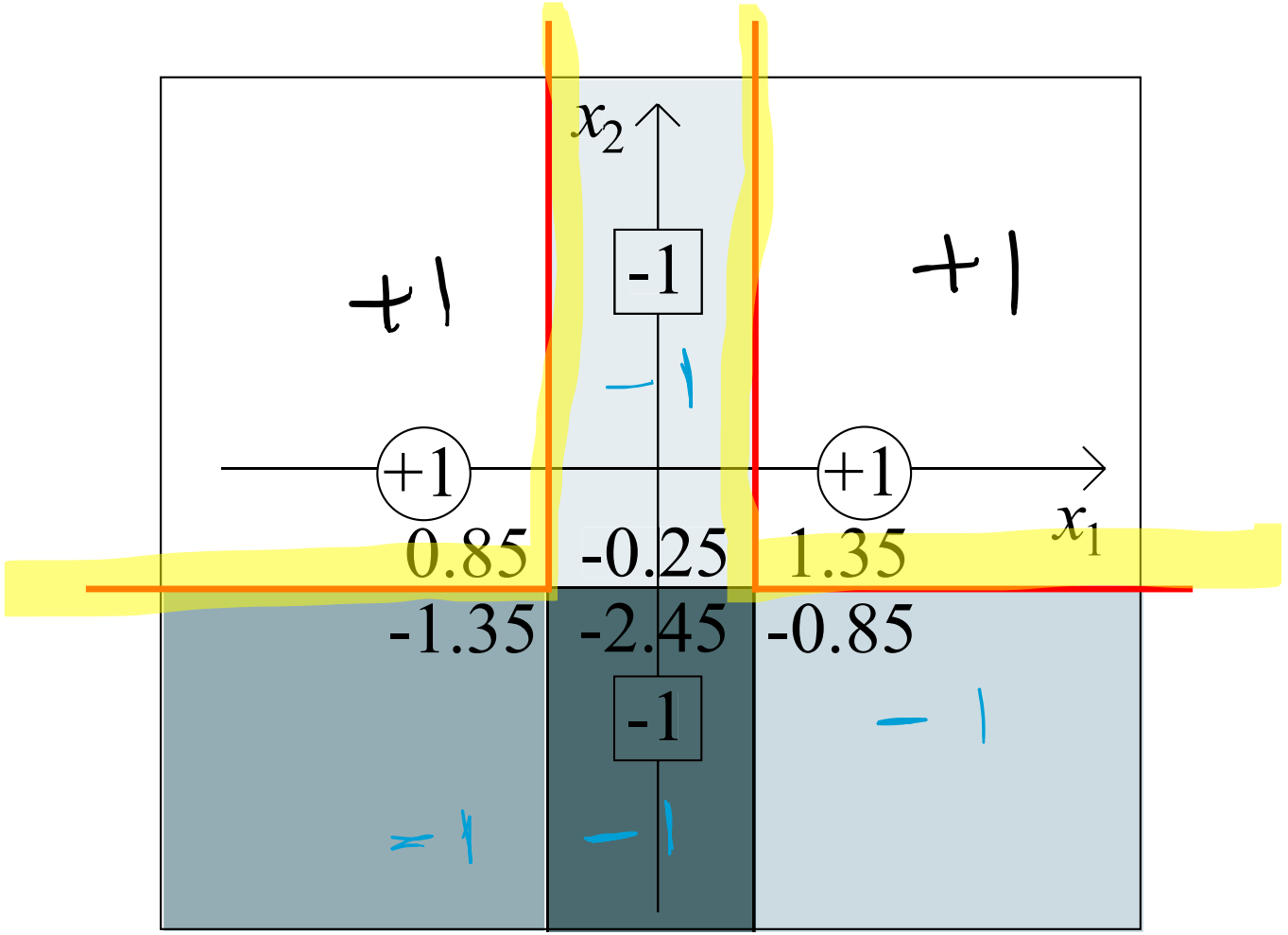
• $Z_3 = \sum_{i=1}^{4} W_2(i)e^{-\alpha_3 y_i \hat{h}_3(\mathbf{x}_i)} = 0.1000 + 0.1667 + 0.3000 + 0.0333 = 0.6333$

• **AdaBoost Classifier in this round:** $h(\mathbf{x}) = \alpha_1 h_2(\mathbf{x}) + \alpha_2 h_3(\mathbf{x}) + \alpha_3 h_5(\mathbf{x}) = 0.5493h_2(\mathbf{x}) + 0.8050h_3(\mathbf{x}) + 1.0986h_5(\mathbf{x})$ gives 0 training error.

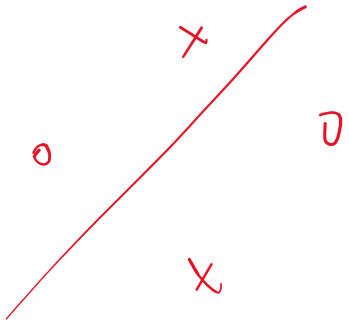
The final hard classifier is: $\text{sgn}\big(0.5493h_2(\mathbf{x}) + 0.8050h_3(\mathbf{x}) + 1.0986h_5(\mathbf{x})\big)$



The final hard classifier is: $\operatorname{sgn}\left(0.5493h_2(\mathbf{x}) + 0.8050h_3(\mathbf{x}) + 1.0986h_5(\mathbf{x})\right)$



How good is this Adaboost classifier?



Initialisation

1

Design/determine a classifier $\hat{h}_k(x)$ with the smallest weighed error weights E_i characterized by $W_k(i)$

2

Determine ε_k

3

Determine α_k

4

Update $W_k(i)$

Is current Adaboost classifier satisfied?

No

Yes

Stop and output the Adaboost classifier

Algorithm: Adaboost

Given $\mathscr{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, $y_i = \{-1, 1\}$, $i = 1, \ldots, n$

begin initialise k_{max}, $W_1(i) = 1/n$, $i = 1, \ldots, n$

$k \leftarrow 0$

do $k \leftarrow k + 1$

1. $\hat{h}_k = \arg\min_{h_j \in \mathscr{H}} E_j$ where $E_j = \sum_{i=1, y_i \neq h_j(x_i)}^{n} W_k(i)$ (Weighted error rate)†
2. ε_k = overall weighted error rate of classifier $\hat{h}_k$ (i.e., the minimum E_j)
   - if $\varepsilon_k > 0.5$
     - $k_{max} = k - 1$
     - Stop
   - end
3. $\alpha_k = \frac{1}{2} \ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$
4. $W_{k+1}(i) = \frac{W_k(i) e^{-\alpha_k y_i \hat{h}_k(x_i)}}{Z_k}$, $i = 1, \ldots, n$ (Update $W_k(i)$)‡

until $k = k_{max}$

end

† Find the classifier $h_k \in \mathscr{H}$ that minimises the error ε_k with respect to the distribution $W_k(i)$.

‡ Z_k is a normalization factor chosen so that $W_{k+1}(i)$ is a normalised distribution.

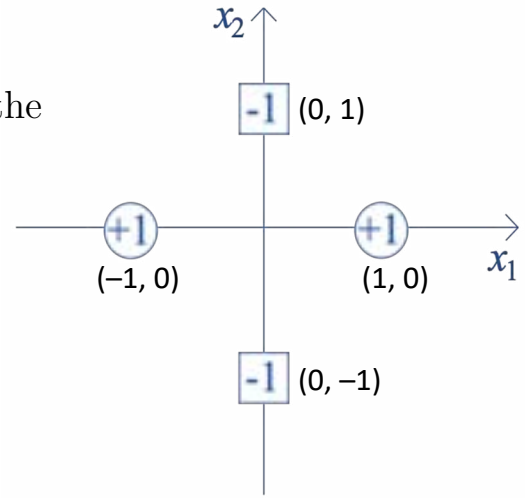
Q2. Using a Bagging algorithm, 8 classifiers shown in Q1 are obtained.

a. Determine the training error (using the training dataset in Q1) given by the Bagging classifier.

b. Instead of using 8 classifiers, which classifiers should be used to form the bagging classifier. Determine the training error using the training dataset in Q1.

Q2. Using a Bagging algorithm, 8 classifiers shown in Q1 are obtained.

a. Determine the training error (using the training dataset in Q1) given by the Bagging classifier.



Classification weight that is 50% because 2 out of 4 samples they are wrongly classified so we obtain this result 0. 5. why? Because by combining the weak classifier we are able to improve the classification performance but this example just show that it is opposite way/conscious. All the weak classifiers required to have a minimum requirement. So how do we accept the weak classifiers in the lcture notes that the weak classifier has to satisfy the minimum requirement, the classificatioen weight that should be at least 50%

The Bagging classifier makes decision based on the majority votes:

Classifier	(1,0), +1	(−1,0), +1	(0,1), −1	(0,−1), −1	Training Error
$h_1(\mathbf{x})$	+1	−1	+1	+1	0.75
$h_2(\mathbf{x})$	−1	+1	−1	−1	0.25
$h_3(\mathbf{x})$	+1	−1	−1	−1	0.25
$h_4(\mathbf{x})$	−1	+1	+1	+1	0.75
$h_5(\mathbf{x})$	+1	+1	+1	−1	0.25
$h_6(\mathbf{x})$	−1	−1	−1	+1	0.75
$h_7(\mathbf{x})$	−1	−1	+1	−1	0.75
$h_8(\mathbf{x})$	+1	+1	−1	+1	0.25
$f_{\text{final}}(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{8} \frac{1}{8} h_i(\mathbf{x})\right)$	sgn(0) = +1	sgn(0) = +1	sgn(0) = +1	sgn(0) = +1	0.5

Sum/8

Correctly classified

Correctly classified

Wrongly classified

Wrongly classified

Training error ¾ which is 0.75

this row sign classified two correct and two incorrect that's why it's 2 out of 4 so 2/4 = 0.5

Is the bagging classifier working well? Why?

When sgn function (0) by just default it assigned the output true label 1 but in exam might comes when it's zero assign -1

2 out of 4 are wrongly classified: 2/4 = 0.5

By default it's +1

Q2. Using a Bagging algorithm, 8 classifiers shown in Q1 are obtained.

b. Instead of using 8 classifiers, which classifiers should be used to form the bagging classifier. Determine the training error using the training dataset in Q1.

Classifier	(1,0), +1	(−1,0), +1	(0,1), −1	(0,−1), −1	Training Error
$h_1(\mathbf{x})$	+1	−1	+1	+1	0.75
$h_2(\mathbf{x})$	−1	+1	−1	−1	0.25
$h_3(\mathbf{x})$	+1	−1	−1	−1	0.25
$h_4(\mathbf{x})$	−1	+1	+1	+1	0.75
$h_5(\mathbf{x})$	+1	+1	+1	−1	0.25
$h_6(\mathbf{x})$	−1	−1	−1	+1	0.75
$h_7(\mathbf{x})$	−1	−1	+1	−1	0.75
$h_8(\mathbf{x})$	+1	+1	−1	+1	0.25

Less training error

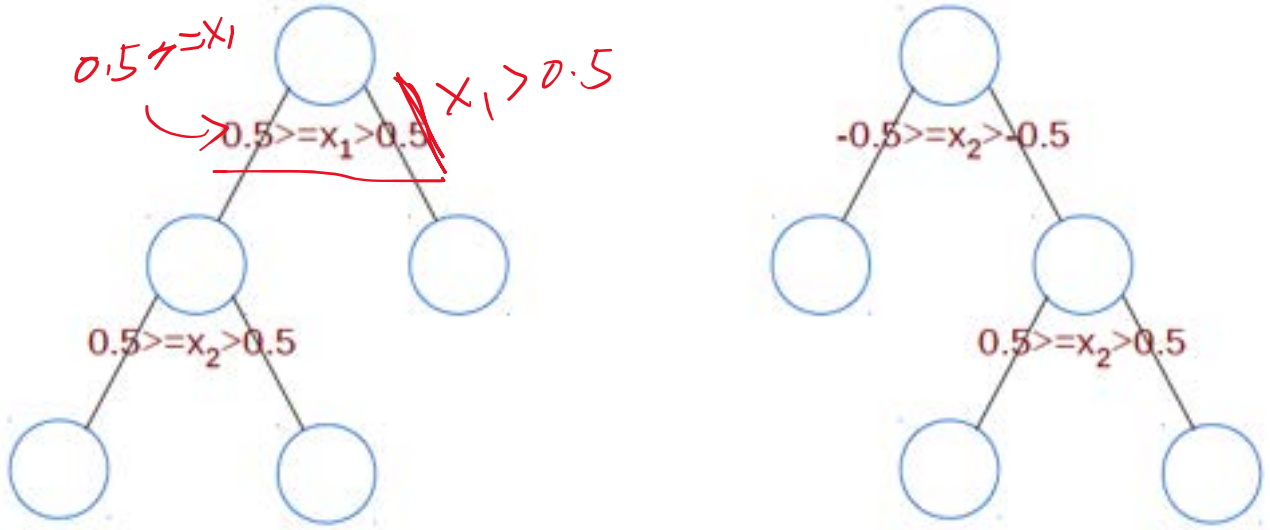
How to improve the performance of bagging classifier?

To improve the Bagging classifier, it is logical to choose only the weak classifiers with training error below 50%, that is $h_2(\mathbf{x})$, $h_3(\mathbf{x})$, $h_5(\mathbf{x})$ and $h_8(\mathbf{x})$.

Classifier	(1,0), +1	(−1,0), +1	(0,1), −1	(0,−1), −1	Training Error
$h_2(\mathbf{x})$	−1	+1	−1	−1	0.25
$h_3(\mathbf{x})$	+1	−1	−1	−1	0.25
$h_5(\mathbf{x})$	+1	+1	+1	−1	0.25
$h_8(\mathbf{x})$	+1	+1	−1	+1	0.25
$f_{\text{final}}(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{4} \frac{1}{4} h_i(\mathbf{x})\right)$	$\text{sgn}(\frac{2}{4}) = +1$	$\text{sgn}(\frac{2}{4}) = +1$	$\text{sgn}(-\frac{2}{4}) = -1$	$\text{sgn}(-\frac{2}{4}) = -1$	0 (0%)

No misclassified so it's error is 0

Q3. Figure 1 shows two binary decision trees.



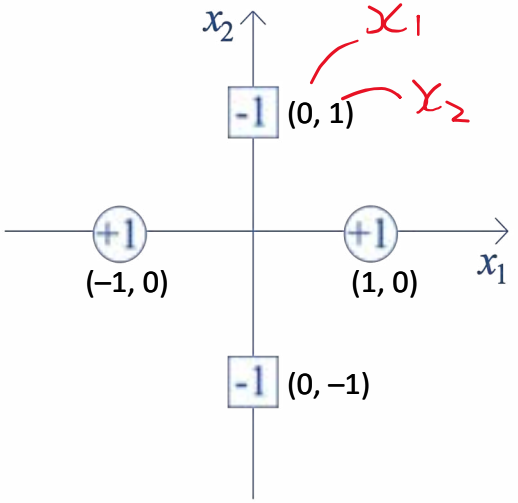


Figure 1: Two decision trees.

a. Using the training data given in Q1, calculate the $P(class = -1|\mathbf{x})$ for each leaf node.

b. Hence, determine the class for a new sample $\mathbf{x} = (0, 0)$ predicted by each decision tree.

c. If the two binary decision trees shown in Figure 1 were learnt using a random forest algorithm. What would be the class for a new sample $\mathbf{x} = (0, 0)$ predicted by this random forest?

Q3. Figure 1 shows two binary decision trees.

a. Using the training data given in Q1, calculate the $P(class = -1|\mathbf{x})$ for each leaf node.

