1 Introduction

1. Give a definition of "Pattern Recognition".

Pattern Recognition is concerned with creating algorithms that can assign names to observations.
Or,
Pattern Recognition is concerned with making decisions based on data.

2. Which of the following problems is a suitable application for pattern recognition?

a. Classifying numbers into primes and non-primes.

No. Can be more easily and more accurately perfomed analytically.

b. Detecting potential fraud in credit card charges.

Yes.

c. Determining where a cannon ball is likely to land given information about elevation and direction of the barrel, wind direction, etc.

No. Better to do it analytically (e.g. using Newton's equations).

d. Identifying the species a newly discovered "bug" belongs to.

Depends! Some species are defined by simple rules (female mammals produce milk, etc.), so given the right data might easily simply write down the solution. Given other data, like images of different bugs, we would need to perform pattern recognition.

3. Give brief definitions of the following terms:

a. Exemplar

a particular datapoint which is represented by a feature vector (also called an item, sample, instance,...).

b. Dataset

the collection of feature vectors for all exemplars.

c. Generalization

how well a model performs on new data.

d. Overfitting

making the model so specific to the training data that it fails to generalise to new data.

e. Decision Theory

methods for making decisions that reduce cost rather than misclassification rate.

f. Feature Space

the (multidimensional) space defined by the feature vectors in the dataset.

g. Linearly Separable

exemplars from two classes can be separated by a hyperplane in feature space.

h. Dichotomizer

a classifier that places exemplars in one of two classes (also called a binary classifier).

i. Hyper-Parameter

a value used by the learning algorithm in its search for the optimal parameters of the classifier.

j. Grid search

a method of trying to find suitable hyper-parameters that searches all possible combinations of values within defined ranges.

k. Training data

the collection of feature vectors used by the learning algorithm to tune the parameters of the classifier.

l. Test data

the collection of feature vectors used by to evaluate the performance of the trained classifier (this dataset should be distinct from the training data to ensure generalisation).

4. What types of learning best describe the following three scenarios, in which a coin classification system is being created for a vending machine.

a. Measurements of a large number of coins are taken. The algorithm finds that these measurements fall in several "bins". It finds decision boundaries that separate these bins and uses these to classify new coins.

unsupervised learning (clustering)

b. The measurements of each coin is presented to the classifier which makes a decision. The classifier changes its decision boundaries based on whether this decision was correct of incorrect.

reinforcement learning

c. Measurements of a large number of coins are taken. The algorithm uses this data, together with known class labels, to infer decision boundaries which it then uses to classify new coins.

supervised learning (classification)

5. Briefly explain the following types of learning method:

a. Classification

A method that learns to predict a class label associated with each exemplar.

b. Regression

A method that learns to predict a continuous value for each exemplar.

c. Semi-supervised

A method that learns using both labelled and unlabelled training exemplars.

d. Transfer

A method that pre-trains a classifier on another task before training it on the main task in the hope that the pre-training will help improve performance on the main task.

6.

Dataset 1	
Class	**Features**
1	**5**
2	**2**
1	**4**
1	**7**
2	**1**

Dataset 2	
Class	**Features**
1	**5.5**
2	**2.3**
1	**4**
1	**7**
2	**1.8**

Dataset 3	
Class	**Features**
1	**(5,4)**
2	**(2,9)**
1	**(4,3)**
1	**(7,4)**
2	**(1,5)**

Dataset 4	
Class	**Features**
1	**(5.3,4)**
2	**(2.3,9.1)**
1	**(4,3)**
1	**(7,4.1)**
2	**(1.8,5.5)**

Identify which of the above datasets are:

a. univariate-continuous

dataset 2

b. multivariate-discrete

dataset 3

c. multivariate-continuous

dataset 4

d. univariate-discrete

dataset 1

7. A classifier is designed to determine if a feature vector is or is not in a certain class. The output produced by the classifier is 1 if the sample is predicted to be in the class, and 0 otherwise. The following table shows the feature vectors for the samples in the test set, along with the class labels predicted by the classifier and the true class labels of each sample.

Features	Predicted Class	True Class
(5.3,4)	**1**	**1**
(2.3,9.1)	**0**	**1**
(4,3)	**1**	**0**
(7,4.1)	**1**	**1**
(1.8,5.5)	**0**	**0**
(6,3.1)	**1**	**1**
(4.5,3.5)	**0**	**1**

Draw a confusion matrix for this data.

		predicted label	
		true	*false*
true label	*true*	*3*	*2*
	false	*1*	*1*

8. For the results given in the previous question calculate the following performance metrics:

a. the error-rate

$$= \frac{FP+FN}{TP+TN+FP+FN} = \frac{1+2}{3+1+1+2} = \frac{3}{7}$$

b. the accuracy

$$= \frac{TP+TN}{TP+TN+FP+FN} = \frac{3+1}{3+1+1+2} = \frac{4}{7}$$

c. the recall

$$= \frac{TP}{TP+FN} = \frac{3}{3+2} = \frac{3}{5}$$

d. the precision

$$= \frac{TP}{TP+FP} = \frac{3}{3+1} = \frac{3}{4}$$

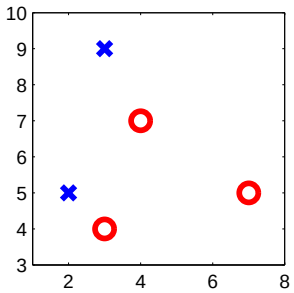
e. the f_1-score

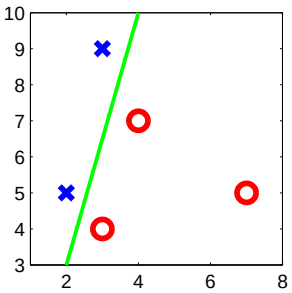
$$= \frac{2\times TP}{2\times TP+FP+FN} = \frac{2\times 3}{2\times 3+1+2} = \frac{6}{9} = \frac{2}{3}$$

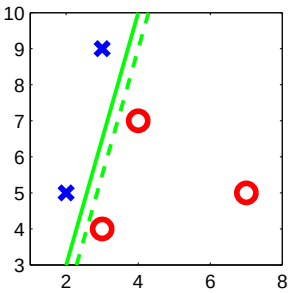
alternatively:f_1-score$= \frac{2\times recall\times precision}{recall+precision} = \frac{2\times\frac{3}{5}\times\frac{3}{4}}{\frac{3}{5}+\frac{3}{4}} = \frac{2\times\frac{9}{20}}{\frac{12}{20}+\frac{15}{20}} = \frac{2\times 9}{27} = \frac{2}{3}$

9. The following table shows exemplars from two classes. Sketch the feature space, and suggest a suitable location for a decision boundary that will minimise the number of mis-classifications. If the cost of erroneously choosing class 1 is higher than the cost of erroneously choosing class two, how will this effect the location of the decision boundary?

Class	Features
1	(3,4)
1	(4,7)
1	(7,5)
2	(2,5)
2	(3,9)







7CCSMPNN: Pattern Recognition, Neural Networks and Deep Learning

KING'S College LONDON

Tutorial - 2: Discriminant functions

Instructors: David Watson, Stefanos Leonardos *Term: Spring 2025 – 20-Jan*

Exercise 1. Consider a dichotomizer defined using the linear discriminant function $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ where $\mathbf{w} = \begin{pmatrix}2\\1\end{pmatrix}$ and $w_0 = -5$. Plot the decision boundary and determine the class of feature vectors: $\mathbf{x}_1 = \begin{pmatrix}1\\1\end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix}2\\2\end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix}3\\3\end{pmatrix}$.

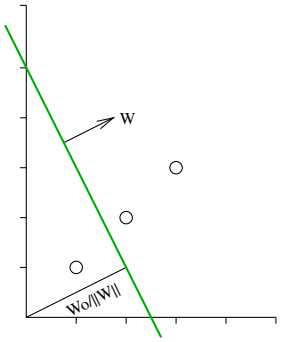
Solution 1. At the decision boundary, it holds that $g(\mathbf{x}) = 0$. Therefore, the decision boundary is defined by

$$\mathbf{w}^T\mathbf{x} + w_0 = 0 \implies (2,1)\cdot\begin{pmatrix}x_1\\x_2\end{pmatrix} - 5 = 0 \implies 2x_1 + x_2 - 5 = 0 \implies x_2 = -2x_1 + 5.$$

In general $x_2 = mx_1 + c = -\frac{w_1}{w_2}x_1 - \frac{w_0}{w_2}$. Since $\mathbf{w}$ has two elements, we know that we are working in a 2D feature space. We also know that we are dealing with a linear discriminant function, so the boundary is a straight line. One way to plot a straight line is by knowing two points on that line. Let's find where this line intercepts the axes.

- x_1-axis $(x_1, 0)$: $0 = -2x_1 + 5 \implies x_1 = -5/-2 = 2.5$.
- x_2-axis $(0, x_2)$: $x_2 = 2\cdot 0 + 5 = 5$.

So, the x_1-axis intercept is at $(2.5, 0)$ and the x_2-axis intercept is at $(0, 5)$.



To classify a point $\mathbf{x}$, we calculate $g(\mathbf{x})$; $\mathbf{x}$ is in class 1 if $g(\mathbf{x}) > 0$:

- $\mathbf{x}_1$: $(2,1)\cdot\begin{pmatrix}1\\1\end{pmatrix} - 5 = 2 + 1 - 5 = -2$. Thus, $\mathbf{x}_1 = \begin{pmatrix}1\\1\end{pmatrix}$ is in class 2.
- $\mathbf{x}_2$: $(2,1)\cdot\begin{pmatrix}2\\2\end{pmatrix} - 5 = 4 + 2 - 5 = 1$. Thus, $\mathbf{x}_2 = \begin{pmatrix}2\\2\end{pmatrix}$ is in class 1.
- $\mathbf{x}_3$: $(2,1)\cdot\begin{pmatrix}3\\3\end{pmatrix} - 5 = 6 + 3 - 5 = 4$. Thus, $\mathbf{x}_3 = \begin{pmatrix}3\\3\end{pmatrix}$ is in class 1.

Note that $\mathbf{w}$, i.e., the vector normal to the hyperplane, points towards class 1 (orientation of the hyperplane), and the distance of the hyperplane from the origin is proportional to w_0 (location of the hyperplane). The value of $g(\mathbf{x})$ provides a measure of how far $\mathbf{x}$ is from the decision boundary. The actual distance is given by $\frac{|g(\mathbf{x})|}{\|\mathbf{w}\|}$.