Samples are:

$\{(\mathbf{x} = (1, 0), y = +1),$
$\mathbf{x} = (-1, 0), y = +1),$
$(\mathbf{x} = (0, 1), y = -1),$
$(\mathbf{x} = (0, -1), y = -1)\}$

Sample 1, $x_1 = 1$, $x_2 = 0$, $y = +1$

–1/+1

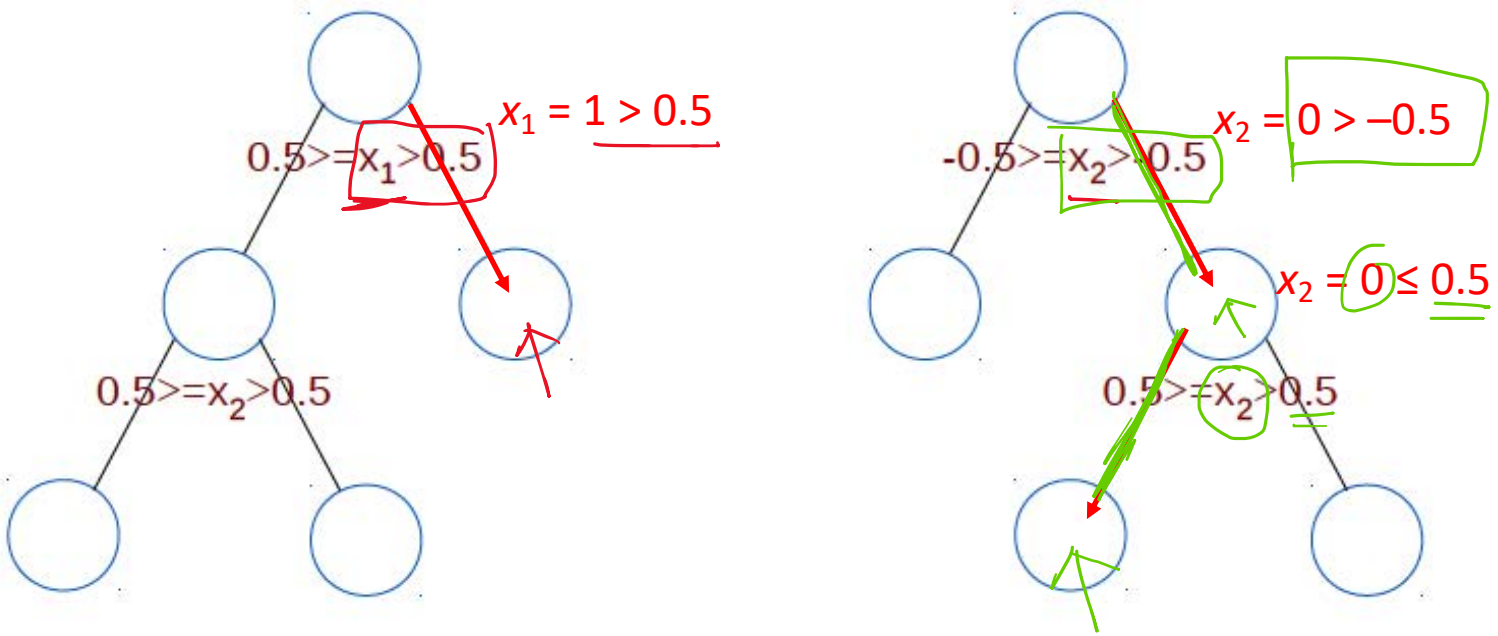


Figure 1: Two decision trees.

Q3. Figure 1 shows two binary decision trees.

a. Using the training data given in Q1, calculate the $P(class = -1|\mathbf{x})$ for each leaf node.

Samples are:

$\{(\mathbf{x} = (1, 0), y = +1),$
$(\mathbf{x} = (-1, 0), y = +1),$
$(\mathbf{x} = (0, 1), y = -1),$
$(\mathbf{x} = (0, -1), y = -1)\}$

Sample 2, $x_1 = -1$, $x_2 = 0$, $y = +1$

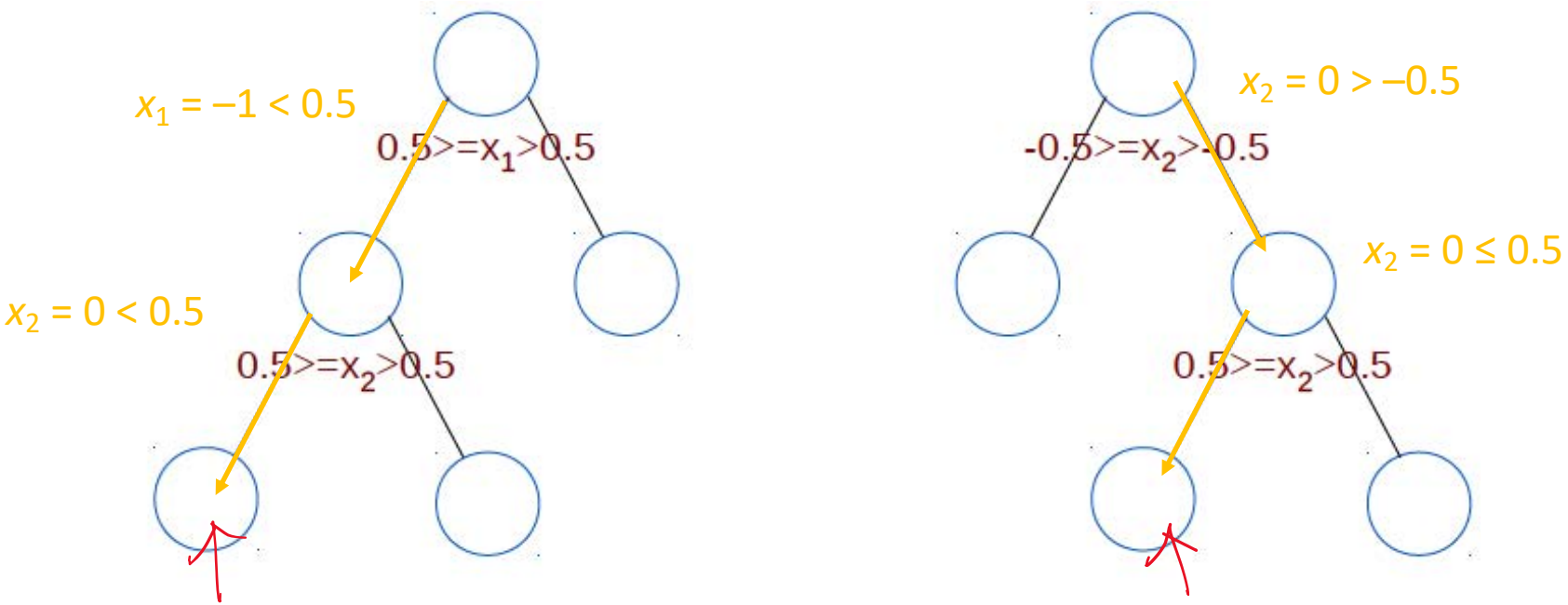


Figure 1: Two decision trees.

Q3. Figure 1 shows two binary decision trees.

a. Using the training data given in Q1, calculate the $P(class = -1|\mathbf{x})$ for each leaf node.

Samples are:

$$\{(\mathbf{x} = (1, 0), y = +1),$$
$$\mathbf{x} = (-1, 0), y = +1),$$
$$(\mathbf{x} = (0, 1), y = -1),$$
$$(\mathbf{x} = (0, -1), y = -1)\}$$

Sample 3, $x_1 = 0$, $x_2 = 1$, $y = -1$

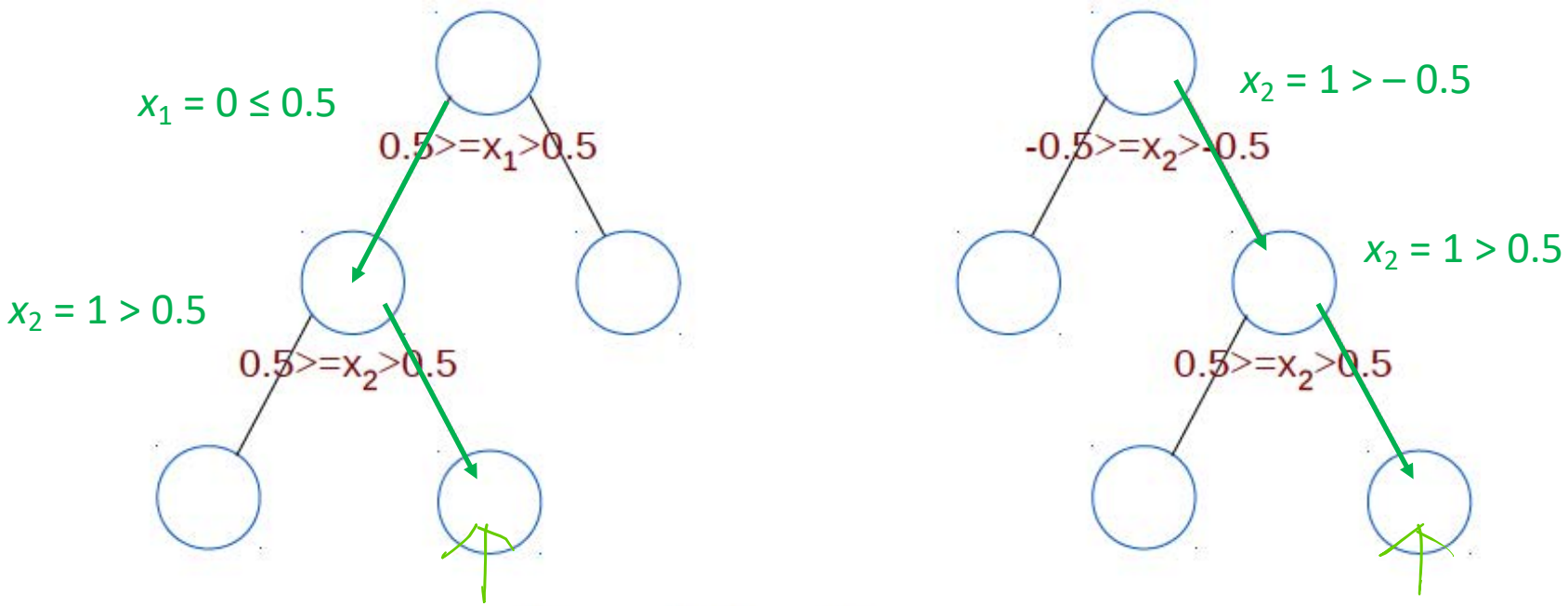


Figure 1: Two decision trees.

Q3. Figure 1 shows two binary decision trees.

a. Using the training data given in Q1, calculate the $P(class = -1|\mathbf{x})$ for each leaf node.

Samples are:

$\{(\mathbf{x} = (1, 0), y = +1),$
$\mathbf{x} = (-1, 0), y = +1),$
$(\mathbf{x} = (0, 1), y = -1),$
$(\mathbf{x} = (0, -1), y = -1)\}$

x_4

Sample 4, $x_1 = 0$, $x_2 = -1$, $y = -1$

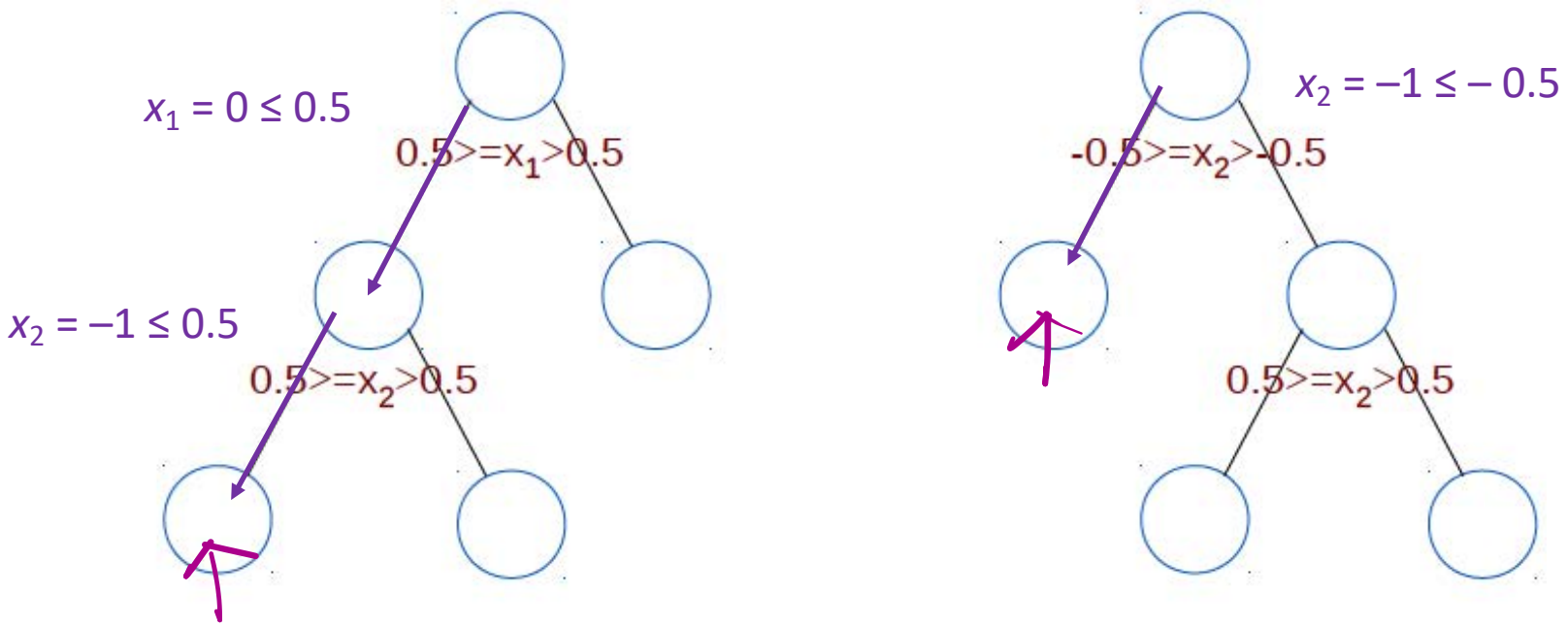


Figure 1: Two decision trees.

Q3. Figure 1 shows two binary decision trees.

a. Using the training data given in Q1, calculate the $P(class = -1|\mathbf{x})$ for each leaf node.

Samples are:

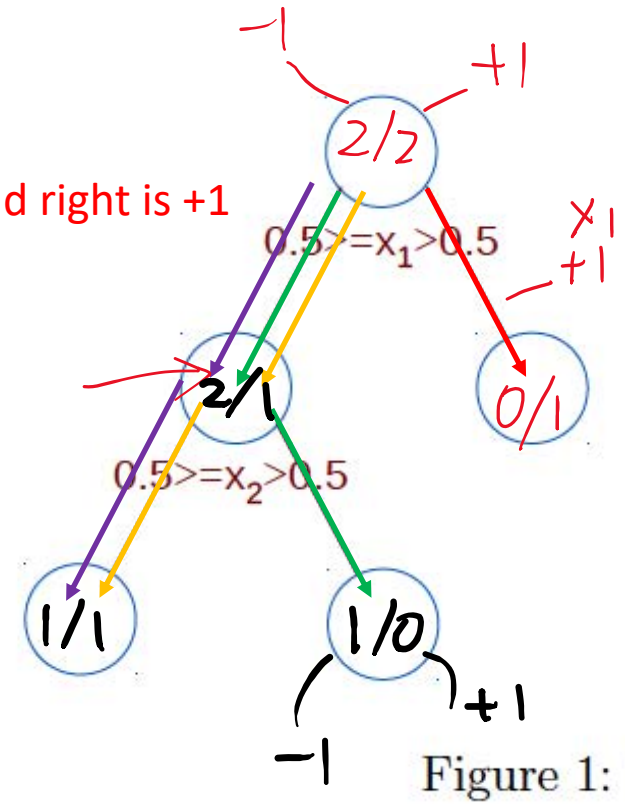
$\{(\mathbf{x} = (1, 0), y = +1),$
$\mathbf{x} = (-1, 0), y = +1),$
$(\mathbf{x} = (0, 1), y = -1),$
$(\mathbf{x} = (0, -1), y = -1)\}$

X_1 X_2 X_3 X_4

We group first we have two +1 and two -1 that's why 2/2 at root node

−1/+1

In every node left is -1 and right is +1



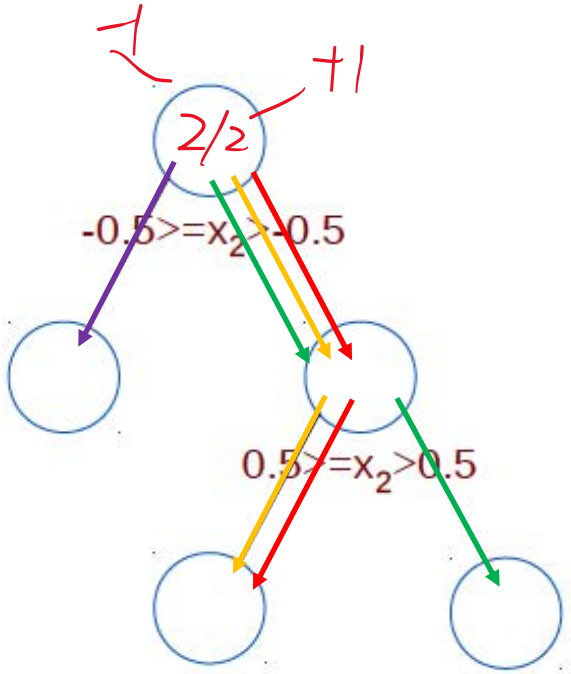


Figure 1: Two decision trees.

Q3. Figure 1 shows two binary decision trees.

a. Using the training data given in Q1, calculate the $P(class = -1|\mathbf{x})$ for each leaf node.

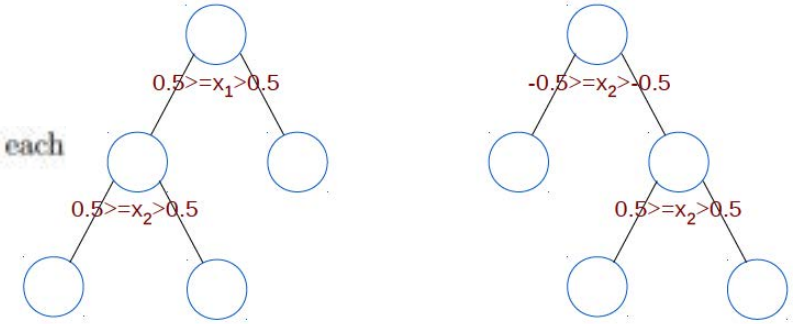


Figure 1: Two decision trees.

The "2" on the left means two samples belong to "–1" class"

The "2" on the right means two samples belong to "+1" class"

Pass each training sample down each tree, and count how many from each class arrive at each leaf node.

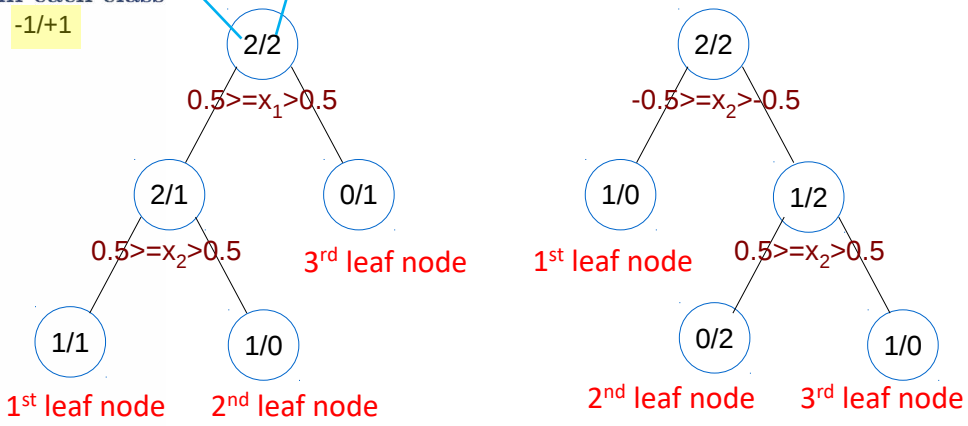
-1/+1

Samples are:
$\{(\mathbf{x} = (1, 0), y = +1),$
$\mathbf{x} = (-1, 0), y = +1),$
$(\mathbf{x} = (0, 1), y = -1),$
$(\mathbf{x} = (0, -1), y = -1)\}$

Sample $(1, 0)$ arrives at the third leaf node in tree 1.
Sample $(-1, 0)$ arrives at the first leaf node in tree 1.
Sample $(0, 1)$ arrives at the second leaf node in tree 1.
Sample $(0, -1)$ arrives at the first leaf node in tree 1.

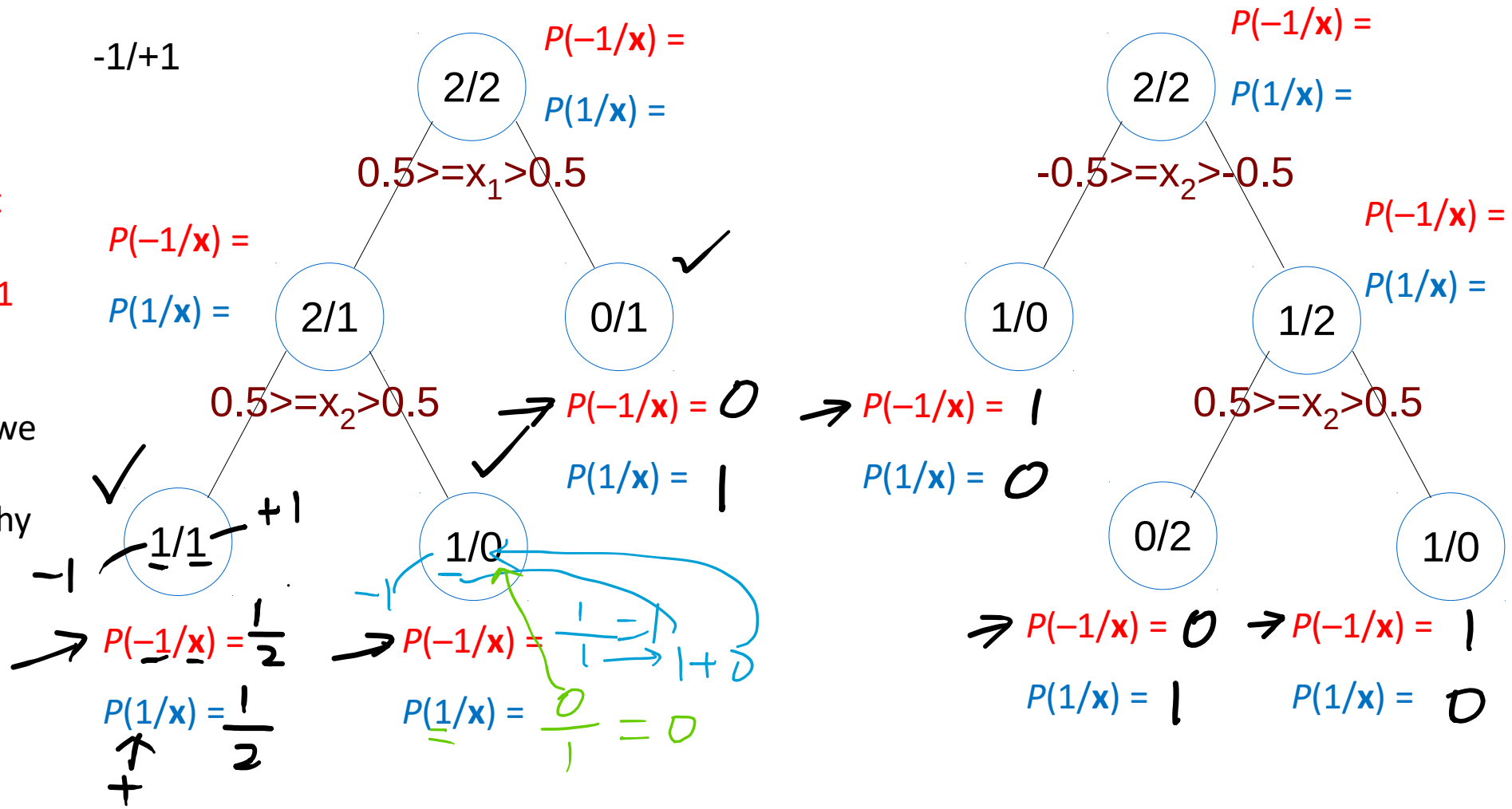
Sample $(1, 0)$ arrives at the second leaf node in tree 2.
Sample $(-1, 0)$ arrives at the second leaf node in tree 2.
Sample $(0, 1)$ arrives at the third leaf node in tree 2.
Sample $(0, -1)$ arrives at the first leaf node in tree 2.

The counts of the number of training samples from each class $(-1/+1)$ arriving at each node are summarised in the figure.



Q3. Figure 1 shows two binary decision trees.

a. Using the training data given in Q1, calculate the $P(class = -1|x)$ for each leaf node.



In every node left is -1 and right is +1

In total we have 2 that's why it's by 2

Q3. Figure 1 shows two binary decision trees.

a. Using the training data given in Q1, calculate the $P(class = -1|\mathbf{x})$ for each leaf node.

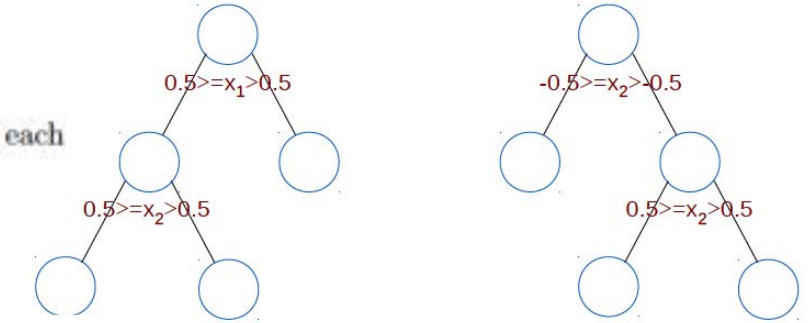
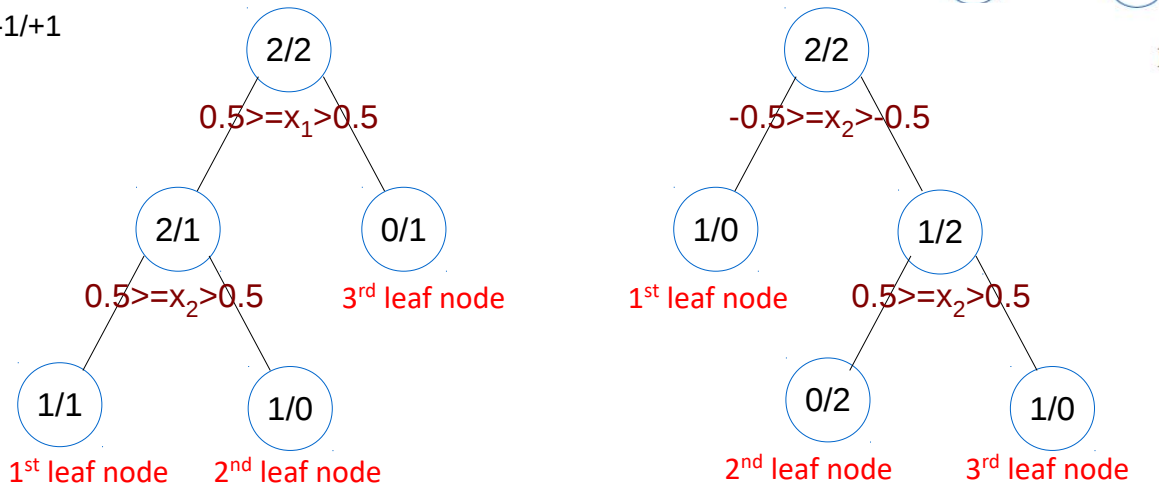


Figure 1: Two decision trees.

-1/+1



1/(1+1) = 0.5

For the first decision tree $P(class = -1|\mathbf{x}) = 0.5$ at the first leaf node, $P(class = -1|\mathbf{x}) = 1$ at the second leaf node, $P(class = -1|\mathbf{x}) = 0$ at the third leaf node.

1/(1+0) = 1

0/(0+1) = 0

1/(1+0) = 1

For the second decision tree $P(class = -1|\mathbf{x}) = 1$ at the first leaf node, $P(class = -1|\mathbf{x}) = 0$ at the second leaf node, $P(class = -1|\mathbf{x}) = 1$ at the third leaf node.

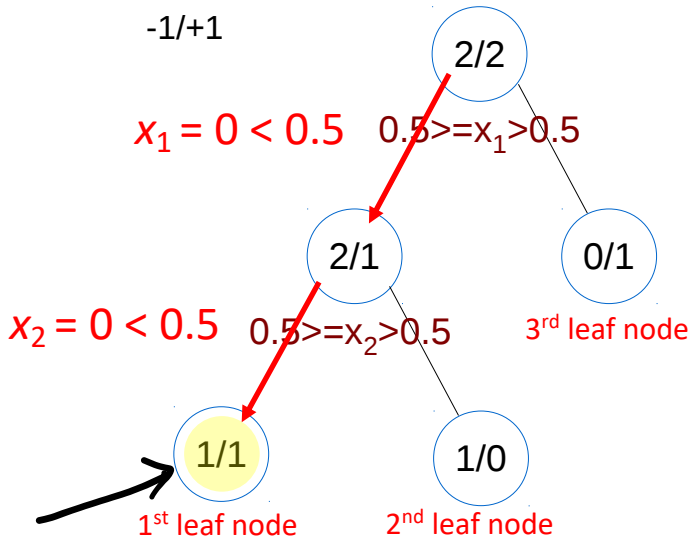
0/(1+1) = 0

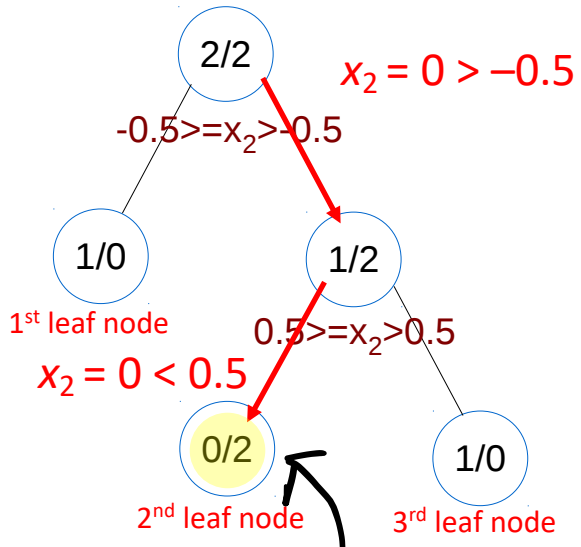
1/(1+0) = 1

Q3. Figure 1 shows two binary decision trees.

b. Hence, determine the class for a new sample $\mathbf{x} = (0, 0)$ predicted by each decision tree.

x_1, x_2





To classify the new sample, it is passed down the tree until it reaches a leaf node. It is given the class label, ω_j, for which $P(\omega_j|\mathbf{x})$ is a maximum at that leaf node.

Sample $(0, 0)$ arrives at the first leaf node in tree 1. At that leaf node $P(class = -1|\mathbf{x}) = 0.5$ and $P(class = +1|\mathbf{x}) = 0.5$ so class of new sample is indeterminate.

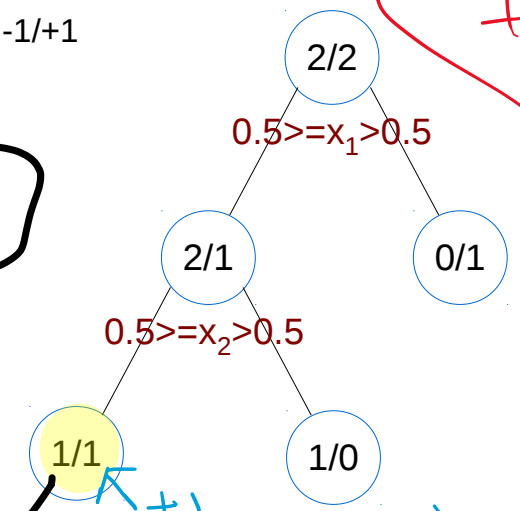
Sample $(0, 0)$ arrives at the second leaf node in tree 2. At that leaf node $P(class = -1|\mathbf{x}) = 0$ and $P(class = +1|\mathbf{x}) = 1$ so class of new sample is $+1$.