Exercise 2. In augmented feature space, a dichotomizer is defined using the linear discriminant function $g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$ where $\mathbf{a}^T = (-5, 2, 1)$ and $y = \begin{pmatrix}1\\\mathbf{x}\end{pmatrix}$. Determine the class of feature vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ (as given in Exercise 1).

Solution 2. To classify a point $\mathbf{x}$, we calculate $g(\mathbf{x})$; $\mathbf{x}$ is in class 1 if $g(\mathbf{x}) > 0$:

- $\mathbf{x}_1$: $(-5,2,1)\cdot\begin{pmatrix}1\\1\\1\end{pmatrix} = -5 + 2 + 1 = -2$. Thus, $\mathbf{x}_1 = \begin{pmatrix}1\\1\end{pmatrix}$ is in class 2.
- $\mathbf{x}_2$: $(-5,2,1)\cdot\begin{pmatrix}1\\2\\2\end{pmatrix} = -5 + 4 + 2 = 1$. Thus, $\mathbf{x}_2 = \begin{pmatrix}2\\2\end{pmatrix}$ is in class 1.
- $\mathbf{x}_3$: $(-5,2,1)\cdot\begin{pmatrix}1\\3\\3\end{pmatrix} - 5 = -5 + 6 + 3 = 4$. Thus, $\mathbf{x}_3 = \begin{pmatrix}3\\3\end{pmatrix}$ is in class 1.

Note: the same as in previous exercise, just using augmented vectors.

Exercise 3. Consider a 3-dimensional feature space and the quadratic discriminant function

$$g(\mathbf{x}) = x_1^2 - x_3^2 + 2x_2x_3 + 4x_1x_2 + 3x_1 - 2x_2 + 2,$$

which defines two classes, ω_1, ω_2, such that $g(\mathbf{x}) > 0$ if $x \in \omega_1$ and $g(\mathbf{x}) \leq 0$ if $x \in \omega_2$. Determine the class of the feature vectors: $\mathbf{x}_1 = (1,1,1)^T, \mathbf{x}_2 = (-1,0,3)^T, \mathbf{x_3} = (-1,0,0)^T$.

Solution 3. To classify a point $\mathbf{x}$, we calculate $g(\mathbf{x})$ and use the given rule:

- $\mathbf{x}_1$: $g(\mathbf{x}_1 = (1,1,1)^T) = 1^2 - 1^2 + 2 + 4 + 3 - 2 + 2 = 9$. Thus, $\mathbf{x}_1 = (1,1,1)^T$ is in class 1.
- $\mathbf{x}_2$: $g(\mathbf{x}_2 = (-1,0,3)^T) = (-1)^2 - 3^2 + 0 + 0 - 3 - 0 + 2 = -9$. Thus, $\mathbf{x}_2 = (-1,0,3)^T$ is in class 2.
- $\mathbf{x}_3$: $g(\mathbf{x}_3 = (-1,0,0)^T) = (-1)^2 - 0 + 0 + 0 - 3 - 0 + 2 = 0$. Thus, $\mathbf{x}_3 = (-1,0,0)^T$ is in class 2.

Exercise 4. Consider a dichotomizer defined in a 2-dimensional feature space using a quadratic function

$$g(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{b} + c$$

where $\mathbf{b} = \begin{pmatrix}1\\2\end{pmatrix}$, and $c = -3$. Classify the feature vectors $\mathbf{x}_1 = \begin{pmatrix}0\\-1\end{pmatrix}$, and $\mathbf{x}_2 = \begin{pmatrix}1\\1\end{pmatrix}$, when:

i) $\mathbf{A} = \begin{pmatrix}2 & 1\\1 & 4\end{pmatrix}$, ii) $\mathbf{A} = \begin{pmatrix}-2 & 5\\5 & -8\end{pmatrix}$.

Solution 4. We calculate the function $g(\mathbf{x})$ for both matrices:

i) $\mathbf{A} = \begin{pmatrix}2 & 1\\1 & 4\end{pmatrix}$: $g(\mathbf{x}) = (x_1, x_2)\begin{pmatrix}2 & 1\\1 & 4\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix} + (x_1, x_2)\begin{pmatrix}1\\2\end{pmatrix} - 3$

$$= (2x_1 + x_2, x_1 + 4x_2)\begin{pmatrix}x_1\\x_2\end{pmatrix} + x_1 + 2x_2 - 3$$

$$= 2x_1^2 + x_1x_2 + x_1x_2 + 4x_2^2 + x_1 + 2x_2 - 3 = 2x_1^2 + 4x_2^2 + 2x_1x_2 + x_1 + 2x_2 - 3$$

To classify a point $\mathbf{x}$, we calculate $g(\mathbf{x})$; $\mathbf{x}$ is in class 1 if $g(\mathbf{x}) > 0$:

- $\mathbf{x}_1 = (0,-1)^T$: $g(\mathbf{x}_1) = 0 + 4(-1)^2 + 0 + 0 + 2(-1) - 3 = -1 \leq 0$. Thus, $\mathbf{x}_1$ is in class 2.
- $\mathbf{x}_2 = (1,1)^T$: $g(\mathbf{x}_2) = 2 + 4 + 2 + 1 + 2 - 3 = 8 \geq 0$. Thus, $\mathbf{x}_2$ is in class 1.

ii) $\mathbf{A} = \begin{pmatrix}-2 & 5\\5 & -8\end{pmatrix}$: $g(\mathbf{x}) = (x_1, x_2)\begin{pmatrix}-2 & 5\\5 & -8\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix} + (x_1, x_2)\begin{pmatrix}1\\2\end{pmatrix} - 3$

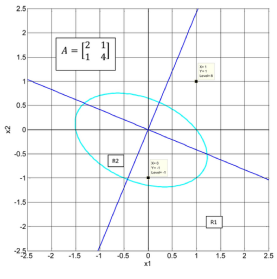
$$= (-2x_1 + 5x_2, 5x_1 - 8x_2)\begin{pmatrix}x_1\\x_2\end{pmatrix} + x_1 + 2x_2 - 3$$

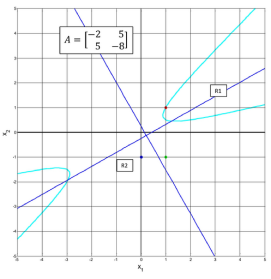
$$= -2x_1^2 + 5x_1x_2 + 5x_1x_2 - 8x_2^2 + x_1 + 2x_2 - 3 = -2x_1^2 - 8x_2^2 + 10x_1x_2 + x_1 + 2x_2 - 3$$

To classify a point $\mathbf{x}$, we calculate $g(\mathbf{x})$; $\mathbf{x}$ is in class 1 if $g(\mathbf{x}) > 0$:

- $\mathbf{x}_1 = (0,-1)^T$: $g(\mathbf{x}_1) = 0 - 8(-1)^2 + 0 + 0 + 2(-1) - 3 = -13 \leq 0$. Thus, $\mathbf{x}_1$ is in class 2.
- $\mathbf{x}_2 = (1,1)^T$: $g(\mathbf{x}_2) = -2 - 8 + 10 + 1 + 2 - 3 = 0 \leq 0$. Thus, $\mathbf{x}_2$ is in class 2.

The decision surfaces look like this:

i) 

ii) 

Exercise 5. In augmented feature space, a dichotomizer is defined using the linear discriminant function $g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$, where $\mathbf{a}^T = (-3,1,2,2,2,4)$, and $y^T = (1, \mathbf{x}^T)$. Classify the feature vectors $\mathbf{x}_1^T = (0,-1,0,0,1)$, and $\mathbf{x}_2^T = (1,1,1,1,1)$.

Solution 5. To classify a point $\mathbf{x}$ we calculate $g(\mathbf{x})$; $\mathbf{x}$ is in class 1 if $g(\mathbf{x}) > 0$:

- $\mathbf{x}_1$: $g(\mathbf{x}_1) = (-3, 1, 2, 2, 2, 4) \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = -3 + 0 - 2 + 0 + 0 + 4 = -1 \leq 0$. Thus, $\mathbf{x}_1$ is in class 2.
- $\mathbf{x}_2$: $g(\mathbf{x}_2) = (-3, 1, 2, 2, 2, 4) \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = -3 + 1 + 2 + 2 + 2 + 4 = 8 \geq 0$. Thus, $\mathbf{x}_2$ is in class 1.

Note: same as part i) of Ex. 4, just using generalised linear discriminant function: $\mathbf{y}^T = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$.

Exercise 6. A linear discriminant function, $g(\mathbf{x})$, is used to define a dichotomizer such that $\mathbf{x}$ is assigned to class 1 if $g(\mathbf{x}) > 0$, and $\mathbf{x}$ is assigned to class 2 otherwise. Use the *batch perceptron learning algorithm* with augmented notation and sample normalisation to find parameters for the linear discriminant function, when the data set is

$\mathbf{x}^T$	$(1,5)$	$(2,5)$	$(4,1)$	$(5,1)$
class	1	1	2	2

Assume initial values $\mathbf{a}^T = (w_0, \mathbf{w}^T) = (-25, 6, 3)$, and use a learning rate of 1.

Solution 6. Using augmented notation and sample normalisation, the data set is

$\mathbf{x}^T$	$(1,5)$	$(2,5)$	$(4,1)$	$(5,1)$
$\mathbf{y}^T$	$(1,1,5)$	$(1,2,5)$	$(-1,-4,-1)$	$(-1,-5,-1)$

For the *batch perceptron learning algorithm*, the weights are updated such that: $\mathbf{a} \leftarrow \mathbf{a} + \eta \sum_{y \in \chi} \mathbf{y}$. Here, $\eta = 1$.

- Epoch 1: initial $\mathbf{a} = (-25, 6, 3)^T$:

$\mathbf{y}^T$	$g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$	misclassified: $g(\mathbf{x}) \leq 0$?
$(1,1,5)$	$(-25 \times 1) + (6 \times 1) + (3 \times 5) = -4$	yes
$(1,2,5)$	$(-25 \times 1) + (6 \times 2) + (3 \times 5) = 2$	no
$(-1,-4,-1)$	$(-25 \times -1) + (6 \times -4) + (3 \times -1) = -2$	yes
$(-1,-5,-1)$	$(-25 \times -1) + (6 \times -5) + (3 \times -1) = -8$	yes

  $\mathbf{a} \leftarrow (-25, 6, 3)^T + (1, 1, 5)^T + (-1, -4, -1)^T + (-1, -5, -1)^T = (-26, -2, 6)^T$.