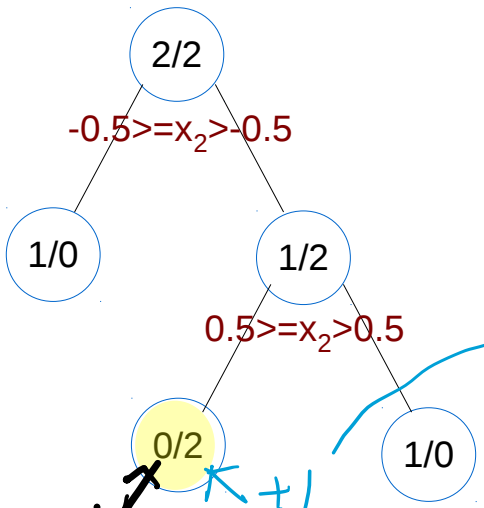
Which class **x** = (0, 0) belongs to?

Q3. Figure 1 shows two binary decision trees.

c. If the two binary decision trees shown in Figure 1 were learnt using a random forest algorithm. What would be the class for a new sample $\mathbf{x} = (0, 0)$ predicted by this random forest?

$$P(\omega|\mathbf{x}) = \frac{1}{M}\sum_{t=1}^{M} P_t(\omega|\mathbf{x})$$





M = 2

+1

-1

P(-1/x) = 1/2 = 0.5

P(+1/x) = 1/2 = 0.5

P(+1/x) = 2/2 = 1

-1

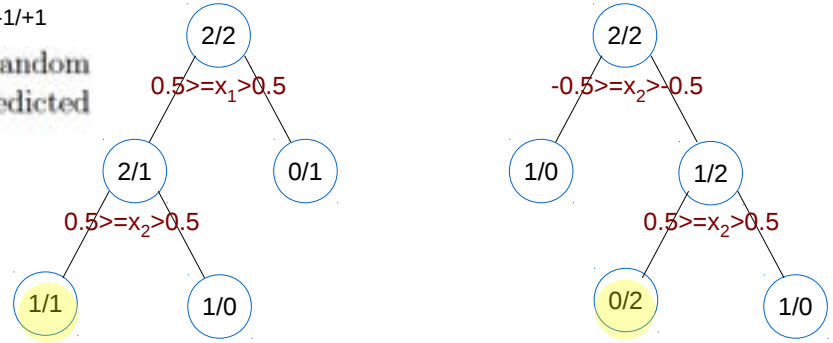
P(-1/x) = 0/2 = 0

P(-1/x) = 1/2 (0.5 + 0) = 0.25

P(+1/x) = 1/2 (0.5 + 1) = 0.75

Q3. Figure 1 shows two binary decision trees.

c. If the two binary decision trees shown in Figure 1 were learnt using a random forest algorithm. What would be the class for a new sample $\mathbf{x} = (0, 0)$ predicted by this random forest?



Classification in a random forest is the same procedure as for bagged decision trees:

- A sample is passed down all trees to reach a leaf node in each tree
- Each leaf node is associated with a class probability distribution $P(\omega|\mathbf{x})$
- Overall class probability distribution is the mean of those associated with each leaf node reached by the sample:

$$P(\omega|\mathbf{x}) = \frac{1}{M}\sum_{t=1}^{M} P_t(\omega|\mathbf{x})$$

So for sample $(0, 0)$:

Tree 1: 1/(1+1) Tree 2: 0/(0+2)

$$P(class = -1|\mathbf{x}) = \frac{1}{2}(0.5 + 0) = 0.25$$

$$P(class = +1|\mathbf{x}) = \frac{1}{2}(0.5 + 1) = 0.75$$

So new sample is in class $+1$.

Tree 1: 1/(1+1) Tree 2: 2/(0+2)

Pattern Recognition, Neural Networks and Deep Learning (7CCSMPNN)

Tutorial 10: Solutions

Q1. Apply the K-means algorithm to the following dataset, S, to find 2 natural groupings (K = 2) using the Euclidean distance criterion. Start with initial values of cluster centres of m_1 and m_2 accordingly.

$$S = \begin{matrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{matrix}, \quad m_1 = \begin{matrix} -1 \\ 3 \end{matrix}, \quad m_2 = \begin{matrix} 5 \\ 1 \end{matrix}$$

Plot the results and show a suitable linear discriminant function for the two clusters and give its weight vector (assuming $g(x) = w^T x = 0$) Once we find the distance which is comes closer we assign the class for that sample

a. **Iteration 1:** Initial cluster centres: $\mathbf{m}_1 = \begin{bmatrix} -1 \\ 3 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$

$\sqrt{(-1-5)^2 + (3-1)^2} = \sqrt{40}$

Sample $\mathbf{x}$	$\|\mathbf{x} - \mathbf{m}_1\|$	$\|\mathbf{x} - \mathbf{m}_2\|$	Class
$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	0	$\sqrt{40}$	1
$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$	$\sqrt{5}$	5	1
$\begin{bmatrix} 0 \\ 5 \end{bmatrix}$	$\sqrt{5}$	$\sqrt{41}$	1
$\begin{bmatrix} 4 \\ -1 \end{bmatrix}$	$\sqrt{41}$	$\sqrt{5}$	2
$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	5	$\sqrt{5}$	2
$\begin{bmatrix} 5 \\ 1 \end{bmatrix}$	$\sqrt{40}$	0	2

New cluster centres:

$$\mathbf{m}_1 = \frac{\mathbf{x}_1+\mathbf{x}_2+\mathbf{x}_3}{3} = \begin{bmatrix} 0 \\ 4 \end{bmatrix}, \quad \mathbf{m}_2 = \frac{\mathbf{x}_4+\mathbf{x}_5+\mathbf{x}_6}{3} = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

All X sample i.e. vectors add up and divide and this is the new cluster center for the next iteration

Algorithm: K-means clustering

begin initialise n, c $(K = c)$, $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$
  do classify n samples according to the nearest $\mathbf{m}_j$, $j \in \{1,2,\ldots,c\}$ †;
    recompute $\mathbf{m}_j$ ‡
  until no change in $\mathbf{m}_j$
  return $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$
end

Q1. Apply the K-means algorithm to the following dataset, S, to find 2 natural groupings (K = 2) using the Euclidean distance criterion. Start with initial values of cluster centres of m_1 and m_2 accordingly.

$$S = \begin{matrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{matrix}, \quad m_1 = \begin{matrix} -1 \\ 3 \end{matrix}, \quad m_2 = \begin{matrix} 5 \\ 1 \end{matrix}$$

Plot the results and show a suitable linear discriminant function for the two clusters and give its weight vector (assuming $g(x) = w^T x = 0$)

Iteration 2: Current cluster centres: $\mathbf{m}_1 = \begin{bmatrix} 0 \\ 4 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$.

Sample $\mathbf{x}$	$\|\mathbf{x} - \mathbf{m}_1\|$	$\|\mathbf{x} - \mathbf{m}_2\|$	Class
$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	$\sqrt{2}$	$\sqrt{34}$	1
$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$	1	5	1
$\begin{bmatrix} 0 \\ 5 \end{bmatrix}$	1	$\sqrt{41}$	1
$\begin{bmatrix} 4 \\ -1 \end{bmatrix}$	$\sqrt{41}$	1	2
$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	5	1	2
$\begin{bmatrix} 5 \\ 1 \end{bmatrix}$	$\sqrt{34}$	$\sqrt{2}$	2

Using the cluster we found after averaging in the previous iteration/slides

Key points:
Once cluster centers doesn't change it's converged. No need to go through again for next iteration

New cluster centres:

$$\mathbf{m}_1 = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3}{3} = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$$

$$\mathbf{m}_2 = \frac{\mathbf{x}_4 + \mathbf{x}_5 + \mathbf{x}_6}{3} = \begin{bmatrix} 4 \\ 0 \end{bmatrix}.$$

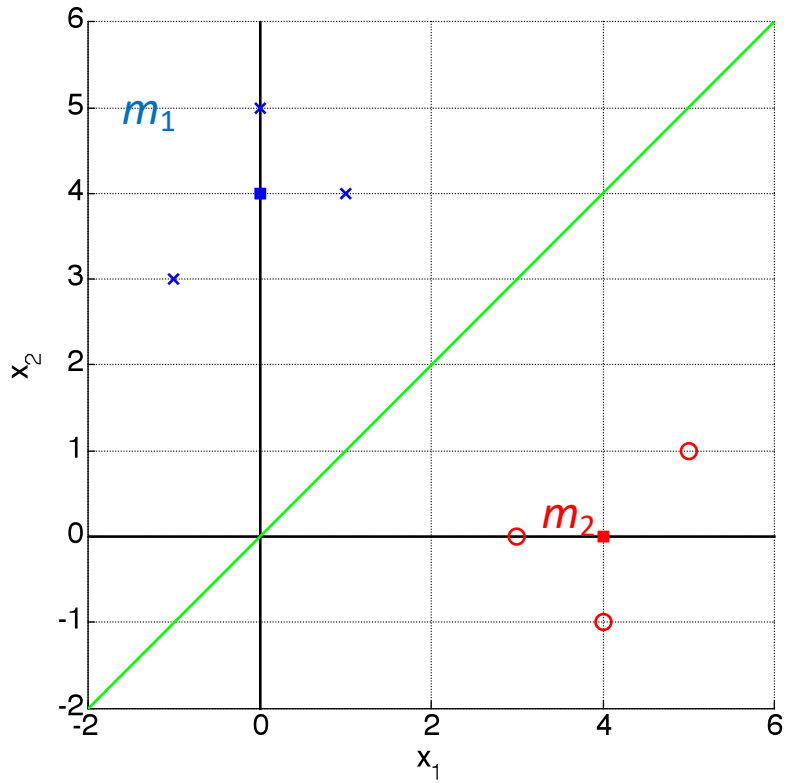
Q1. Apply the K-means algorithm to the following dataset, S, to find 2 natural groupings (K = 2) using the Euclidean distance criterion. Start with initial values of cluster centres of m_1 and m_2 accordingly.

$$S = \begin{matrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{matrix}, \quad m_1 = \begin{matrix} -1 \\ 3 \end{matrix}, \quad m_2 = \begin{matrix} 5 \\ 1 \end{matrix}$$

Plot the results and show a suitable linear discriminant function for the two clusters and give its weight vector (assuming $g(x) = w^T x = 0$)

Iteration 3: No change, hence means have converged no further updates needed.

Final cluster centres: $\mathbf{m}_1 = \begin{bmatrix} 0 \\ 4 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$.



Q2. Consider the dataset $X = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, use principal component analysis (PCA) to:

a. project the data onto 2D plane,

b. project the data onto 1D plane.

c. The dataset consists of samples from two classes. By observation, find the cluster centres based on the transformed 2D and 1D samples. Classify the new data $x = \begin{bmatrix} 3 \\ -2 \\ 5 \end{bmatrix}$ (after removing the mean) using both sets of clusters centres.

Q2. Consider the dataset $X = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, use principal component analysis (PCA) to:

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$

a. project the data onto 2D plane,

Transform (Karhunen-Loève Transform)

Dataset: $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \end{bmatrix} \in R^{d \times n}$

Sample: $\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix} \in R^d$

Mean: $\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$

Covariance matrix: $\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^T$

Singular value decomposition: $\mathbf{C} = \mathbf{V}\mathbf{D}\mathbf{V}^T$

KL transformed sample: $\mathbf{x}_{KL} = \hat{\mathbf{V}}^T(\mathbf{x} - \mathbf{m})$

$$\mathbf{m} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix},$$

Finding mean by row of X matrix

$$\mathbf{X}_m = \begin{bmatrix} 3 & -1 & 1 & -3 \\ 1 & -3 & 3 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

Zero mean matrix. Matrix X – Mean vector

$$\mathbf{C} = \begin{bmatrix} 5 & 3 & 0 \\ 3 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Co-variance matrix

V is eigen vectors,
D is eigen values

$$\mathbf{V} = \begin{bmatrix} -0.7071 & -0.7071 & 0 \\ -0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\hat{\mathbf{V}}^T = \begin{bmatrix} -0.7071 & -0.7071 & 0 \\ -0.7071 & 0.7071 & 0 \end{bmatrix}$$

We need to choose the two eigenvectors corresponding to the two largest eigenvectors and Projection of the data onto the subspace spanned by the 1st two principal components

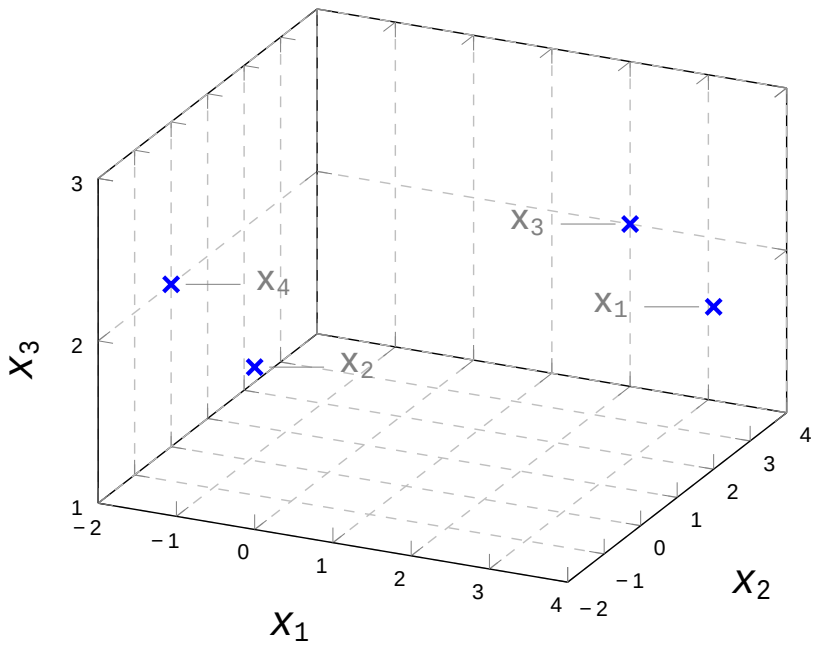
$$\mathbf{X}_{KL} = \begin{bmatrix} -2.8284 & 2.8284 & -2.8284 & 2.8284 \\ -1.4142 & -1.4142 & 1.4142 & 1.4142 \end{bmatrix} \text{(using two components)}$$

X_KL = V^T matrix matrix multiplication with X_m

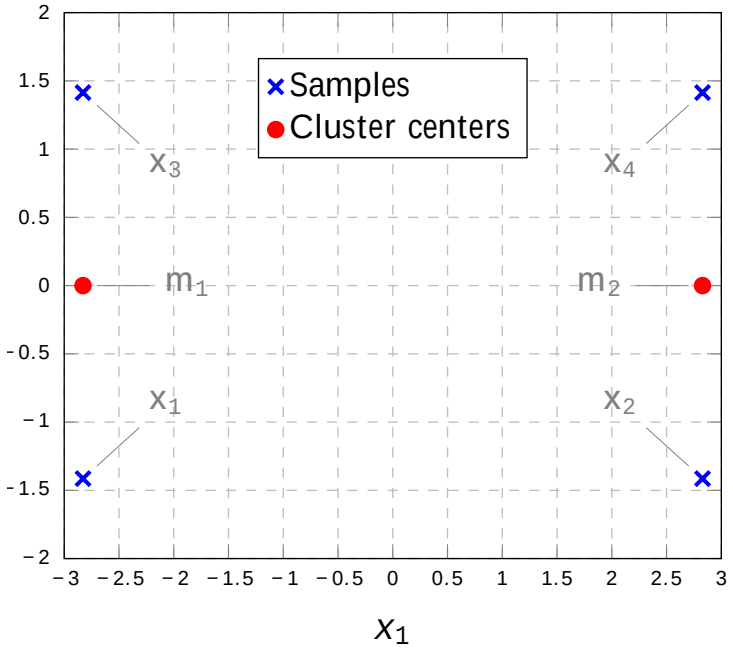
Q2. Consider the dataset $X = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, use principal component analysis (PCA) to:

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$

a. project the data onto 2D plane,



(a) 3D plot of samples.