- Epoch 2: $\mathbf{a} = (-26, -2, 6)^T$:

$\mathbf{y}^T$	$g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$	misclassified: $g(\mathbf{x}) \leq 0$?
$(1,1,5)$	$(-26 \times 1) + (-2 \times 1) + (6 \times 5) = 2$	no
$(1,2,5)$	$(-26 \times 1) + (-2 \times 2) + (6 \times 5) = 0$	yes
$(-1,-4,-1)$	$(-26 \times -1) + (-2 \times -4) + (6 \times -1) = 28$	no
$(-1,-5,-1)$	$(-26 \times -1) + (-2 \times -5) + (6 \times -1) = 30$	no

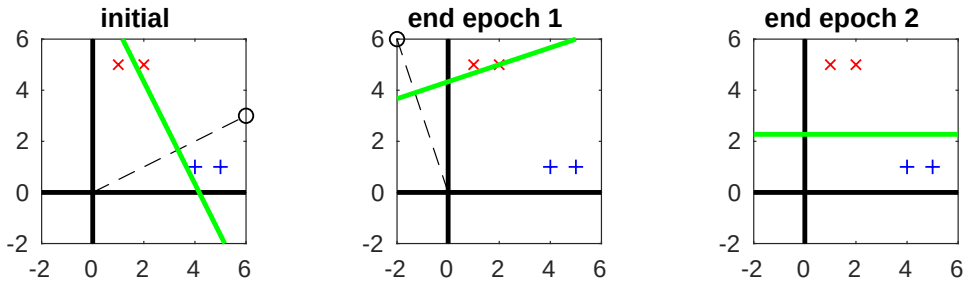
  $\mathbf{a} \leftarrow (-26, -2, 6)^T + (1, 2, 5)^T = (-25, 0, 11)^T$.

- Epoch 3: $\mathbf{a} = (-25, 0, 11)^T$:

$\mathbf{y}^T$	$g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$	misclassified: $g(\mathbf{x}) \leq 0$?
$(1,1,5)$	$(-25 \times 1) + (0 \times 1) + (11 \times 5) = 30$	no
$(1,2,5)$	$(-25 \times 1) + (0 \times 2) + (11 \times 5) = 30$	no
$(-1,-4,-1)$	$(-25 \times -1) + (0 \times -4) + (11 \times -1) = 14$	no
$(-1,-5,-1)$	$(-25 \times -1) + (0 \times -5) + (11 \times -1) = 14$	no

  Learning has converged, so required parameters are $\mathbf{a} = (-25, 0, 11)^T$.

In feature space, the decision surface found at each epoch looks like this:



Exercise 7. Repeat the previous exercise using the *sequential perceptron learning algorithm* with augmented notation and sample normalisation.

Solution 7. For the *sequential perceptron learning algorithm*, weights are updated such that: $\mathbf{a} \leftarrow \mathbf{a} + \eta \mathbf{y}_k$, where $\mathbf{y}_k$ is a misclassified exemplar. Here, $\eta = 1$.

- Epoch 1: initial $\mathbf{a} = (-25, 6, 3)^T$:

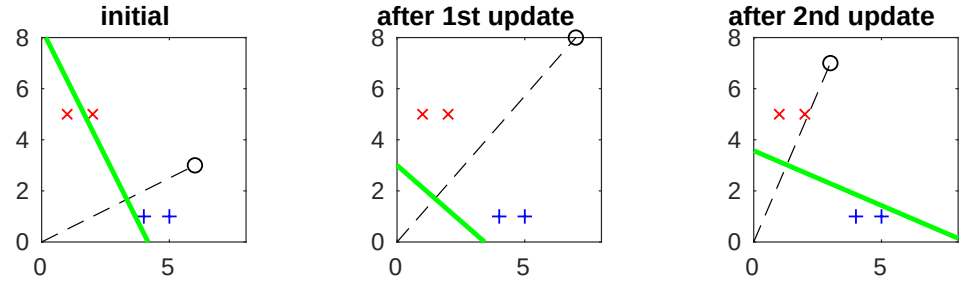
$\mathbf{y}^T$	$g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$	updated $\mathbf{a}^T$
$(1, 1, 5)$	$(-25 \times 1) + (6 \times 1) + (3 \times 5) = -4$	$(-25, 6, 3) + (1, 1, 5) = (-24, 7, 8)$
$(1, 2, 5)$	$(-24 \times 1) + (7 \times 2) + (8 \times 5) = 30$	$(-24, 7, 8)$
$(-1, -4, -1)$	$(-24 \times -1) + (7 \times -4) + (8 \times -1) = -12$	$(-24, 7, 8) + (-1, -4, -1) = (-25, 3, 7)$
$(-1, -5, -1)$	$(-25 \times -1) + (3 \times -5) + (7 \times -1) = 3$	$(-25, 3, 7)$

- Epoch 2: $\mathbf{a} = (-25, 3, 7)^T$:

$\mathbf{y}^T$	$g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$	updated $\mathbf{a}^T$
$(1, 1, 5)$	$(-25 \times 1) + (3 \times 1) + (7 \times 5) = 13$	$(-25, 3, 7)$
$(1, 2, 5)$	$(-25 \times 1) + (3 \times 2) + (7 \times 5) = 16$	$(-25, 3, 7)$
$(-1, -4, -1)$	$(-25 \times -1) + (3 \times -4) + (7 \times -1) = 6$	$(-25, 3, 7)$
$(-1, -5, -1)$	$(-25 \times -1) + (3 \times -5) + (7 \times -1) = 3$	$(-25, 3, 7)$

Learning has converged, so required parameters are $\mathbf{a} = (-25, 3, 7)^T$.

In feature space, the decision surface found at each update looks like this:



Exercise 8. Write pseudo-code for the *sequential perceptron learning algorithm*.

Solution 8. Two equivalent versions of the sequential perceptron learning algorithm are presented below:

Algorithm 1a. Sequential perceptron learning

1: Augment and apply sample normalisation to the feature vectors.
2: Initialise $\mathbf{a}$ and set the learning rate.
3: **for** each sample, $\mathbf{y}_k$, in the data-set: **do**
4: calculate $g(\mathbf{x}_k) = \mathbf{a}^T\mathbf{y}_k$
5: **if** $\mathbf{y}_k$ is misclassified (i.e., if $g(\mathbf{x}_k) \leq 0$), **then**
6: update solution such that: $\mathbf{a} \leftarrow \mathbf{a} + \eta \mathbf{y}_k$
7: Repeat until $\mathbf{a}$ unchanged by all samples (i.e., all samples correctly classified).

Algorithm 1b. Alternative formulation

1: Set $\omega_k = 1$ for samples in class 1, and $\omega_k = -1$ for samples in class 2.
2: Initialise $\mathbf{a}$ and set the learning rate.
3: **for** each sample, $\mathbf{y}_k$, in the data-set: **do**
4: calculate $g(\mathbf{x}_k) = \mathbf{a}^T\mathbf{y}_k$
5: **if** $\mathbf{y}_k$ is misclassified (i.e., if $\text{sign}(g(\mathbf{x}_k)) \neq \omega_k$), **then**
6: update solution such that: $\mathbf{a} \leftarrow \mathbf{a} + \eta\omega_k\mathbf{y}_k$
7: Repeat until $\mathbf{a}$ unchanged by all samples (i.e., all samples correctly classified).

Exercise 9. Consider the linearly separable data set

$\mathbf{x}^T$	$(0,2)$	$(1,2)$	$(2,1)$	$(-3,1)$	$(-2,-1)$	$(-3,-2)$
class (ω)	1	1	1	-1	-1	-1

Apply the *sequential perceptron learning algorithm* to determine the parameters of a linear discriminant function that will correctly classify the data. Assume initial values $\mathbf{a}^T = (w_0, \mathbf{w}^T) = (1,0,0)$, and use a learning rate of 1.

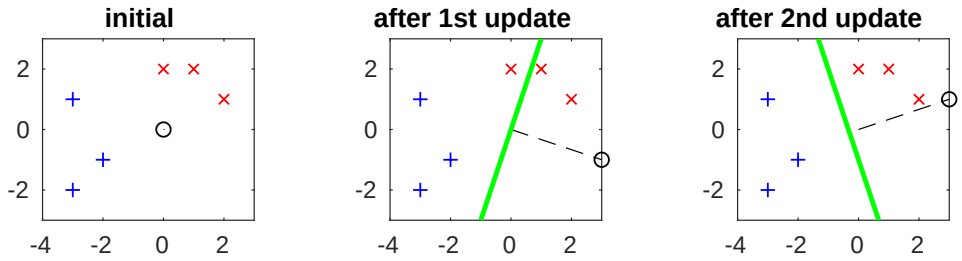
Solution 9. For the *sequential perceptron learning algorithm*, weights are updated such that

$$\mathbf{a}^T_{\text{new}} = \begin{cases} \mathbf{a}^T_{\text{old}} + \eta\omega_k \mathbf{y}^T_k, & \text{if } \mathbf{y}_k \text{ misclassified} \\ \mathbf{a}^T_{\text{old}}, & \text{if } \mathbf{y}_k \text{ correctly classified,} \end{cases}$$

where ω_k is the class label associated with exemplar $\mathbf{y}_k$. Here, $\eta = 1$.

iteration	$\mathbf{a}^T_{\text{old}}$	$\mathbf{y}^T$	$g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$	ω_k	updated: $\mathbf{a}^T_{\text{new}}$
1	$[1,0,0]$	$[1,0,2]$	1	1	$[1,0,0]$
2	$[1,0,0]$	$[1,1,2]$	1	1	$[1,0,0]$
3	$[1,0,0]$	$[1,2,1]$	1	1	$[1,0,0]$
4	$[1,0,0]$	$[1,-3,1]$	1	-1	$[0,3,-1]$
5	$[0,3,-1]$	$[1,-2,-1]$	-5	-1	$[0,3,-1]$
6	$[0,3,-1]$	$[1,-3,-2]$	-7	-1	$[0,3,-1]$
7	$[0,3,-1]$	$[1,0,2]$	-2	1	$[1,3,1]$
8	$[1,3,1]$	$[1,1,2]$	6	1	$[1,3,1]$
9	$[1,3,1]$	$[1,2,1]$	8	1	$[1,3,1]$
10	$[1,3,1]$	$[1,-3,1]$	-7	-1	$[1,3,1]$
11	$[1,3,1]$	$[1,-2,-1]$	-6	-1	$[1,3,1]$
12	$[1,3,1]$	$[1,-3,-2]$	-10	-1	$[1,3,1]$
13	$[1,3,1]$	$[1,0,2]$	3	1	$[1,3,1]$

Learning has converged (we have gone through all the data without needing to update the parameters), so required parameters are $\mathbf{a} = (1,3,1)^T$. In feature space, the decision surface found after each update looks like this:



Exercise 10. Repeat the previous exercise using the sample normalisation method of implementing the *sequential perceptron learning algorithm.*

Solution 10. Using augmented notation and sample normalisation, the data set is:

$\mathbf{x}^T$	$(0,2)$	$(1,2)$	$(2,1)$	$(-3,1)$	$(-2,-1)$	$(-3,-2)$
$\mathbf{y}^T$	$(1,0,2)$	$(1,1,2)$	$(1,2,1)$	$(-1,3,-1)$	$(-1,2,1)$	$(-1,3,2)$

For the *sequential perceptron learning algorithm* with sample normalisation, weights are updated such that:

$$\mathbf{a}^T_{\text{new}} = \begin{cases} \mathbf{a}^T_{\text{old}} + \eta\mathbf{y}^T_k, & \text{if } \mathbf{y}_k \text{ misclassified} \\ \mathbf{a}^T_{\text{old}}, & \text{if } \mathbf{y}_k \text{ correctly classified.} \end{cases}$$

Here, $\eta = 1$.

iteration	$\mathbf{a}_{\text{old}}^T$	$\mathbf{y}^T$	$g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$	updated: $\mathbf{a}_{new}^T$
1	$[1,0,0]$	$[1,0,2]$	1	$[1,0,0]$
2	$[1,0,0]$	$[1,1,2]$	1	$[1,0,0]$
3	$[1,0,0]$	$[1,2,1]$	1	$[1,0,0]$
4	$[1,0,0]$	$[-1,3,-1]$	-1	$[0,3,-1]$
5	$[0,3,-1]$	$[-1,2,1]$	5	$[0,3,-1]$
6	$[0,3,-1]$	$[-1,3,2]$	7	$[0,3,-1]$
7	$[0,3,-1]$	$[1,0,2]$	-2	$[1,3,1]$
8	$[1,3,1]$	$[1,1,2]$	6	$[1,3,1]$
9	$[1,3,1]$	$[1,2,1]$	8	$[1,3,1]$
10	$[1,3,1]$	$[-1,3,-1]$	7	$[1,3,1]$
11	$[1,3,1]$	$[-1,2,1]$	6	$[1,3,1]$
12	$[1,3,1]$	$[-1,3,2]$	10	$[1,3,1]$
13	$[1,3,1]$	$[1,0,2]$	3	$[1,3,1]$

Learning has converged, so required parameters are $\mathbf{a} = (1,3,1)^T$.

Exercise 11. A data set consists of exemplars from three classes

$$\text{Class 1: } \begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}2\\0\end{pmatrix}, \quad \text{Class 2: } \begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}-1\\1\end{pmatrix}, \quad \text{Class 3: } \begin{pmatrix}-1\\-1\end{pmatrix}.$$

Use the *sequential multi-class perceptron learning algorithm* to find the parameters for three linear discriminant functions that will correctly classify this data. Assume that initial values for all parameters are zero, and use a learning rate of 1. Note: if more than one discriminant function produce the maximum output, then choose the function that represents the largest class label.

Solution 11. Pseudo-code for the *sequential multi-class perceptron learning* is provided below

Algorithm 2. Sequential multi-class perceptron learning

1: Initialise $\mathbf{a}_j$ for each class.
2: **for** exemplar $(\mathbf{y}_k, \omega_k)$ **do** in turn:
3: find predicted class $j = \arg\max_{j'}(\mathbf{a}_{j'}^T\mathbf{y}_k)$
4: **if** predicted class is not true class, i.e., $j \neq \omega_k$, **then** update weights:
5: $\mathbf{a}_{\omega_k} = \mathbf{a}_{\omega_k} + \eta\mathbf{y}_k$
6: $\mathbf{a}_j = \mathbf{a}_j - \eta\mathbf{y}_k$
7: Repeat until weights stop changing.

Using augmented notation, the data set is:

$\mathbf{x}^T$	$(1,1,1)$	$(1,2,0)$	$(1,0,2)$	$(1,-1,1)$	$(1,-1,-1)$
ω	1	1	2	2	3

it	old parameters			$\mathbf{y}^T$	$g_i(\mathbf{x}) = \mathbf{a}_i^T\mathbf{y}$			ω	new parameters		
	$\mathbf{a}_1^T$	$\mathbf{a}_2^T$	$\mathbf{a}_3^T$		g_1	g_2	g_3		$\mathbf{a}_1^T$	$\mathbf{a}_2^T$	$\mathbf{a}_3^T$
1	$[0,0,0]$	$[0,0,0]$	$[0,0,0]$	$[1,1,1]$	0	0	0	1	$[1,1,1]$	$[0,0,0]$	$[-1,-1,-1]$
2	$[1,1,1]$	$[0,0,0]$	$[-1,-1,-1]$	$[1,2,0]$	3	0	-3	1	$[1,1,1]$	$[0,0,0]$	$[-1,-1,-1]$
3	$[1,1,1]$	$[0,0,0]$	$[-1,-1,-1]$	$[1,0,2]$	3	0	-3	2	$[0,1,-1]$	$[1,0,2]$	$[-1,-1,-1]$
4	$[0,1,-1]$	$[1,0,2]$	$[-1,-1,-1]$	$[1,-1,1]$	-2	3	-1	2	$[0,1,-1]$	$[1,0,2]$	$[-1,-1,-1]$
5	$[0,1,-1]$	$[1,0,2]$	$[-1,-1,-1]$	$[1,-1,-1]$	0	-1	1	3	$[0,1,-1]$	$[1,0,2]$	$[-1,-1,-1]$
6	$[0,1,-1]$	$[1,0,2]$	$[-1,-1,-1]$	$[1,1,1]$	0	3	-3	1	$[1,2,0]$	$[0,-1,1]$	$[-1,-1,-1]$
7	$[1,2,0]$	$[0,-1,1]$	$[-1,-1,-1]$	$[1,2,0]$	5	-2	-3	1	$[1,2,0]$	$[0,-1,1]$	$[-1,-1,-1]$
8	$[1,2,0]$	$[0,-1,1]$	$[-1,-1,-1]$	$[1,0,2]$	1	2	-3	2	$[1,2,0]$	$[0,-1,1]$	$[-1,-1,-1]$
9	$[1,2,0]$	$[0,-1,1]$	$[-1,-1,-1]$	$[1,-1,1]$	-1	2	-1	2	$[1,2,0]$	$[0,-1,1]$	$[-1,-1,-1]$
10	$[1,2,0]$	$[0,-1,1]$	$[-1,-1,-1]$	$[1,-1,-1]$	-1	0	1	3	$[1,2,0]$	$[0,-1,1]$	$[-1,-1,-1]$
11	$[1,2,0]$	$[0,-1,1]$	$[-1,-1,-1]$	$[1,1,1]$	3	0	-3	1	$[1,2,0]$	$[0,-1,1]$	$[-1,-1,-1]$

Learning has converged, so required parameters are $\mathbf{a}_1 = (1,2,0)^T, \mathbf{a}_2 = (0,-1,1)^T, \mathbf{a}_3 = (-1,-1,-1)^T$.

Exercise 12. Consider the linearly separable data set

$\mathbf{x}^T$	$(0,2)$	$(1,2)$	$(2,1)$	$(-3,1)$	$(-2,-1)$	$(-3,-2)$
class (ω)	1	1	1	-1	-1	-1

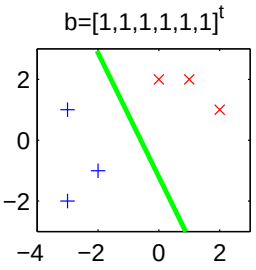
Use the pseudo-inverse (command `pinv` in MATLAB) to calculate the parameters of a linear discriminant function that can be used to classify this data. Use an arbitrary margin vector $\mathbf{b}^T = (1,1,1,1,1,1)$.

Solution 12. Using augmented notation and sample normalisation, the data set is:

$\mathbf{x}^T$	$(0,2)$	$(1,2)$	$(2,1)$	$(-3,1)$	$(-2,-1)$	$(-3,-2)$
$\mathbf{y}^T$	$(1,0,2)$	$(1,1,2)$	$(1,2,1)$	$(-1,3,-1)$	$(-1,2,1)$	$(-1,3,2)$

We will find $\mathbf{a}$ through the equation $\mathbf{Ya} = \mathbf{b}$ where $\mathbf{Y} = \begin{pmatrix} 1 & 0 & 2 \\ 1 & 1 & 2 \\ 1 & 2 & 1 \\ -1 & 3 & -1 \\ -1 & 2 & 1 \\ -1 & 3 & 2 \end{pmatrix}$. To find the pseudo-inverse $\mathbf{Y}^\dagger$ of $\mathbf{Y}$, we will use the MATLAB command `pinv`: $\mathbf{Y}^\dagger = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T = \begin{pmatrix} 0.0682 & 0.1648 & 0.3807 & 0.1023 & -0.2330 & -0.2557 \\ -0.0341 & 0.0426 & 0.1847 & 0.1989 & -0.0085 & 0.0028 \\ 0.1402 & 0.0748 & -0.1203 & -0.2064 & 0.1184 & 0.1828 \end{pmatrix}$.
Thus,

$$\mathbf{a} = \mathbf{Y}^\dagger\mathbf{b} = \begin{pmatrix} 0.0682 & 0.1648 & 0.3807 & 0.1023 & -0.2330 & -0.2557 \\ -0.0341 & 0.0426 & 0.1847 & 0.1989 & -0.0085 & 0.0028 \\ 0.1402 & 0.0748 & -0.1203 & -0.2064 & 0.1184 & 0.1828 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.2273 \\ 0.3864 \\ 0.1894 \end{pmatrix}.$$



Check: $g(\mathbf{x}) = \mathbf{a}^T\mathbf{y} = (0.2273, 0.3864, 0.1894)\mathbf{y} \geq 0$?