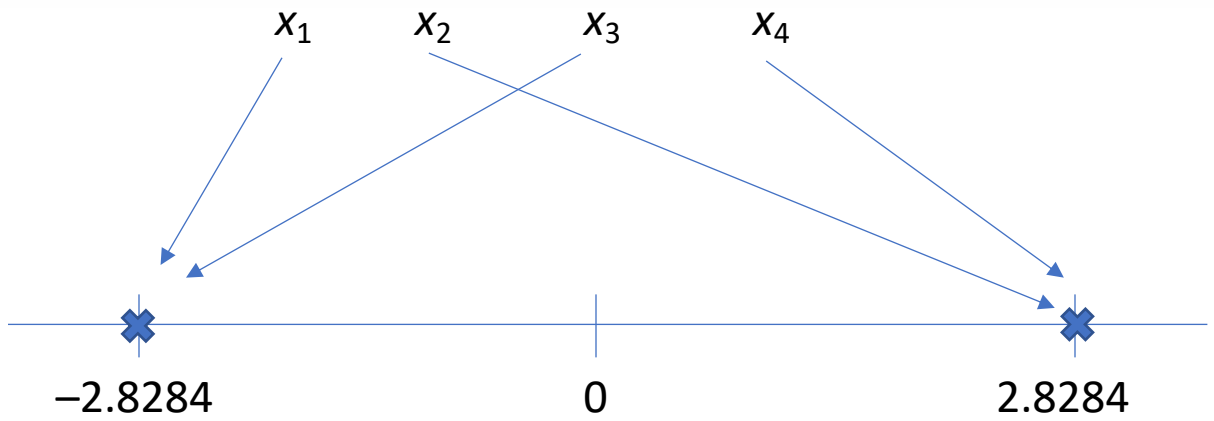
(b) 2D plot of transformed samples.

$$\mathbf{X}_{KL} = \begin{bmatrix} -2.8284 & 2.8284 & -2.8284 & 2.8284 \\ -1.4142 & -1.4142 & 1.4142 & 1.4142 \end{bmatrix} \text{(using two components)}$$

Q2. Consider the dataset $X = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, use principal component analysis (PCA) to:

b. project the data onto 1D plane.

$$\hat{\mathbf{V}}^T = \begin{bmatrix} -0.7071 & -0.7071 & 0 \end{bmatrix}$$
$$\mathbf{X}_{KL} = \begin{bmatrix} -2.8284 & 2.8284 & -2.8284 & 2.8284 \end{bmatrix} \text{ (using one component)}$$



Q2. Consider the dataset $X = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, use principal component analysis (PCA) to:

c. The dataset consists of samples from two classes. By observation, find the cluster centres based on the transformed 2D and 1D samples. Classify the new data $x = \begin{bmatrix} 3 \\ -2 \\ 5 \end{bmatrix}$ (after removing the mean) using both sets of clusters centres.

From the 2D plot, we choose:

$$\mathbf{m}_1 = \begin{bmatrix} -2.8284 \\ 0 \end{bmatrix} \text{ and } \mathbf{m}_2 = \begin{bmatrix} 2.8284 \\ 0 \end{bmatrix}.$$

Transformed data:

$$\mathbf{x}_{LT} = \hat{\mathbf{V}}^T\mathbf{x} = \begin{bmatrix} -0.7071 & -0.7071 & 0 \\ -0.7071 & 0.7071 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \\ 5 \end{bmatrix} = \begin{bmatrix} -0.7071 \\ -3.5355 \end{bmatrix}.$$

Computing the Euclidean distance:

$\|\mathbf{x}_{LT} - \mathbf{m}_1\| = 4.1231$ and $\|\mathbf{x}_{LT} - \mathbf{m}_2\| = 5$,

the new data is classified as class 1

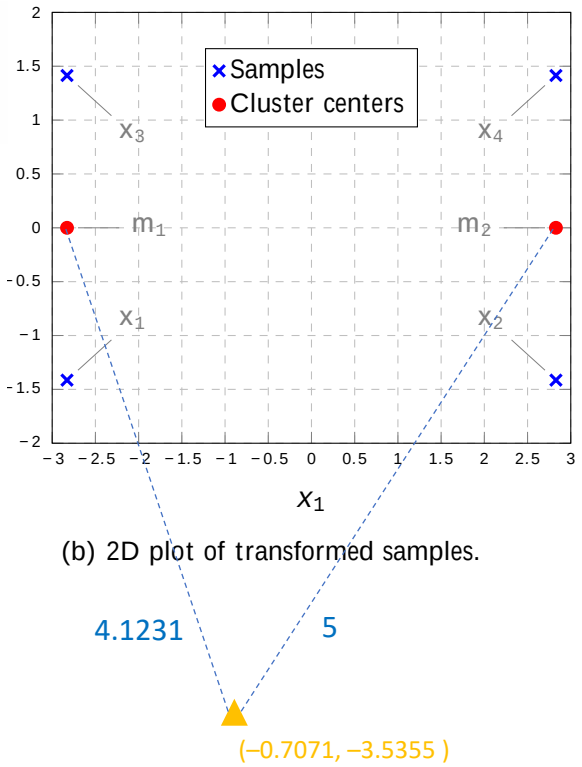
Remark: ?

$\mathbf{m}_1$: class 1

- $\mathbf{x}_1$ and $\mathbf{x}_3$ belong to class 1

$\mathbf{m}_2$: class 2

- $\mathbf{x}_2$ and $\mathbf{x}_4$ belong to class 2



(b) 2D plot of transformed samples.

Q2. Consider the dataset $X = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, use principal component analysis (PCA) to:

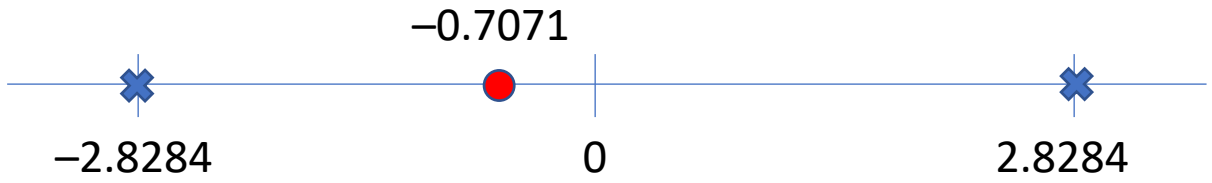
c. The dataset consists of samples from two classes. By observation, find the cluster centres based on the transformed 2D and 1D samples. Classify the new data $x = \begin{bmatrix} 3 \\ -2 \\ 5 \end{bmatrix}$ (after removing the mean) using both sets of clusters centres.

Form the 1D plot, we choose:
$\mathbf{m}_1 = [\ -2.8284\ ]$ and $\mathbf{m}_2 = [\ 2.8284\ ]$.

Transformed data:

$$\mathbf{x}_{LT} = \hat{\mathbf{V}}^T\mathbf{x} = \begin{bmatrix} -0.7071 & -0.7071 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \\ 5 \end{bmatrix} = [\ -0.7071\ ].$$

As -0.7071 is closer to m_1 ($\|\mathbf{x}_{LT} - \mathbf{m}_1\| = 2.1213$ and $\|\mathbf{x}_{LT} - \mathbf{m}_2\| = 3.5355$), the new data is classified as class 1. ?



Q3. Competitive learning algorithm (without normalisation) is employed to find 3 cluster centres for the dataset S in Q1. The initial cluster centres are chosen as $x_1/2$, $x_3/2$ and $x_5/2$, and $\eta = 0.1$. The algorithm is run for 5 iterations and the samples are chosen in the order of x_3, x_1, x_1, x_5, x_6 where the left most data is chosen first.

a. Find the cluster centres for each iteration.

b. Plot the samples S and cluster centres in a figure. Classify the samples.

c. Classify the new data $x = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$.

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$$

Q3. Competitive learning algorithm (without normalisation) is employed to find 3 cluster centres for the dataset S in Q1. The initial cluster centres are chosen as $x_1/2$, $x_3/2$ and $x_5/2$, and $\eta = 0.1$. The algorithm is run for 5 iterations and the samples are chosen in the order of x_3, x_1, x_1, x_5, x_6 where the left most data is chosen first.

Pseudo Code: Algorithm for Competitive learning without normalisation

begin initialise η (> 0), n, c, $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_c$
  do randomly select a pattern $\mathbf{x}$;
    $j \leftarrow \arg\min_{j'} \|\mathbf{x} - \mathbf{w}_{j'}\|$ (classify $\mathbf{x}$)
    $\mathbf{w}_j \leftarrow \mathbf{w}_j + \eta(\mathbf{x} - \mathbf{w}_j)$ (weight update)
  until no significant change in $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_c$
  return $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_c$
end

a. Find the cluster centres for each iteration.

$$\mathbf{m}_1 = \begin{bmatrix} -0.5 \\ 1.5 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 0 \\ 2.5 \end{bmatrix}, \mathbf{m}_3 = \begin{bmatrix} 1.5 \\ 0 \end{bmatrix}.$$

M (cluster centers) divided by 2 as question suggested

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$$

Sample in iteration is chosen as in the order as question

Distance between sample and (updated) cluster centre

Identify the cluster centre with the shortest distance from sample

Updated the identified cluster centre

Last column once we find identify cluster center we update that cluster so in this case it's m2 and we use new m2 for m2 to find distance in next iteration

Updated $\mathbf{m}_2$ is used

Iteration	Sample $\mathbf{x}$	$\|\mathbf{x} - \mathbf{m}_1\|$	$\|\mathbf{x} - \mathbf{m}_2\|$	$\|\mathbf{x} - \mathbf{m}_3\|$	$j \leftarrow \arg\min_{j'} \|\mathbf{x} - \mathbf{m}_{j'}\|$	$\mathbf{m}_j \leftarrow \mathbf{m}_j + \eta(\mathbf{x} - \mathbf{m}_j)$
1	x_3 $\begin{bmatrix} 0 \\ 5 \end{bmatrix}$	3.5355	2.5000	5.2202	2	$\mathbf{m}_2 = \begin{bmatrix} 0 \\ 2.5 \end{bmatrix} + 0.1 \times \left(\begin{bmatrix} 0 \\ 5 \end{bmatrix} - \begin{bmatrix} 0 \\ 2.5 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 2.75 \end{bmatrix}$
2	x_1 $\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	1.5811	1.0308	3.9051	2	$\mathbf{m}_2 = \begin{bmatrix} 0 \\ 2.75 \end{bmatrix} + 0.1 \times \left(\begin{bmatrix} -1 \\ 3 \end{bmatrix} - \begin{bmatrix} 0 \\ 2.75 \end{bmatrix} \right) = \begin{bmatrix} -0.1 \\ 2.775 \end{bmatrix}$
3	x_1 $\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	1.5811	0.9277	3.9051	2	$\mathbf{m}_2 = \begin{bmatrix} -0.1 \\ 2.775 \end{bmatrix} + 0.1 \times \left(\begin{bmatrix} -1 \\ 3 \end{bmatrix} - \begin{bmatrix} -0.1 \\ 2.775 \end{bmatrix} \right) = \begin{bmatrix} -0.19 \\ 2.7975 \end{bmatrix}$
4	x_5 $\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	3.8079	4.2429	1.5000	3	$\mathbf{m}_3 = \begin{bmatrix} 1.5 \\ 0 \end{bmatrix} + 0.1 \times \left(\begin{bmatrix} 3 \\ 0 \end{bmatrix} - \begin{bmatrix} 1.5 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 1.65 \\ 0 \end{bmatrix}$
5	x_6 $\begin{bmatrix} 5 \\ 1 \end{bmatrix}$	5.5227	5.4925	3.4961	3	$\mathbf{m}_3 = \begin{bmatrix} 1.65 \\ 0 \end{bmatrix} + 0.1 \times \left(\begin{bmatrix} 5 \\ 1 \end{bmatrix} - \begin{bmatrix} 1.65 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 1.985 \\ 0.1 \end{bmatrix}$

Updated $\mathbf{m}_3$ is used

After 5 iterations, we have $\mathbf{m}_1 = \begin{bmatrix} -0.5 \\ 1.5 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} -0.19 \\ 2.7975 \end{bmatrix}, \mathbf{m}_3 = \begin{bmatrix} 1.985 \\ 0.1 \end{bmatrix}$.

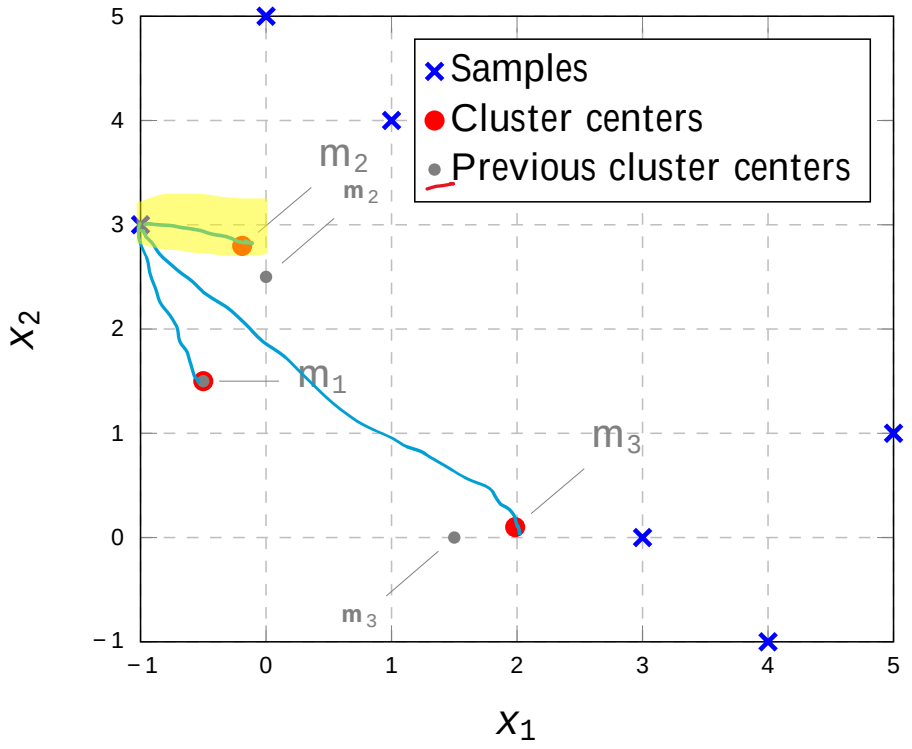
This part is the m/cluster centers updated after 5 runs so m1 did update and m2 and m3 are updated

Q3. Competitive learning algorithm (without normalisation) is employed to find 3 cluster centres for the dataset S in Q1. The initial cluster centres are chosen as $x_1/2$, $x_3/2$ and $x_5/2$, and $\eta = 0.1$. The algorithm is run for 5 iterations and the samples are chosen in the order of x_3, x_1, x_1, x_5, x_6 where the left most data is chosen first.

b. Plot the samples S and cluster centres in a figure. Classify the samples.

$$\mathbf{m}_1 = \begin{bmatrix} -0.5 \\ 1.5 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} -0.19 \\ 2.7975 \end{bmatrix}, \mathbf{m}_3 = \begin{bmatrix} 1.985 \\ 0.1 \end{bmatrix}.$$