$\mathbf{y}^T$	$(1,0,2)$	$(1,1,2)$	$(1,2,1)$	$(-1,3,-1)$	$(-1,2,1)$	$(-1,3,2)$
$g(\mathbf{x})$	0.6061	0.9924	1.1894	0.7424	0.7348	1.3106

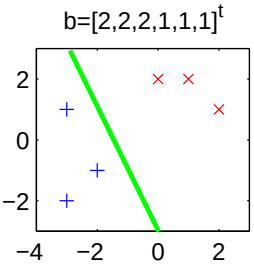
All positive, so discriminant function provides correct classification. In feature space, the decision surface is shown in the figure on the right:

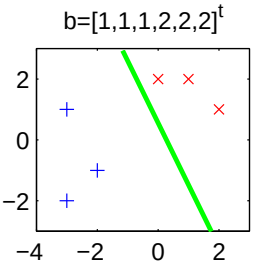
Exercise 13. Repeat the previous exercise using i) $\mathbf{b}^T = (2,2,2,1,1,1)$, and ii) $\mathbf{b}^T = (1,1,1,2,2,2)$.

Solution 13. Using the expression for $\mathbf{Y}^\dagger = \begin{pmatrix} 0.0682 & 0.1648 & 0.3807 & 0.1023 & -0.2330 & -0.2557 \\ -0.0341 & 0.0426 & 0.1847 & 0.1989 & -0.0085 & 0.0028 \\ 0.1402 & 0.0748 & -0.1203 & -0.2064 & 0.1184 & 0.1828 \end{pmatrix}$ that we found in the previous exercise, we have that

i) $\mathbf{a} = \mathbf{Y}^\dagger \begin{pmatrix} 2 \\ 2 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.8409 \\ 0.5795 \\ 0.2841 \end{pmatrix}$, ii) $\mathbf{a} = \mathbf{Y}^\dagger \begin{pmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \end{pmatrix} = \begin{pmatrix} -0.1591 \\ 0.5795 \\ 0.2841 \end{pmatrix}$.

In feature space, the decision surfaces look as in the Figures on the right. Each element of $\mathbf{b}$ corresponds to a data point. Increasing the value of $\mathbf{b}$ increases the margin (i.e., the distance) of the hyperplane from the corresponding data point.





Exercise 14. For the same data set as in the previous exercise, apply 12 iterations of the *sequential Widrow-Hoff learning algorithm*. Assume initial values $\mathbf{a}^T = (w_0, \mathbf{w}^T) = (1, 0, 0)$, and use a margin vector $\mathbf{b}^T = (1, 1, 1, 1, 1, 1)$, and a learning rate of 0.1.

Solution 14. Using augmented notation and sample normalisation, the data set is:

$\mathbf{x}^T$	$(0,2)$	$(1,2)$	$(2,1)$	$(-3,1)$	$(-2,-1)$	$(-3,-2)$
$\mathbf{y}^T$	$(1,0,2)$	$(1,1,2)$	$(1,2,1)$	$(-1,3,-1)$	$(-1,2,1)$	$(-1,3,2)$

For the *sequential Widrow-Hoff learning algorithm*, the weights are updated such that: $\mathbf{a} \leftarrow \mathbf{a} + \eta(b_k - \mathbf{a}^T\mathbf{y}_k)\mathbf{y}_k$. Here, $\eta = 0.1$.

it	$\mathbf{a}^T$	$\mathbf{y}_k^T$	$\mathbf{a}^T\mathbf{y}_k$	$\mathbf{a}_{\text{new}}^T = \mathbf{a}^T + 0.1(b_k - \mathbf{a}^T\mathbf{y}_k)\mathbf{y}_k^T$
1	$[1,0,0]$	$[1,0,2]$	1	$[1,0,0] + 0.1(1-1)[1,0,2] = [1,0,0]$
2	$[1,0,0]$	$[1,1,2]$	1	$[1,0,0] + 0.1(1-1)[1,1,2] = [1,0,0]$
3	$[1,0,0]$	$[1,2,1]$	1	$[1,0,0] + 0.1(1-1)[1,2,1] = [1,0,0]$
4	$[1,0,0]$	$[-1,3,-1]$	-1	$[1,0,0] + 0.1(1-(-1))[-1,3,-1] = [0.8,0.6,-0.2]$
5	[0.8,0.6,−0.2]	$[-1,2,1]$	0.2	[0.8,0.6,−0.2]+0.1(1−0.2)[−1,2,1]=[0.72,0.76,−0.12]
6	[0.72,0.76,−0.12]	$[-1,3,2]$	1.32	[0.72,0.76,−0.12]+0.1(1−1.32)[−1,3,2]=[0.752,0.664,−0.184]
7	[0.752,0.664,−0.184]	$[1,0,2]$	0.384	[0.752,0.664,−0.184]+0.1(1−0.384)[[1,0,2]=[0.8136,0.6640,−0.0608]
8	[0.8136,0.6640,−0.0608]	$[1,1,2]$	1.356	[0.8136,0.6640,−0.0608]+0.1(1−1.356)[1,1,2]=[0.778,0.6284,−0.1320]
9	[0.778,0.6284,−0.1320]	$[1,2,1]$	1.9028	[0.778,0.6284,−0.1320]+0.1(1−1.9028)[1,2,1]=[0.6877,0.4478,−0.2223]
10	[0.6877,0.4478,−0.2223]	$[-1,3,-1]$	0.8781	[0.6877,0.4478,−0.2223]+0.1(1−0.8781)[−1,3,−1]=[0.6755,0.4844,−0.2345]
11	[0.6755,0.4844,−0.2345]	$[-1,2,1]$	0.0588	[0.6755,0.4844,−0.2345]+0.1(1−0.0588)[−1,2,1]=[0.5814,0.6726,−0.1404]
12	[0.5814,0.6726,−0.1404]	$[-1,3,2]$	1.1558	[0.5814,0.6726,−0.1404]+0.1(1−1.1558)[−1,3,2]=[0.597,0.6259,−0.1715]

Exercise 15. The table shows the training data set for a simple classification problem

feature vector ($\mathbf{x}^T$)	$(0.15, 0.35)$	$(0.15, 0.28)$	$(0.12, 0.2)$	$(0.1, 0.32)$	$(0.06, 0.25)$
class (ω)	1	2	2	3	3

Apply the *k-nearest-neighbour* classifier with Euclidean distance and i) $k = 1$, ii) $k = 3$, to determine the class of a new feature vector $\mathbf{x} = (0.1, 0.25)$. How would these results be affected if the first dimension of the feature space was scaled by a factor of two?

Solution 15. First, we calculate the distance of each sample to $\mathbf{x}$:

class	feature vector	Euclidean distance to (0.1,0.25)
1	(0.15,0.35)	$\sqrt{(0.1-0.15)^2 + (0.25-0.35)^2} = 0.1118$
2	(0.15,0.28)	$\sqrt{(0.1-0.15)^2 + (0.25-0.28)^2} = 0.0583$
2	(0.12,0.20)	$\sqrt{(0.1-0.12)^2 + (0.25-0.20)^2} = 0.0539$
3	(0.10,0.32)	$\sqrt{(0.1-0.10)^2 + (0.25-0.32)^2} = 0.0700$
3	(0.06,0.25)	$\sqrt{(0.1-0.06)^2 + (0.25-0.25)^2} = 0.0400$

i) The nearest neighbour has class label 3. Thus, class the new sample as 3.
ii) The three nearest neighbours have class labels 3, 2, and 2. Thus, class the new sample as 2.

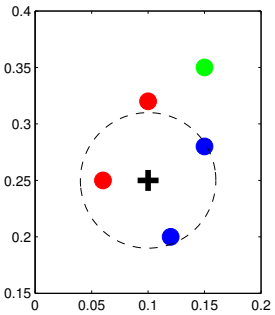
If we scale the first dimension by a factor of two:

class	feature vector	Euclidean distance to (0.2,0.25)
1	(0.30,0.35)	$\sqrt{(0.2-0.30)^2 + (0.25-0.35)^2} = 0.1414$
2	(0.30,0.28)	$\sqrt{(0.2-0.30)^2 + (0.25-0.28)^2} = 0.1044$
2	(0.24,0.20)	$\sqrt{(0.2-0.24)^2 + (0.25-0.20)^2} = 0.0640$
3	(0.20,0.32)	$\sqrt{(0.2-0.20)^2 + (0.25-0.32)^2} = 0.0700$
3	(0.12,0.25)	$\sqrt{(0.2-0.12)^2 + (0.25-0.25)^2} = 0.0800$

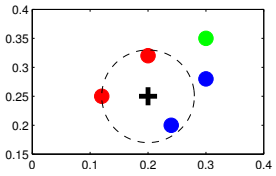
i) For $k = 1$, class of new sample is 2.
ii) For $k = 3$, class of new sample is 3.

This is the opposite of the result we got previously. kNN is very sensitive to the scale of the feature dimensions! The feature spaces before and after scaling look like this

before scaling:



after scaling:



Exercise 16. Consider the data set

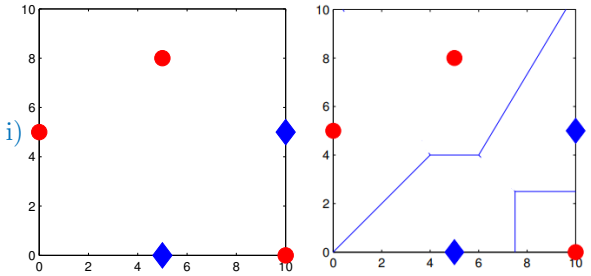
feature vector ($\mathbf{x}^T$)	$(0,5)$	$(5,8)$	$(10,0)$	$(5,0)$	$(10,5)$
class (ω)	1	1	1	2	2

Assuming Euclidean distance, plot the decision boundaries that result from applying

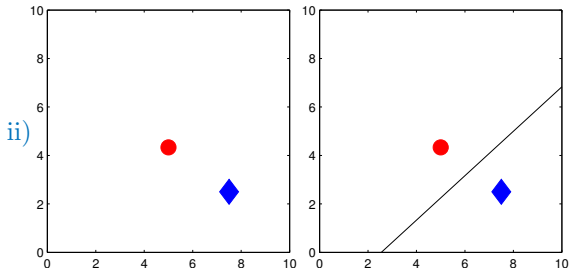
i) a *nearest neighbour classifier*, i.e., a kNN classifier with $k = 1$,
ii) a *nearest mean classifier*, i.e., one in which a new feature vector, $\mathbf{x}$, is classified by assigning it to the same category as the nearest sample mean.

Solution 16.

i)



ii)



3 Neural Networks

1. Give brief definitions of the following terms:

- **neuron**

  an information processing unit.

- **action potential**

  the signal outputted by a biological neuron.

- **firing rate**

  the number of action potentials emitted during a defined time-period.

- **synapse**

  the connection between two neurons.

- **an artificial neural network**

  a parallel architecture composed of many simple processing elements interconnected to achieve certain collective computational capabilities.

2. A neuron has a transfer function which is a linear weighted sum of its inputs and an activation function that is the Heaviside function. If the weights are $\mathbf{w} = [0.1, -0.5, 0.4]$, and the threshold zero, what is the output of this neuron when the input is: $\mathbf{x}_1 = [0.1, -0.5, 0.4]^t$ and $\mathbf{x}_2 = [0.1, 0.5, 0.4]^t$?

Output of neuron is defined as:

$$y = H(\mathbf{w}\mathbf{x} - \theta)$$

$$y_1 = H(\mathbf{w}\mathbf{x}_1 - \theta) = H\left((0.1 \times 0.1) + (-0.5 \times -0.5) + (0.4 \times 0.4) - 0\right) = H(0.42) = 1$$

$$y_2 = H(\mathbf{w}\mathbf{x}_2 - \theta) = H\left((0.1 \times 0.1) + (-0.5 \times 0.5) + (0.4 \times 0.4) - 0\right) = H(-0.08) = 0$$

3. A Linear Threshold Unit has one input, x_1, that can take binary values. Apply the sequential Delta learning rule so that the output of this neuron, y, is equal to $\bar{x_1}$ (or NOT(x_1)), *i.e.*, such that:

x_1	y
0	1
1	0

Assume initial values of $\theta = 1.5$ and $w_1 = 2$, and use a learning rate of 1.

Using Augmented notation, $y = H(\mathbf{w}\mathbf{x})$ where $\mathbf{w} = [-\theta, w_1]$, and $\mathbf{x} = [1, x_1]^T$.

For the Delta rule, weights are updated such that: $\mathbf{w} \leftarrow \mathbf{w} + \eta(t-y)\mathbf{x}^t$

Initial $\mathbf{w} = [-1.5, 2]$

$\mathbf{x}^t$	t	$y = H(\mathbf{wx})$	$t-y$	$\eta(t-y)\mathbf{x}^t$	$\mathbf{w}$
$(1,0)$	1	$H(-1.5\times 1+2\times 0) = H(-1.5) = 0$	1	$[1,0]$	$[-0.5,2]$
$(1,1)$	0	$H(-0.5\times 1+2\times 1) = H(1.5) = 1$	-1	$[-1,-1]$	$[-1.5,1]$
$(1,0)$	1	$H(-1.5\times 1+1\times 0) = H(-1.5) = 0$	1	$[1,0]$	$[-0.5,1]$
$(1,1)$	0	$H(-0.5\times 1+1\times 1) = H(0.5) = 1$	-1	$[-1,-1]$	$[-1.5,0]$
$(1,0)$	1	$H(-1.5\times 1+0\times 0) = H(-1.5) = 0$	1	$[1,0]$	$[-0.5,0]$
$(1,1)$	0	$H(-0.5\times 1+0\times 1) = H(-0.5) = 0$	0	$[0,0]$	$[-0.5,0]$
$(1,0)$	1	$H(-0.5\times 1+0\times 0) = H(-0.5) = 0$	1	$[1,0]$	$[0.5,0]$
$(1,1)$	0	$H(0.5\times 1+0\times 1) = H(0.5) = 1$	-1	$[-1,-1]$	$[-0.5,-1]$
$(1,0)$	1	$H(-0.5\times 1-1\times 0) = H(-0.5) = 0$	1	$[1,0]$	$[0.5,-1]$
$(1,1)$	0	$H(0.5\times 1+-1\times 1) = H(-0.5) = 0$	0	$[0,0]$	$[0.5,-1]$
$(1,0)$	1	$H(0.5\times 1-1\times 0) = H(0.5) = 1$	0	$[0,0]$	$[0.5,-1]$

Learning has converged, so required weights are $\mathbf{w} = [0.5, -1]$*, or equivalently* $\theta = -0.5$*,* $w_1 = -1$*.*

4. Repeat the above question using the batch Delta learning rule.

Epoch 1, initial $\mathbf{w} = [-1.5, 2]$

$\mathbf{x}^t$	t	$y = H(\mathbf{wx})$	$t-y$	$\eta(t-y)\mathbf{x}^t$	$\mathbf{w}$
$(1,0)$	1	$H(-1.5\times 1+2\times 0) = H(-1.5) = 0$	1	$[1,0]$	
$(1,1)$	0	$H(-1.5\times 1+2\times 1) = H(0.5) = 1$	-1	$[-1,-1]$	
		total weight change		$[0,-1]$	$[-1.5,1]$

Epoch 2, initial $\mathbf{w} = [-1.5, 1]$

$\mathbf{x}^t$	t	$y = H(\mathbf{wx})$	$t-y$	$\eta(t-y)\mathbf{x}^t$	$\mathbf{w}$
$(1,0)$	1	$H(-1.5\times 1+1\times 0) = H(-1.5) = 0$	1	$[1,0]$	
$(1,1)$	0	$H(-1.5\times 1+1\times 1) = H(-0.5) = 0$	0	$[0,0]$	
		total weight change		$[1,0]$	$[-0.5,1]$

Epoch 3, initial $\mathbf{w} = [-0.5, 1]$

$\mathbf{x}^t$	t	$y = H(\mathbf{wx})$	$t-y$	$\eta(t-y)\mathbf{x}^t$	$\mathbf{w}$
$(1,0)$	1	$H(-0.5\times 1+1\times 0) = H(-0.5) = 0$	1	$[1,0]$	
$(1,1)$	0	$H(-0.5\times 1+1\times 1) = H(0.5) = 1$	-1	$[-1,-1]$	
		total weight change		$[0,-1]$	$[-0.5,0]$

Epoch 4, initial $\mathbf{w} = [-0.5, 0]$

$\mathbf{x}^t$	t	$y = H(\mathbf{wx})$	$t-y$	$\eta(t-y)\mathbf{x}^t$	$\mathbf{w}$
$(1,0)$	1	$H(-0.5\times 1+0\times 0) = H(-0.5) = 0$	1	$[1,0]$	
$(1,1)$	0	$H(-0.5\times 1+0\times 1) = H(-0.5) = 0$	0	$[0,0]$	
		total weight change		$[1,0]$	$[0.5,0]$

Epoch 5, initial $\mathbf{w} = [0.5, 0]$

$\mathbf{x}^t$	t	$y = H(\mathbf{wx})$	$t - y$	$\eta(t-y)\mathbf{x}^t$	$\mathbf{w}$
$(1,0)$	1	$H(0.5 \times 1 + 0 \times 0) = H(0.5) = 1$	0	$[0,0]$	
$(1,1)$	0	$H(0.5 \times 1 + 0 \times 1) = H(0.5) = 1$	-1	$[-1,-1]$	
			total weight change	$[-1,-1]$	$[-0.5,-1]$

Epoch 6, initial $\mathbf{w} = [-0.5, -1]$

$\mathbf{x}^t$	t	$y = H(\mathbf{wx})$	$t - y$	$\eta(t-y)\mathbf{x}^t$	$\mathbf{w}$
$(1,0)$	1	$H(-0.5 \times 1 - 1 \times 0) = H(-0.5) = 0$	1	$[1,0]$	
$(1,1)$	0	$H(-0.5 \times 1 - 1 \times 1) = H(-1.5) = 0$	0	$[0,0]$	
			total weight change	$[1,0]$	$[0.5,-1]$

Epoch 7, initial $\mathbf{w} = [0.5, -1]$

$\mathbf{x}^t$	t	$y = H(\mathbf{wx})$	$t - y$	$\eta(t-y)\mathbf{x}^t$	$\mathbf{w}$
$(1,0)$	1	$H(0.5 \times 1 - 1 \times 0) = H(0.5) = 1$	0	$[0,0]$	
$(1,1)$	0	$H(0.5 \times 1 - 1 \times 1) = H(-0.5) = 0$	0	$[0,0]$	
			total weight change	$[0,0]$	$[0.5,-1]$

Learning has converged, so required weights are $\mathbf{w} = [0.5, -1]$*, or equivalently* $\theta = -0.5$*,* $w_1 = -1$*.*

5. A Linear Threshold Unit has two inputs, x_1 **and** x_2**, that can take binary values. Apply the sequential Delta learning rule so that the output of this neuron,** y**, is equal to** x_1**AND** x_2**,** *i.e.***, such that:**

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Assume initial values of $\theta = -0.5$**,** $w_1 = 1$ **and** $w_2 = 1$**, and use a learning rate of 1.**

Using Augmented notation, $y = H(\mathbf{wx})$ *where* $\mathbf{w} = [-\theta, w_1, w_2]$*, and* $\mathbf{x} = [1, x_1, x_2]^t$*.*

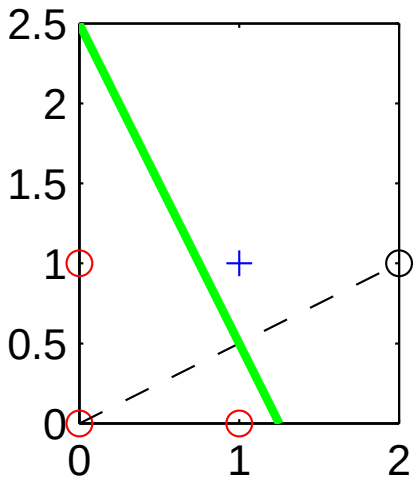
For the Delta rule, weights are updated such that: $\mathbf{w} \leftarrow \mathbf{w} + \eta(t-y)\mathbf{x}^t$