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$$

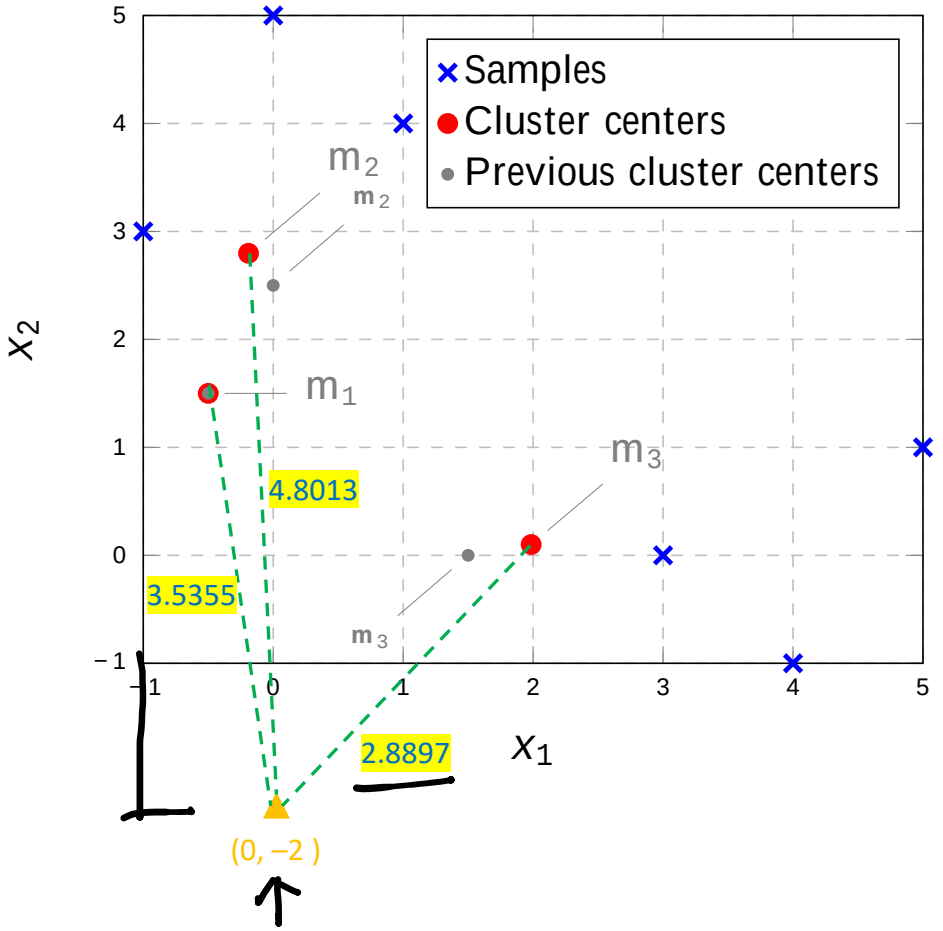


Sample $\mathbf{x}$	Class
$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	2
$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$	2
$\begin{bmatrix} 0 \\ 5 \end{bmatrix}$	2
$\begin{bmatrix} 4 \\ -1 \end{bmatrix}$	3
$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	3
$\begin{bmatrix} 5 \\ 1 \end{bmatrix}$	3

Q3. Competitive learning algorithm (without normalisation) is employed to find 3 cluster centres for the dataset S in Q1. The initial cluster centres are chosen as $x_1/2$, $x_3/2$ and $x_5/2$, and $\eta = 0.1$. The algorithm is run for 5 iterations and the samples are chosen in the order of x_3, x_1, x_1, x_5, x_6 where the left most data is chosen first.

c. Classify the new data $x = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$. ← m_3

To classify we use updated cluster centers to classify the class for the new dataset



Q4. Basic leader follower algorithm (without normalisation), with $\theta = 3$ and $\eta = 0.5$, is employed to find the cluster centres of the dataset S in Q1. The algorithm is run for 5 iterations and the samples are chosen in the order of x_3, x_1, x_1, x_5, x_6 where the left most data is chosen first.

a. Find the cluster centres for each iteration.

b. Classify the samples in S.

c. Classify the new data $x = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$.

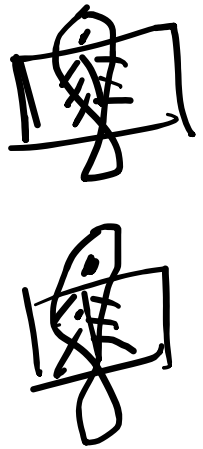
Theta is the threshold

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$$

Version 1: Basic leader-follower clustering without normalisation

Algorithm: Basic leader-follower clustering without normalisation

begin initialise η (> 0), $\theta \leftarrow$ threshold
  $\mathbf{m}_1 \leftarrow \mathbf{x}$ (create first cluster)
  do accept new $\mathbf{x}$;
    $j \leftarrow \arg\min_{j'} \|\mathbf{x} - \mathbf{m}_{j'}\|$ (find the nearest cluster)
    if $\|\mathbf{x} - \mathbf{m}_j\| < \theta$
      $\mathbf{m}_j \leftarrow \mathbf{m}_j + \eta(\mathbf{x} - \mathbf{m}_j)$ (update cluster centre)
    else
      add new $\mathbf{m} \leftarrow \mathbf{x}$ (create a new cluster $\mathbf{m}$)
    end
  until no significant change in $\mathbf{w}$
  return $\mathbf{m}_1, \mathbf{m}_2, \cdots$
end



Q4. Basic leader follower algorithm (without normalisation), with $\theta = 3$ and $\eta = 0.5$, is employed to find the cluster centres of the dataset S in Q1. The algorithm is run for 5 iterations and the samples are chosen in the order of x_3, x_1, x_2, x_5, x_6 where the left most data is chosen first.

a. Find the cluster centres for each iteration.

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$$

Version 1: Basic leader-follower clustering without normalisation

Algorithm: Basic leader-follower clustering without normalisation

begin initialise η (> 0), $\theta \leftarrow$ threshold
  $\mathbf{m}_1 \leftarrow \mathbf{x}$ (create first cluster)
  do accept new $\mathbf{x}$;
    $j \leftarrow \arg\min_j \|\mathbf{x} - \mathbf{m}_j\|$ (find the nearest cluster)
    if $\|\mathbf{x} - \mathbf{m}_j\| < \theta$
      $\mathbf{m}_j \leftarrow \mathbf{m}_j + \eta(\mathbf{x} - \mathbf{m}_j)$ (update cluster centre)
    else
      add new $\mathbf{m} \leftarrow \mathbf{x}$ (create a new cluster $\mathbf{m}$)
    end
  until no significant change in $\mathbf{w}$
  return $\mathbf{m}_1, \mathbf{m}_2, \cdots$
end

Each iteration we do use updated new cluster

Created cluster centres

Distance between sample and (updated) cluster centre

Identify the cluster centre with the shortest distance from sample

Distance less than threshold? If < threshold, update cluster centre, otherwise create a new cluster

Updated the identified cluster centre:
$\mathbf{m}_j \leftarrow \mathbf{m}_j + \eta(\mathbf{x} - \mathbf{m}_j)$
or create a new cluster centre

iteration	Sample $\mathbf{x}$	Cluster Centres	$\|\mathbf{x} - \mathbf{m}_j\|$	$j \leftarrow \arg\min_{j'} \|\mathbf{x} - \mathbf{m}_{j'}\|$	$\|\mathbf{x} - \mathbf{m}_j\| < \theta$	Update/New Cluster
1	$\begin{bmatrix} 0 \\ 5 \end{bmatrix}$	$\mathbf{m}_1 = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$	$\|\mathbf{x} - \mathbf{m}_1\| = 0$	1	yes	$\mathbf{m}_1 = \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.5 \times \left(\begin{bmatrix} 0 \\ 5 \end{bmatrix} - \begin{bmatrix} 0 \\ 5 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$
2	$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	$\mathbf{m}_1 = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$	$\|\mathbf{x} - \mathbf{m}_1\| = 2.2361$	1	yes	$\mathbf{m}_1 = \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.5 \times \left(\begin{bmatrix} -1 \\ 3 \end{bmatrix} - \begin{bmatrix} 0 \\ 5 \end{bmatrix} \right) = \begin{bmatrix} -0.5 \\ 4 \end{bmatrix}$
3	$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	$\mathbf{m}_1 = \begin{bmatrix} -0.5 \\ 4 \end{bmatrix}$	$\|\mathbf{x} - \mathbf{m}_1\| = 1.1180$	1	yes	$\mathbf{m}_1 = \begin{bmatrix} -0.5 \\ 4 \end{bmatrix} + 0.5 \times \left(\begin{bmatrix} -1 \\ 3 \end{bmatrix} - \begin{bmatrix} -0.5 \\ 4 \end{bmatrix} \right) = \begin{bmatrix} -0.75 \\ 3.5 \end{bmatrix}$
4	$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	$\mathbf{m}_1 = \begin{bmatrix} -0.75 \\ 3.5 \end{bmatrix}$	$\|\mathbf{x} - \mathbf{m}_1\| = 5.1296$	1	no	$\mathbf{m}_1 = \begin{bmatrix} -0.75 \\ 3.5 \end{bmatrix}$, $\mathbf{m}_2 = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$
5	$\begin{bmatrix} 5 \\ 1 \end{bmatrix}$	$\mathbf{m}_1 = \begin{bmatrix} -0.75 \\ 3.5 \end{bmatrix}$, $\mathbf{m}_2 = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$	$\|\mathbf{x} - \mathbf{m}_1\| = 6.2700$, $\|\mathbf{x} - \mathbf{m}_2\| = 2.2361$	2	yes	$\mathbf{m}_1 = \begin{bmatrix} -0.75 \\ 3.5 \end{bmatrix}$, $\mathbf{m}_2 = \begin{bmatrix} 3 \\ 0 \end{bmatrix} + 0.5 \times \left(\begin{bmatrix} 5 \\ 1 \end{bmatrix} - \begin{bmatrix} 3 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 4 \\ 0.5 \end{bmatrix}$

Note: In each iteration, the updated cluster centres are used.

After 5 iterations, we have $\mathbf{m}_1 = \begin{bmatrix} -0.75 \\ 3.5 \end{bmatrix}$, $\mathbf{m}_2 = \begin{bmatrix} 4 \\ 0.5 \end{bmatrix}$.

Q4. Basic leader follower algorithm (without normalisation), with $\theta = 3$ and $\eta = 0.5$, is employed to find the cluster centres of the dataset S in Q1. The algorithm is run for 5 iterations and the samples are chosen in the order of x_3, x_1, x_1, x_5, x_6 where the left most data is chosen first.

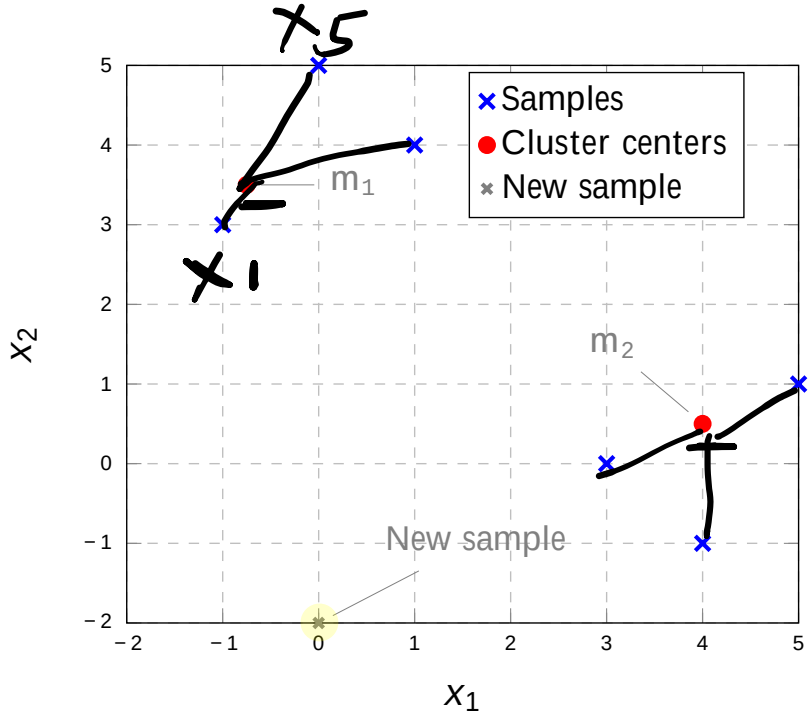
$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$$ ← Dataset

b. Classify the samples in S.

After 5 iterations, we have $m_1 = \begin{bmatrix} -0.75 \\ 3.5 \end{bmatrix}$, $m_2 = \begin{bmatrix} 4 \\ 0.5 \end{bmatrix}$.

These cluster centers are updated one after 5 iterations

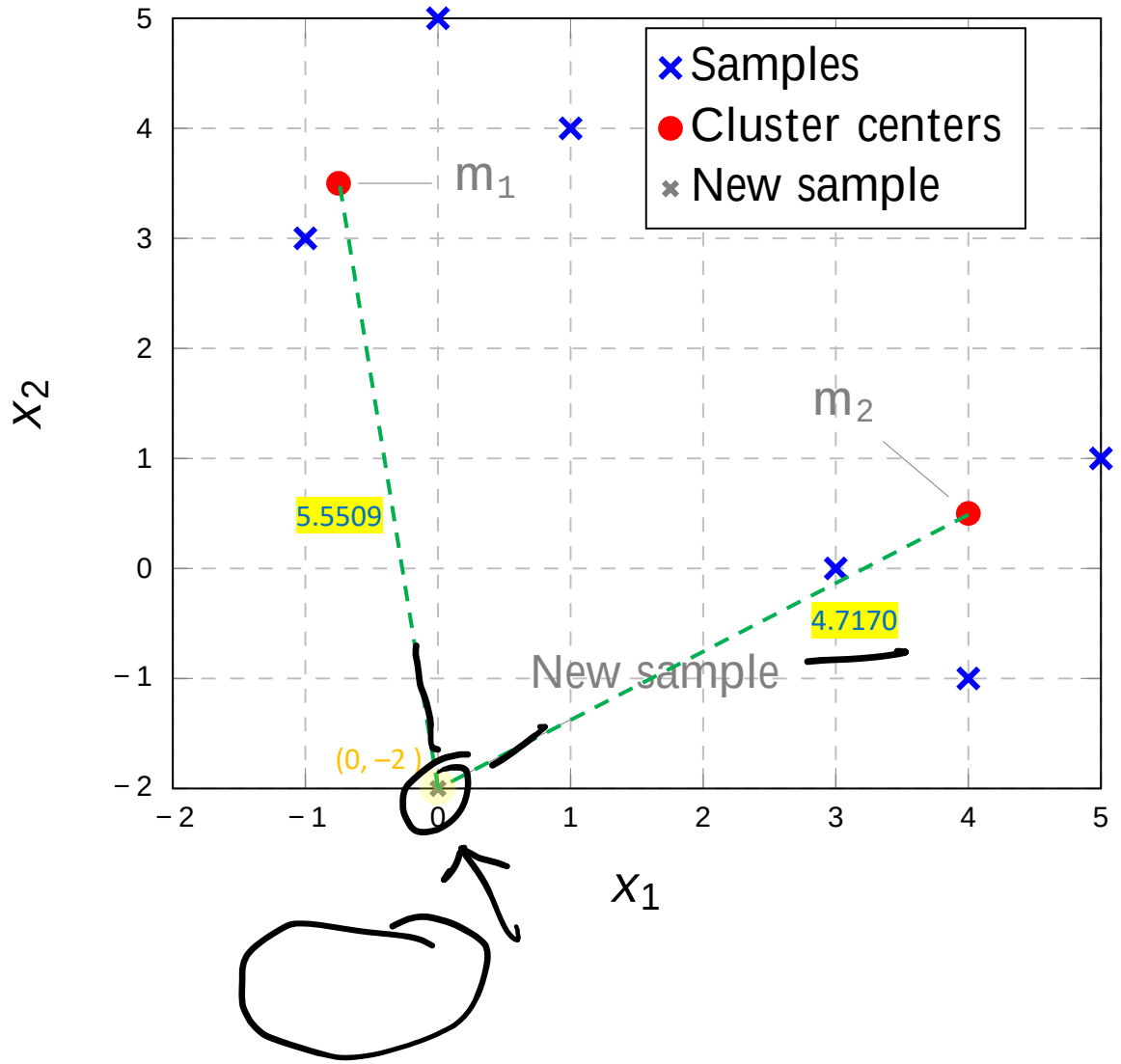
Sample x	$x - m_1$	$x - m_2$	Class
$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	0.5590	5.5902	1
$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$	1.8200	4.6098	1
$\begin{bmatrix} 0 \\ 5 \end{bmatrix}$	1.6771	6.0208	1
$\begin{bmatrix} 4 \\ -1 \end{bmatrix}$	6.5431	1.5000	2
$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	5.1296	1.1180	2
$\begin{bmatrix} 5 \\ 1 \end{bmatrix}$	6.2700	1.1180	2



Q4. Basic leader follower algorithm (without normalisation), with $\theta = 3$ and $\eta = 0.5$, is employed to find the cluster centres of the dataset S in Q1. The algorithm is run for 5 iterations and the samples are chosen in the order of x_3, x_1, x_1, x_5, x_6 where the left most data is chosen first.

c. Classify the new data $x = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$.