Initial w=[0.5,1,1]

$\mathbf{x}^t$	t	$y = H(\mathbf{wx})$	$t-y$	$\eta(t-y)\mathbf{x}^t$	$\mathbf{w}$
$(1,0,0)$	0	$H(0.5\times1+1\times0+1\times0)=H(0.5)=1$	-1	$[-1,0,0]$	$[-0.5,1,1]$
$(1,0,1)$	0	$H(-0.5\times1+1\times0+1\times1)=H(0.5)=1$	-1	$[-1,0,-1]$	$[-1.5,1,0]$
$(1,1,0)$	0	$H(-1.5\times1+1\times1+0\times0)=H(-0.5)=0$	0	$[0,0,0]$	$[-1.5,1,0]$
$(1,1,1)$	1	$H(-1.5\times1+1\times1+0\times1)=H(-0.5)=0$	1	$[1,1,1]$	$[-0.5,2,1]$
$(1,0,0)$	0	$H(-0.5\times1+2\times0+1\times0)=H(-0.5)=0$	0	$[0,0,0]$	$[-0.5,2,1]$
$(1,0,1)$	0	$H(-0.5\times1+2\times0+1\times1)=H(0.5)=1$	-1	$[-1,0,-1]$	$[-1.5,2,0]$
$(1,1,0)$	0	$H(-1.5\times1+2\times1+0\times0)=H(0.5)=1$	-1	$[-1,-1,0]$	$[-2.5,1,0]$
$(1,1,1)$	1	$H(-2.5\times1+1\times1+0\times1)=H(-1.5)=0$	1	$[1,1,1]$	$[-1.5,2,1]$
$(1,0,0)$	0	$H(-1.5\times1+2\times0+1\times0)=H(-1.5)=0$	0	$[0,0,0]$	$[-1.5,2,1]$
$(1,0,1)$	0	$H(-1.5\times1+2\times0+1\times1)=H(-0.5)=0$	0	$[0,0,0]$	$[-1.5,2,1]$
$(1,1,0)$	0	$H(-1.5\times1+2\times1+1\times0)=H(0.5)=1$	-1	$[-1,-1,0]$	$[-2.5,1,1]$
$(1,1,1)$	1	$H(-2.5\times1+1\times1+1\times1)=H(-0.5)=0$	1	$[1,1,1]$	$[-1.5,2,2]$
$(1,0,0)$	0	$H(-1.5\times1+2\times0+2\times0)=H(-1.5)=0$	0	$[0,0,0]$	$[-1.5,2,2]$
$(1,0,1)$	0	$H(-1.5\times1+2\times0+2\times1)=H(0.5)=1$	-1	$[-1,0,-1]$	$[-2.5,2,1]$
$(1,1,0)$	0	$H(-2.5\times1+2\times1+1\times0)=H(-0.5)=0$	0	$[0,0,0]$	$[-2.5,2,1]$
$(1,1,1)$	1	$H(-2.5\times1+2\times1+1\times1)=H(0.5)=1$	0	$[0,0,0]$	$[-2.5,2,1]$
$(1,0,0)$	0	$H(-2.5\times1+2\times0+1\times0)=H(-2.5)=0$	0	$[0,0,0]$	$[-2.5,2,1]$
$(1,0,1)$	0	$H(-2.5\times1+2\times0+1\times1)=H(-1.5)=0$	0	$[0,0,0]$	$[-2.5,2,1]$

Learning has converged, so required weights are $\mathbf{w} = [-2.5, 2, 1]$*, or equivalently* $\theta = 2.5$*,* $w_1 = 2$*,* $w_2 = 1$*.*

The decision surface looks like this:



6. Consider the following linearly separable data set.

$\mathbf{x}^t$	$class$
$(0,2)$	1
$(1,2)$	1
$(2,1)$	1
$(-3,1)$	0
$(-2,-1)$	0
$(-3,-2)$	0

Apply the Sequential Delta Learning Algorithm to find the parameters of a linear threshold neuron that will correctly classify this data. Assume initial values of $\theta = -1$**,** $w_1 = 0$ **and** $w_2 = 0$**, and a learning rate of 1.**

Using Augmented notation, $y = H(\mathbf{wx})$ *where* $\mathbf{w} = [-\theta, w_1, w_2]$*, and* $\mathbf{x} = [1, x_1, x_2]^T$*. So initial weight values are* $\mathbf{w} = [1, 0, 0]$ *and the dataset is:*

$\mathbf{x}^t$	t
$(1, 0, 2)$	1
$(1, 1, 2)$	1
$(1, 2, 1)$	1
$(1, -3, 1)$	0
$(1, -2, -1)$	0
$(1, -3, -2)$	0

For the Sequential Delta Learning Algorithm, weights are updated such that: $\mathbf{w} \leftarrow \mathbf{w} + \eta(t - y)\mathbf{x}^t$*. Here,* $\eta = 1$*.*

iteration	$\mathbf{w}$	$\mathbf{x}^t$	$y = H(\mathbf{wx})$	t	$\mathbf{w} \leftarrow \mathbf{w} + (t - y)\mathbf{x}^t$
1	$[1, 0, 0]$	$[1, 0, 2]$	1	1	$[1, 0, 0]$
2	$[1, 0, 0]$	$[1, 1, 2]$	1	1	$[1, 0, 0]$
3	$[1, 0, 0]$	$[1, 2, 1]$	1	1	$[1, 0, 0]$
4	$[1, 0, 0]$	$[1, -3, 1]$	1	0	$[1, 0, 0] - [1, -3, 1] = [0, 3, -1]$
5	$[0, 3, -1]$	$[1, -2, -1]$	0	0	$[0, 3, -1]$
6	$[0, 3, -1]$	$[1, -3, -2]$	0	0	$[0, 3, -1]$
7	$[0, 3, -1]$	$[1, 0, 2]$	0	1	$[0, 3, -1] + [1, 0, 2] = [1, 3, 1]$
8	$[1, 3, 1]$	$[1, 1, 2]$	1	1	$[1, 3, 1]$
9	$[1, 3, 1]$	$[1, 2, 1]$	1	1	$[1, 3, 1]$
10	$[1, 3, 1]$	$[1, -3, 1]$	0	0	$[1, 3, 1]$
11	$[1, 3, 1]$	$[1, -2, -1]$	0	0	$[1, 3, 1]$
12	$[1, 3, 1]$	$[1, -3, -2]$	0	0	$[1, 3, 1]$
13	$[1, 3, 1]$	$[1, 0, 2]$	1	1	$[1, 3, 1]$

Learning has converged (we have gone through all the data without needing to update the weights), so required parameters are $\mathbf{w} = (1, 3, 1)$*.*

7. A negative feedback network has three inputs and two output neurons, that are connected with weights $\mathbf{W} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$**. Determine the activation of the output neurons after 5 iterations when the input is** $\mathbf{x} = (1, 1, 0)^T$**, assuming that the output neurons are updated using parameter** $\alpha = 0.25$**, and the activations of the output neurons are initialised to be all zero.**

The activation of a negative feedback network is determined by iteratively evaluating the following equations:

$$\mathbf{e} = \mathbf{x} - \mathbf{W}^T\mathbf{y}$$

$$\mathbf{y} \leftarrow \mathbf{y} + \alpha\mathbf{We}$$

iteration	$\mathbf{e}^T$	$(\mathbf{We})^T$	$\mathbf{y}^T$	$(\mathbf{W}^T\mathbf{y})^T$
1	$(1, 1, 0)$	$(2, 2)$	$(0.5, 0.5)$	$(1, 1, 0.5)$
2	$(0, 0, -0.5)$	$(0, -0.5)$	$(0.5, 0.375)$	$(0.875, 0.875, 0.375)$
3	$(0.125, 0.125, -0.375)$	$(0.25, -0.125)$	$(0.5625, 0.34375)$	$(0.90625, 0.90625, 0.34375)$
4	$(0.09375, 0.09375, -0.34375)$	$(0.1875, -0.15625)$	$(0.60938, 0.30469)$	$(0.91406, 0.91406, 0.30469)$
5	$(0.085938, 0.085938, -0.30469)$	$(0.17188, -0.13281)$	$(0.65234, 0.27148)$	$(0.92383, 0.92383, 0.27148)$

So output is $\begin{pmatrix} 0.65234 \\ 0.27148 \end{pmatrix}$

Note, competition results in the first neuron increasing its output, while the output of the second neuron is suppressed. Also, note that the vector $\mathbf{W}^T\mathbf{y}$ *becomes similar to the input* $\mathbf{x}$*.* $\mathbf{W}^T\mathbf{y}$ *converges towards a reconstruction of the input.*

8. Repeat the previous question using a value of $\alpha = 0.5$**.**

$$\mathbf{e} = \mathbf{x} - \mathbf{W}^T\mathbf{y}$$
$$\mathbf{y} \leftarrow \mathbf{y} + \alpha\mathbf{W}\mathbf{e}$$

iteration	$\mathbf{e}^T$	$(\mathbf{W}\mathbf{e})^T$	$\mathbf{y}^T$	$(\mathbf{W}^T\mathbf{y})^T$
1	$(1, 1, 0)$	$(2, 2)$	$(1, 1)$	$(2, 2, 1)$
2	$(-1, -1, -1)$	$(-2, -3)$	$(0, -0.5)$	$(-0.5, -0.5, -0.5)$
3	$(1.5, 1.5, 0.5)$	$(3, 3.5)$	$(1.5, 1.25)$	$(2.75, 2.75, 1.25)$
4	$(-1.75, -1.75, -1.25)$	$(-3.5, -4.75)$	$(-0.25, -1.125)$	$(-1.375, -1.375, -1.125)$
5	$(2.375, 2.375, 1.125)$	$(4.75, 5.875)$	$(2.125, 1.8125)$	$(3.9375, 3.9375, 1.8125)$

So output is $\begin{pmatrix} 2.125 \\ 1.8125 \end{pmatrix}$

Note, competition results in oscillatory responses. If α *is too large the network becomes unstable. Instability is a common problem with recurrent neural networks.*

9. A more stable method of calculating the activations in a negative feedback network is to use the following update rules:

$$\mathbf{e} = \mathbf{x} \oslash \left[\mathbf{W}^T\mathbf{y}\right]_{\epsilon_2}$$
$$\mathbf{y} \leftarrow [\mathbf{y}]_{\epsilon_1} \odot \tilde{\mathbf{W}}\mathbf{e}$$

where $[v]_\epsilon = \max(\epsilon, v)$**;** ϵ_1 **and** ϵ_2 **are parameters;** $\tilde{\mathbf{W}}$ **is equal to** $\mathbf{W}$ **but with each row normalised to sum to one; and** $\oslash$ **and** $\odot$ **indicate element-wise division and multiplication respectively.**

This is called Regulatory Feedback or Divisive Input Modulation.