For classifying new dataset we use the updated cluster centeres m



Q5. Apply the Fuzzy K-means algorithm, with $K = 2$, to the dataset: $\mathbf{S} = \begin{matrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{matrix}$.

Set parameter $b = 2$, terminate when both coordinates of both cluster centres change by less than 0.5 from one iteration to the next, and start with initial membership values equal to: $\mu = \begin{matrix} 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{matrix}$.

$M = \begin{bmatrix} 0.1234 \\ 0.2345 \end{bmatrix} \neq 1$

Algorithm: Fuzzy K-means clustering

begin initialise n, c ($K = c$), μ_{ij} ($i = 1, 2, \ldots, c$; $j = 1, 2, \ldots, n$), $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

normalize cluster memberships: $\mu_{ij} = \dfrac{\mu_{ij}}{\sum_{r=1}^{c} \mu_{rj}}$

do

update cluster centres: $\mathbf{m}_i = \dfrac{\sum_{j=1}^{n} \mu_{ij}^{b} \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^{b}}$

update memberships: $\mu_{ij} = \dfrac{\left(1/\|\mathbf{x}_j - \mathbf{m}_i\|\right)^{\frac{2}{b-1}}}{\sum_{r=1}^{c} \left(1/\|\mathbf{x}_j - \mathbf{m}_r\|\right)^{\frac{2}{b-1}}}$

until no change in $\mathbf{m}_i$

return $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

end

U_ij is the membership values

Q5. Apply the Fuzzy K-means algorithm, with $K = 2$, to the dataset: $\mathsf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$.

Set parameter $b = 2$, terminate when both coordinates of both cluster centres change by less than 0.5 from one iteration to the next, and start with initial membership values equal to: $\mu = \begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}$.

Algorithm: Fuzzy K-means clustering

begin initialise $n, c\ (K = c), \mu_{ij}\ (i = 1, 2, \ldots, c;\ j = 1, 2, \ldots, n), \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

normalize cluster memberships: $\mu_{ij} = \dfrac{\mu_{ij}}{\sum_{r=1}^{c} \mu_{rj}}$

do

update cluster centres: $\mathbf{m}_i = \dfrac{\sum_{j=1}^{n} \mu_{ij}^b \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^b}$

update memberships: $\mu_{ij} = \dfrac{\left(1/\|\mathbf{x}_j - \mathbf{m}_i\|\right)^{\frac{2}{b-1}}}{\sum_{r=1}^{c} \left(1/\|\mathbf{x}_j - \mathbf{m}_r\|\right)^{\frac{2}{b-1}}}$

until no change in $\mathbf{m}_i$

return $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

end

$$\mu'_{11} = \frac{\mu_{11}}{\mu_{11}+\mu_{21}} = \frac{1}{1+0} = 1$$
$$\mu'_{21} = \frac{\mu_{21}}{\mu_{11}+\mu_{21}} = \frac{0}{1+0} = 0$$
$$\mu'_{11} + \mu'_{21} = 1$$

Dataset is: $\begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 & \mathbf{x}_5 & \mathbf{x}_6 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$.

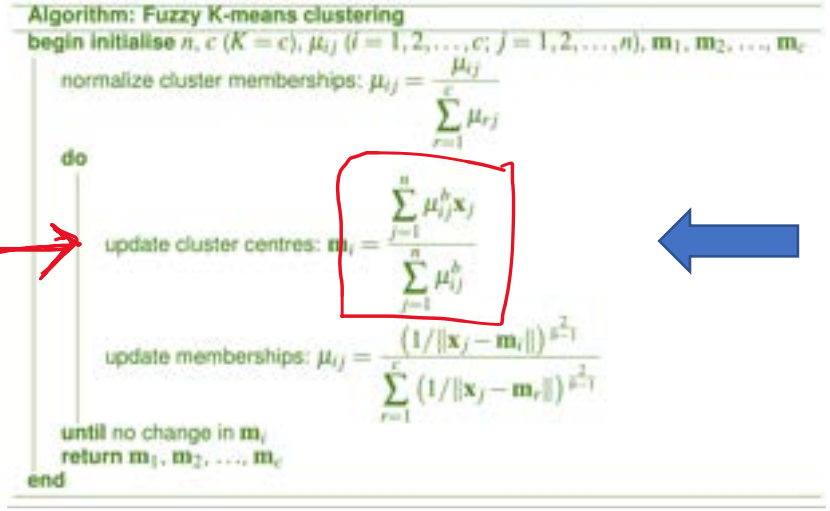
Normalise memberships:

$$\mu = \begin{bmatrix} \mu_{11} & \mu_{12} & \mu_{13} & \mu_{14} & \mu_{15} & \mu_{16} \\ \mu_{21} & \mu_{22} & \mu_{23} & \mu_{24} & \mu_{25} & \mu_{26} \end{bmatrix} = \begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}.$$

Q5. Apply the Fuzzy K-means algorithm, with K = 2, to the dataset: $\mathsf{S} = \begin{matrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{matrix}$.

Set parameter $b = 2$, terminate when both coordinates of both cluster centres change by less than 0.5 from one iteration to the next, and start with initial membership values equal to: $\mu = \begin{matrix} 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{matrix}$.

6 samples



Normalise memberships:

$$\mu = \begin{bmatrix} \mu_{11} & \mu_{12} & \mu_{13} & \mu_{14} & \mu_{15} & \mu_{16} \\ \mu_{21} & \mu_{22} & \mu_{23} & \mu_{24} & \mu_{25} & \mu_{26} \end{bmatrix} = \begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}.$$

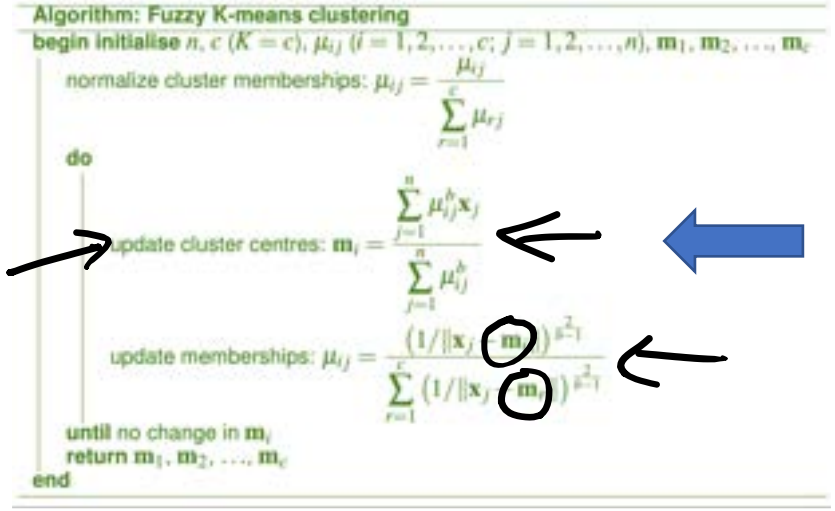
Iteration 1

Update cluster centres:

$$\mathbf{m}_i = \frac{\sum_{j=1}^{n} \mu_{ij}^b \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^b} = \frac{\sum_{j=1}^{6} \mu_{ij}^2 \mathbf{x}_j}{\sum_{j=1}^{6} \mu_{ij}^2} = \frac{\mu_{i1}^2 \mathbf{x}_1 + \mu_{i2}^2 \mathbf{x}_2 + \mu_{i3}^2 \mathbf{x}_3 + \mu_{i4}^2 \mathbf{x}_4 + \mu_{i5}^2 \mathbf{x}_5 + \mu_{i6}^2 \mathbf{x}_6}{\mu_{i1}^2 + \mu_{i2}^2 + \mu_{i3}^2 + \mu_{i4}^2 + \mu_{i5}^2 + \mu_{i6}^2}$$

Q5. Apply the Fuzzy K-means algorithm, with $K = 2$, to the dataset: $S = \begin{matrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{matrix}$.

Set parameter $b = 2$, terminate when both coordinates of both cluster centres change by less than 0.5 from one iteration to the next, and start with initial membership values equal to: $\mu = \begin{matrix} 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{matrix}$.



Normalise memberships:

$$\mu = \begin{bmatrix} \mu_{11} & \mu_{12} & \mu_{13} & \mu_{14} & \mu_{15} & \mu_{16} \\ \mu_{21} & \mu_{22} & \mu_{23} & \mu_{24} & \mu_{25} & \mu_{26} \end{bmatrix} = \begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}.$$

$$\mathbf{m}_i = \frac{\sum_{j=1}^{n} \mu_{ij}^b \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^b} = \frac{\sum_{j=1}^{6} \mu_{ij}^2 \mathbf{x}_j}{\sum_{j=1}^{6} \mu_{ij}^2} = \frac{\mu_{i1}^2 \mathbf{x}_1 + \mu_{i2}^2 \mathbf{x}_2 + \mu_{i3}^2 \mathbf{x}_3 + \mu_{i4}^2 \mathbf{x}_4 + \mu_{i5}^2 \mathbf{x}_5 + \mu_{i6}^2 \mathbf{x}_6}{\mu_{i1}^2 + \mu_{i2}^2 + \mu_{i3}^2 + \mu_{i4}^2 + \mu_{i5}^2 + \mu_{i6}^2}$$

$$\mathbf{m}_1 = \frac{1^2 \begin{bmatrix} -1 \\ 3 \end{bmatrix} + 0.5^2 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 0.5^2 \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.5^2 \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 0.5^2 \begin{bmatrix} 3 \\ 0 \end{bmatrix} + 0^2 \begin{bmatrix} 5 \\ 1 \end{bmatrix}}{1^2 + 0.5^2 + 0.5^2 + 0.5^2 + 0.5^2 + 0^2} = \begin{bmatrix} 0.5 \\ 2.5 \end{bmatrix}$$

$$\mathbf{m}_2 = \frac{0^2 \begin{bmatrix} -1 \\ 3 \end{bmatrix} + 0.5^2 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 0.5^2 \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.5^2 \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 0.5^2 \begin{bmatrix} 3 \\ 0 \end{bmatrix} + 1^2 \begin{bmatrix} 5 \\ 1 \end{bmatrix}}{0^2 + 0.5^2 + 0.5^2 + 0.5^2 + 0.5^2 + 1^2} = \begin{bmatrix} 3.5 \\ 1.5 \end{bmatrix}$$

Q5. Apply the Fuzzy K-means algorithm, with K = 2, to the dataset: $\mathsf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$.
Set parameter $b = 2$, terminate when both coordinates of both cluster centres change by less than 0.5 from one iteration to the next, and start with initial membership values equal to: $\mu = \begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}$.

Normalise memberships:

$$\mu = \begin{bmatrix} \mu_{11} & \mu_{12} & \mu_{13} & \mu_{14} & \mu_{15} & \mu_{16} \\ \mu_{21} & \mu_{22} & \mu_{23} & \mu_{24} & \mu_{25} & \mu_{26} \end{bmatrix} = \begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}.$$

Algorithm: Fuzzy K-means clustering

begin initialise n, c $(K = c)$, μ_{ij} $(i = 1, 2, \ldots, c;\ j = 1, 2, \ldots, n)$, $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

normalize cluster memberships: $\mu_{ij} = \dfrac{\mu_{ij}}{\sum_{r=1}^{c} \mu_{rj}}$

do

update cluster centres: $\mathbf{m}_i = \dfrac{\sum_{j=1}^{n} \mu_{ij}^b \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^b}$

update memberships: $\mu_{ij} = \dfrac{\left(1/\|\mathbf{x}_j - \mathbf{m}_i\|\right)^{\frac{2}{b-1}}}{\sum_{r=1}^{c} \left(1/\|\mathbf{x}_j - \mathbf{m}_r\|\right)^{\frac{2}{b-1}}}$

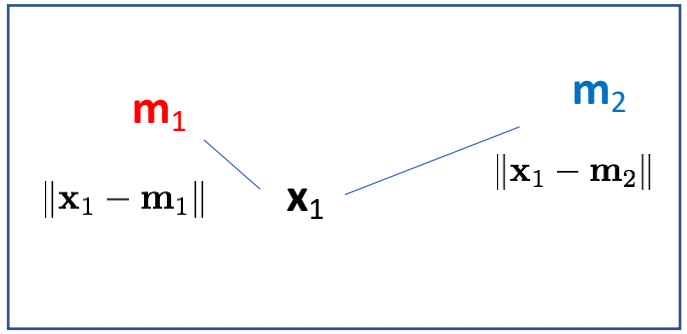
until no change in $\mathbf{m}_i$

return $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

end

Calculate membership values:

$$\mu_{ij} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^{\frac{2}{b-1}}}{\sum_{r=1}^{c} (1/\|\mathbf{x}_j - \mathbf{m}_r\|)^{\frac{2}{b-1}}} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^{2}}{\sum_{r=1}^{2} (1/\|\mathbf{x}_j - \mathbf{m}_r\|)^{2}} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^2}{(1/\|\mathbf{x}_j - \mathbf{m}_1\|)^2 + (1/\|\mathbf{x}_j - \mathbf{m}_2\|)^2}$$



$$\mu_{11} = \frac{(1/\|\mathbf{x}_1 - \mathbf{m}_1\|)^2}{(1/\|\mathbf{x}_1 - \mathbf{m}_1\|)^2 + (1/\|\mathbf{x}_1 - \mathbf{m}_2\|)^2} = \frac{0.4}{0.4 + 0.0444} = 0.9$$

$$\mu_{21} = \frac{(1/\|\mathbf{x}_2 - \mathbf{m}_1\|)^2}{(1/\|\mathbf{x}_1 - \mathbf{m}_2\|)^2 + (1/\|\mathbf{x}_1 - \mathbf{m}_2\|)^2} = \frac{0.0444}{0.4 + 0.0444} = 0.1$$

Algorithm: Fuzzy K-means clustering

begin initialise $n, c\ (K = c), \mu_{ij}\ (i = 1, 2, \ldots, c;\ j = 1, 2, \ldots, n), \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

normalize cluster memberships: $\mu_{ij} = \frac{\mu_{ij}}{\sum_{r=1}^{c} \mu_{rj}}$

do

update cluster centres: $\mathbf{m}_i = \frac{\sum_{j=1}^{n} \mu_{ij}^b \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^b}$

update memberships: $\mu_{ij} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^{\frac{2}{b-1}}}{\sum_{r=1}^{c} (1/\|\mathbf{x}_j - \mathbf{m}_r\|)^{\frac{2}{b-1}}}$

until no change in $\mathbf{m}_i$

return $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

end

M1 = 0.5, 2.5 and m2 = 3.5, 1.5

j	x	$\|\mathbf{x} - \mathbf{m}_1\|$	$\|\mathbf{x} - \mathbf{m}_2\|$	$\left(\frac{1}{\|\mathbf{x}-\mathbf{m}_1\|}\right)^2$	$\left(\frac{1}{\|\mathbf{x}-\mathbf{m}_2\|}\right)^2$	u_{1j}	u_{2j}
1	$\begin{bmatrix}-1\\3\end{bmatrix}$	1.5811	4.7434	0.4	0.0444	0.9	0.1
2	$\begin{bmatrix}1\\4\end{bmatrix}$	1.5811	3.5355	0.4	0.08	0.8333	0.1667
3	$\begin{bmatrix}0\\5\end{bmatrix}$	2.5495	4.9497	0.1538	0.0408	0.7903	0.2097
4	$\begin{bmatrix}4\\-1\end{bmatrix}$	4.9497	2.5495	0.0408	0.1538	0.2097	0.7903
5	$\begin{bmatrix}3\\0\end{bmatrix}$	3.5355	1.5811	0.08	0.4	0.1667	0.8333
6	$\begin{bmatrix}5\\1\end{bmatrix}$	4.7434	1.5811	0.0444	0.4	0.1	0.9

Iteration 2

Normalise memberships:

$$\mu = \begin{bmatrix} \mu_{11} & \mu_{12} & \mu_{13} & \mu_{14} & \mu_{15} & \mu_{16} \\ \mu_{21} & \mu_{22} & \mu_{23} & \mu_{24} & \mu_{25} & \mu_{26} \end{bmatrix} = \begin{bmatrix} 0.9 & 0.8333 & 0.7903 & 0.2097 & 0.1667 & 0.1 \\ 0.1 & 0.1667 & 0.2097 & 0.7903 & 0.8333 & 0.9 \end{bmatrix}.$$

Update cluster centres:

$$\mathbf{m}_i = \frac{\sum_{j=1}^{n} \mu_{ij}^b \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^b} = \frac{\sum_{j=1}^{6} \mu_{ij}^2 \mathbf{x}_j}{\sum_{j=1}^{6} \mu_{ij}^b} = \frac{\mu_{i1}^b \mathbf{x}_1 + \mu_{i2}^b \mathbf{x}_2 + \mu_{i3}^b \mathbf{x}_3 + \mu_{i4}^b \mathbf{x}_4 + \mu_{i5}^b \mathbf{x}_5 + \mu_{i6}^b \mathbf{x}_6}{\mu_{i1}^b + \mu_{i2}^b + \mu_{i3}^b + \mu_{i4}^b + \mu_{i5}^b + \mu_{i6}^b}$$

$$\mathbf{m}_1 = \frac{0.9^2 \begin{bmatrix} -1 \\ 3 \end{bmatrix} + 0.8333^2 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 0.7903^2 \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.2097^2 \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 0.1667^2 \begin{bmatrix} 3 \\ 0 \end{bmatrix} + 0.1^2 \begin{bmatrix} 5 \\ 1 \end{bmatrix}}{0.9^2 + 0.8333^2 + 0.7903^2 + 0.2097^2 + 0.1667^2 + 0.1^2}$$

$$= \begin{bmatrix} 0.0876 \\ 3.7529 \end{bmatrix}$$

$$\mathbf{m}_2 = \frac{0.1^2 \begin{bmatrix} -1 \\ 3 \end{bmatrix} + 0.1667^2 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 0.2097^2 \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.7903^2 \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 0.8333^2 \begin{bmatrix} 3 \\ 0 \end{bmatrix} + 0.9^2 \begin{bmatrix} 5 \\ 1 \end{bmatrix}}{0.1^2 + 0.1667^2 + 0.2097^2 + 0.7903^2 + 0.8333^2 + 0.9^2}$$

$$= \begin{bmatrix} 3.9124 \\ 0.2471 \end{bmatrix}$$

Calculate membership values:

$$\mu_{ij} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^{\frac{2}{b-1}}}{\sum_{r=1}^{c} (1/\|\mathbf{x}_j - \mathbf{m}_r\|)^{\frac{2}{b-1}}} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^2}{\sum_{r=1}^{2} (1/\|\mathbf{x}_j - \mathbf{m}_r\|)^2} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^2}{(1/\|\mathbf{x}_j - \mathbf{m}_1\|)^2 + (1/\|\mathbf{x}_j - \mathbf{m}_2\|)^2}$$

j	$\mathbf{x}$	$\|\mathbf{x}-\mathbf{m}_1\|$	$\|\mathbf{x}-\mathbf{m}_2\|$	$\left(\frac{1}{\|\mathbf{x}-\mathbf{m}_1\|}\right)^2$	$\left(\frac{1}{\|\mathbf{x}-\mathbf{m}_2\|}\right)^2$	u_{1j}	u_{2j}
1	$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	1.3228	5.6312	0.5715	0.0315	0.9477	0.0523
2	$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$	0.9453	4.7504	1.1191	0.0443	0.9619	0.0381
3	$\begin{bmatrix} 0 \\ 5 \end{bmatrix}$	1.2502	6.1560	0.6398	0.0264	0.9604	0.0396
4	$\begin{bmatrix} 4 \\ -1 \end{bmatrix}$	6.1560	1.2502	0.0264	0.6398	0.0396	0.9604
5	$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	4.7504	0.9453	0.0443	1.1191	0.0381	0.9619
6	$\begin{bmatrix} 5 \\ 1 \end{bmatrix}$	5.6312	1.3228	0.0315	0.5715	0.0523	0.9477

Algorithm: Fuzzy K-means clustering

begin initialise n, c ($K = c$), μ_{ij} ($i = 1, 2, \ldots, c$; $j = 1, 2, \ldots, n$), $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

normalize cluster memberships: $\mu_{ij} = \frac{\mu_{ij}}{\sum_{r=1}^{c} \mu_{rj}}$

do

update cluster centres: $\mathbf{m}_i = \frac{\sum_{j=1}^{n} \mu_{ij}^b \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^b}$

update memberships: $\mu_{ij} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^{\frac{2}{b-1}}}{\sum_{r=1}^{c} (1/\|\mathbf{x}_j - \mathbf{m}_r\|)^{\frac{2}{b-1}}}$

until no change in $\mathbf{m}_i$

return $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

end

In second iteration we used the updated cluster from the previous iteration last two columns u_1j and u_2j

In iteration 1:

$$\mathbf{m}_1 = \begin{bmatrix} 0.5 \\ 2.5 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 3.5 \\ 1.5 \end{bmatrix}.$$

In iteration 2:

$$\mathbf{m}_1 = \begin{bmatrix} 0.0876 \\ 3.7529 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 3.9124 \\ 0.2471 \end{bmatrix}.$$

Change is more than 0.5, algorithm does not converge yet.

Iteration 3

Normalise memberships:

$$\mu = \begin{bmatrix} \mu_{11} & \mu_{12} & \mu_{13} & \mu_{14} & \mu_{15} & \mu_{16} \\ \mu_{21} & \mu_{22} & \mu_{23} & \mu_{24} & \mu_{25} & \mu_{26} \end{bmatrix} = \begin{bmatrix} 0.9477 & 0.9619 & 0.9604 & 0.0396 & 0.0381 & 0.0523 \\ 0.0523 & 0.0381 & 0.0396 & 0.9604 & 0.9619 & 0.9477 \end{bmatrix}.$$

Update cluster centres:

$$\mathbf{m}_i = \frac{\sum_{j=1}^{n} \mu_{ij}^b \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^b} = \frac{\sum_{j=1}^{6} \mu_{ij}^2 \mathbf{x}_j}{\sum_{j=1}^{6} \mu_{ij}^b} = \frac{\mu_{i1}^b \mathbf{x}_1 + \mu_{i2}^b \mathbf{x}_2 + \mu_{i3}^b \mathbf{x}_3 + \mu_{i4}^b \mathbf{x}_4 + \mu_{i5}^b \mathbf{x}_5 + \mu_{i6}^b \mathbf{x}_6}{\mu_{i1}^b + \mu_{i2}^b + \mu_{i3}^b + \mu_{i4}^b + \mu_{i5}^b + \mu_{i6}^b}$$

$$\mathbf{m}_1 = \frac{0.9477^2 \begin{bmatrix} -1 \\ 3 \end{bmatrix} + 0.9619^2 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 0.9604^2 \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.0396^2 \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 0.0381^2 \begin{bmatrix} 3 \\ 0 \end{bmatrix} + 0.0523^2 \begin{bmatrix} 5 \\ 1 \end{bmatrix}}{0.9477^2 + 0.9619^2 + 0.9604^2 + 0.0396^2 + 0.0381^2 + 0.0522^2}$$

$$= \begin{bmatrix} 0.0187 \\ 4.0009 \end{bmatrix}$$

$$\mathbf{m}_2 = \frac{0.0523^2 \begin{bmatrix} -1 \\ 3 \end{bmatrix} + 0.0381^2 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 0.0396^2 \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.9604^2 \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 0.9619^2 \begin{bmatrix} 3 \\ 0 \end{bmatrix} + 0.9477^2 \begin{bmatrix} 5 \\ 1 \end{bmatrix}}{0.0523^2 + 0.0381^2 + 0.0396^2 + 0.9604^2 + 0.9619^2 + 0.9477^2}$$

$$= \begin{bmatrix} 3.9813 \\ -0.0009 \end{bmatrix}$$

Final cluster centres: $\mathbf{m}_1 = \begin{bmatrix} 0.0187 \\ 4.0009 \end{bmatrix}$, $\mathbf{m}_2 = \begin{bmatrix} 3.9813 \\ -0.0009 \end{bmatrix}$.

$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $\|\mathbf{x} - \mathbf{m}_1\|$ $\|\mathbf{x} - \mathbf{m}_2\|$

Algorithm: Fuzzy K-means clustering

begin initialise $n, c\ (K = c), \mu_{ij}\ (i = 1, 2, \ldots, c;\ j = 1, 2, \ldots, n), \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

normalize cluster memberships: $\mu_{ij} = \dfrac{\mu_{ij}}{\sum_{r=1}^{c} \mu_{rj}}$

do

update cluster centres: $\mathbf{m}_i = \dfrac{\sum_{j=1}^{n} \mu_{ij}^b \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^b}$

update memberships: $\mu_{ij} = \dfrac{\left(1/\|\mathbf{x}_j - \mathbf{m}_i\|\right)^{\frac{2}{b-1}}}{\sum_{r=1}^{c} \left(1/\|\mathbf{x}_j - \mathbf{m}_r\|\right)^{\frac{2}{b-1}}}$

until no change in $\mathbf{m}_i$

return $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

end

In iteration 2:

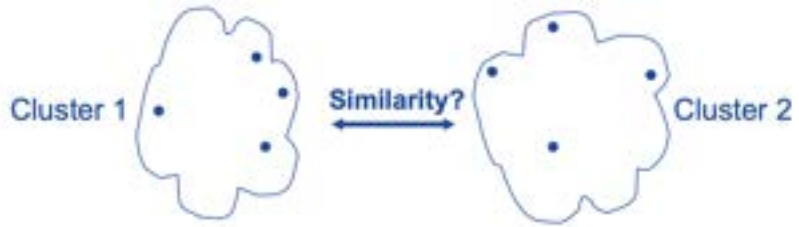
$$\mathbf{m}_1 = \begin{bmatrix} 0.0876 \\ 3.7529 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 3.9124 \\ 0.2471 \end{bmatrix}.$$

In iteration 3: < 0.5

$$\mathbf{m}_1 = \begin{bmatrix} 0.0187 \\ 4.0009 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 3.9813 \\ -0.0009 \end{bmatrix}.$$

Change is less than 0.5, algorithm converges.

Q6. Given the following dataset $\mathbf{x}_1^T = [-1, 3], \mathbf{x}_2^T = [1, 2], \mathbf{x}_3^T = [0, 1], \mathbf{x}_4^T = [4, 0], \mathbf{x}_5^T = [5, 4], \mathbf{x}_6^T = [3, 2]$ perform hierarchical agglomerative clustering until three clusters are formed $(c = 3)$. Use the Euclidean distance as the similarity metric. To determine the distance between clusters use single-link clustering (the minimum distance between samples in the two clusters).



- $d_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\mathbf{x} \in \mathcal{D}_i, \mathbf{x}' = \mathcal{D}_j} \|\mathbf{x} - \mathbf{x}'\|$
  - Single-link clustering - distance between clusters is the minimum distance between constituent samples

Pseudo Code: Algorithm for Agglomerative hierarchical clustering

```
begin initialise c, ĉ ← n, D_i ← {x_i}, i = 1, 2, ..., n
    do ĉ = ĉ − 1
        Find the nearest clusters, say, D_i and D_j
        Merge D_i and D_j
    until c = ĉ
    return c clusters
end
```

c: target number of clusters; $\hat{c}$: current number of clusters.

Initialisation: make each sample a cluster.

Iteration 1

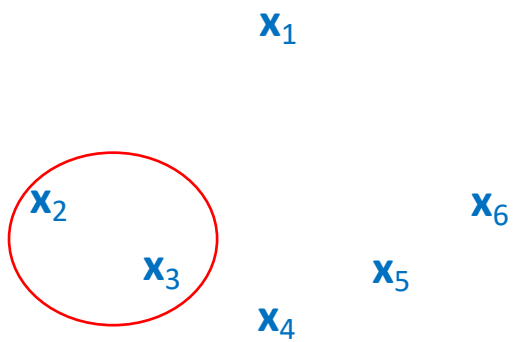
Find most similar clusters:

Calculate the distance between each pair of clusters. The distances can be conveniently written in an array (called the proximity matrix)

$\mathbf{x}^T$	cluster	1	2	3	4	5	6
$[-1, 3])$	1	-	-	-	-	-	-
$[1, 2]$	2	2.2361	-	-	-	-	-
$[0, 1]$	3	2.2361	1.4142	-	-	-	-
$[4, 0]$	4	5.8310	3.6056	4.1231	-	-	-
$[5, 4]$	5	6.0828	4.4721	5.8310	4.1231	-	-
$[3, 2]$	6	4.1231	2.0000	3.1623	2.2361	2.8284	-

Clusters 3 and 2 have minimum distance between them (1.4142).

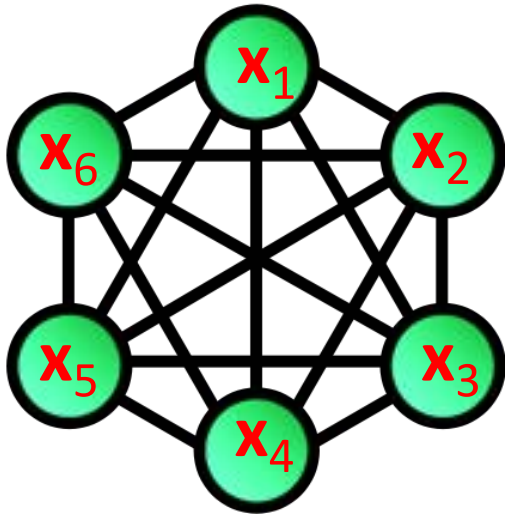
Merge these clusters: clusters 3 and 2 are a new cluster called "2+3"

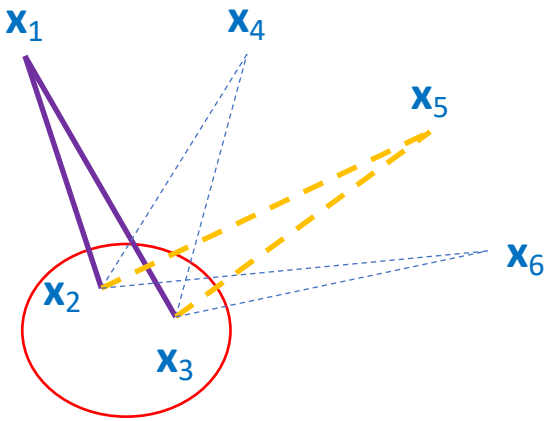


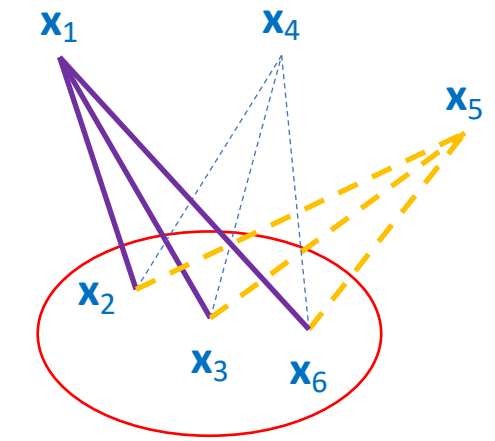
Pseudo Code: Algorithm for Agglomerative hierarchical clustering

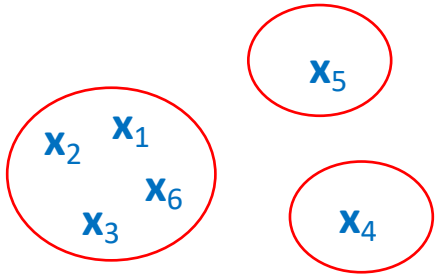
```
begin initialise c, ĉ ← n, 𝒟_i ← {x_i}, i = 1, 2, ..., n
    do ĉ = ĉ − 1
        Find the nearest clusters, say, 𝒟_i and 𝒟_j
        Merge 𝒟_i and 𝒟_j
    until c = ĉ
    return c clusters
end
```

c: target number of clusters; $\hat{c}$: current number of clusters.




x1
x4
x5
x6
x2
x3


x_1
x_4
x_5
x_2
x_3
x_6


x_5
x_2
x_1
x_6
x_3
x_4