Determine the activation of the output neurons after 5 iterations when the input is $\mathbf{x} = (1, 1, 0)^T$**, and** $\mathbf{W} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$**, assuming that** $\epsilon_1 = \epsilon_2 = 0.01$**, and the activations of the output neurons are initialised to be all zero.**

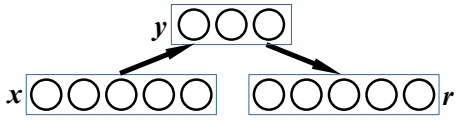
$$\tilde{\mathbf{W}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$

iteration	$\mathbf{e}^T$	$(\tilde{\mathbf{We}})^T$	$\mathbf{y}^T$	$(\mathbf{W}^T\mathbf{y})^T$
1	$(100, 100, 0)$	$(100, 66.66667)$	$(1, 0.66667)$	$(1.6667, 1.6667, 0.66667)$
2	$(0.6, 0.6, 0)$	$(0.6, 0.4)$	$(0.6, 0.26667)$	$(0.86667, 0.86667, 0.26667)$
3	$(1.1538, 1.1538, 0)$	$(1.1538, 0.76923)$	$(0.69231, 0.20513)$	$(0.89744, 0.89744, 0.20513)$
4	$(1.1143, 1.1143, 0)$	$(1.1143, 0.74286)$	$(0.77143, 0.15238)$	$(0.92381, 0.92381, 0.15238)$
5	$(1.0825, 1.0825, 0)$	$(1.0825, 0.72165)$	$(0.83505, 0.10997)$	$(0.94502, 0.94502, 0.10997)$

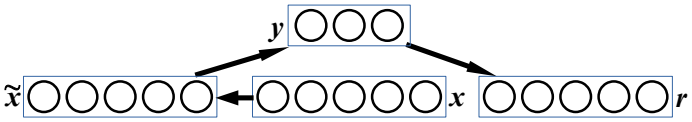
So output is $\begin{pmatrix} 0.83505 \\ 0.10997 \end{pmatrix}$

Note, competition results in the first neuron increasing its output, while the output of the second neuron is suppressed. Also, note that the vector $\mathbf{W}^T\mathbf{y}$ *becomes similar to the input* $\mathbf{x}$*.* $\mathbf{W}^T\mathbf{y}$ *converges towards a reconstruction of the input.*

10. The figure below shows an autoencoder neural network.



Draw a diagram of a de-noising autoencoder and briefly explain how a de-noising autoencoder is trained.



The network is trained so that the output, $\mathbf{r}$*, reconstructs the input,* $\mathbf{x}$*. However, before encoding is performed the input is corrupted with noise. This mitigates overfitting.*

7CCSMPNN: Pattern Recognition, Neural Networks and Deep Learning

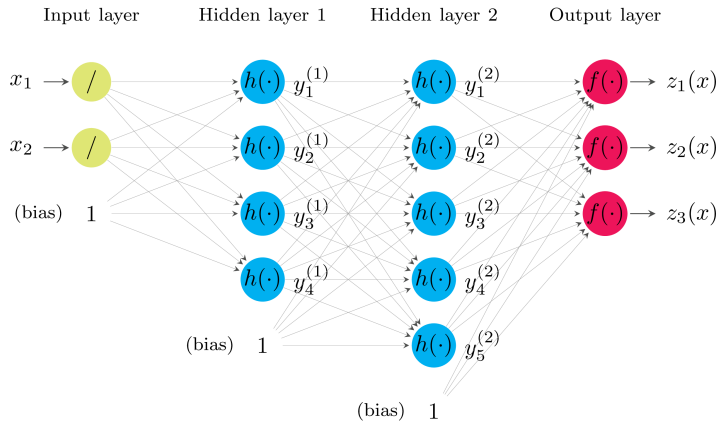
KING'S College LONDON

Tutorial - 4: MLPs and backpropagation

Instructors: David Watson, Stefanos Leonardos *Term: Spring 2025 – 03-Feb*

Exercise 1. Draw the diagram of a 2-input-3-output feedforward fully-connected neural network having 2 hidden layers with 4 and 5 units (nodes), respectively, and bias terms in the input and all hidden layers. How many connection weights does it have in total?

Solution 1. The fully connected 2-4-5-3 neural network is shown below.



The number of weights, $w(\text{layer}_{\text{out}}, \text{layer}_{\text{in}})$, between two consecutive layers is:

$$w(\text{input}, \text{hidden } 1) = (2+1) \times 4 = 12,$$
$$w(\text{hidden } 1, \text{hidden } 2) = (4+1) \times 5 = 25,$$
$$w(\text{hidden } 2, \text{output}) = (5+1) \times 3 = 18.$$

So, the total number of weights is: $12+25+18 = 55$.

Exercise 2. Derive the equation for the number of weights in a multi-layer feedforward fully-connected neural network including biases in the input and all hidden layers in terms of: i) d = number of inputs, ii) L = number of hidden layers, iii) n_h = number of units (nodes) in the h-th hidden layer, $h = 1, 2, \ldots, L$, and iv) z = number of outputs.

Solution 2. The number of weights, $w(h, h+1)$, between two consecutive layers, h and $h+1$, is given by the equation:

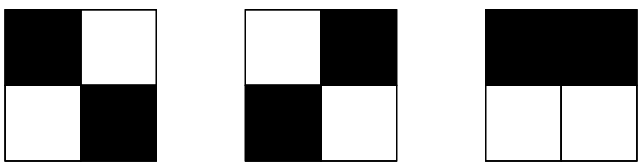
$$\boxed{\text{w}(h, h+1) = (\text{nodes}(h)+1) \times \text{nodes}(h+1) = \big(n_h + 1\big) \times n_{h+1}.}$$

where the $+1$ accounts for the bias term. Using the notation above, we have that:

$$w(\text{input}, \text{hidden } 1) = (d+1) \times n_1.$$
$$w(\text{hidden } h, \text{hidden } h+1) = (n_h+1) \times n_{h+1}, \quad h = 1, 2, \ldots, L-1.$$
$$w(\text{hidden } L, \text{output}) = (n_L+1) \times z.$$

Thus, the total number of weights, $N(w)$, is given by: $N(w) := (d+1) \times n_1 + \sum_{h=1}^{L-1} \big(n_h+1\big) \times n_{h+1} + \big(n_L+1\big) \times z.$

Exercise 3. Figure 1 shows three patterns of a classification problem. A feedforward fully-connected neural network



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

Figure 1: Three patterns.

is employed to classify the patterns. How many inputs and outputs should be used for the neural network? List the input patterns and target outputs in a table. Note: any pattern different from the above is classified as "otherwise".

Solution 3. One possible assignment is shown in the figure below:

x_1	x_2
x_4	x_3

For this classification task, we can employ a 4-input-1-output feedforward fully-connected neural network with the input patterns and target outputs shown in the table on the right: (where 0 = white, and 1 = black).

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

Note: What activation function(s) should/should not be used in the output node? How can we determine the configuration of the network (number and size of hidden layers)?

Exercise 4. Figure 2 shows a 4-3-2 feedforward fully-connected neural network which is used to classify the patterns shown in Figure 1. The assignment of input $\mathbf{x}^T = (x_1, x_2, x_3, x_4)$ is shown in Figure 3 where 0 represents an unfilled grid and 1 represents a filled grid. The activation functions in the input, hidden and output layers are linear, hyperbolic tangent sigmoid and logarithmic sigmoid, respectively. Given the weight and bias matrices

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

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

determine the output patterns representing the three classes.

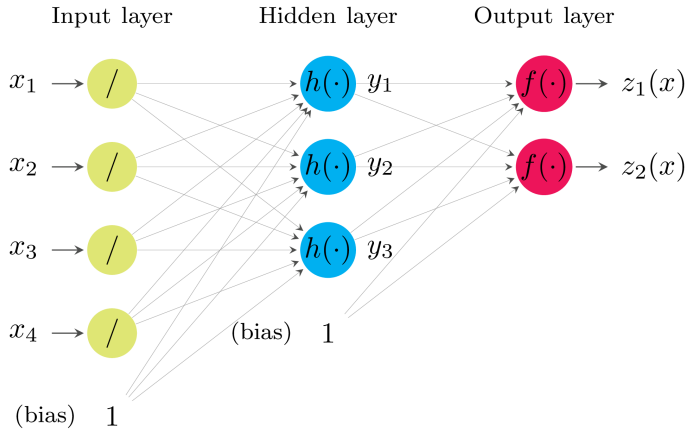


Figure 2: Diagram of the 4-3-2 feedforward ected neural network.

x_1	x_2
x_4	x_3

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

Solution 4. The first line of $\mathbf{W}_{ji}$ shows the weights w_{11}, w_{12}, w_{13} and w_{14}. The other lines are analogous. Equivalently, the first column of $\mathbf{W}_{ji}$ shows the weights w_{11}, w_{21} and w_{31}. Similarly, for the other matrices. Thus, using the notation in Figure 2, it holds that:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = h(\mathbf{W}_{ji}\mathbf{x} + \mathbf{W}_{j0}),$$

$$\mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = f(\mathbf{W}_{kj}\mathbf{y} + \mathbf{W}_{k0}).$$

pattern	$\mathbf{x}^T$	$\mathbf{y}^T$	$\mathbf{z}^T$
1	$(1,0,1,0)$	$(1, 0.4304, 0.9923)$	$(0,1)$
2	$(0,1,0,1)$	$(1, -0.9930, -0.9983)$	$(1,0)$
3	$(1,1,0,0)$	$(1, 0.9929, -0.9799)$	$(1,1)$

So, pattern 1 is represented by $(0,1)^T$, pattern 2 by $(1,0)^T$ and pattern 3 by $(1,1)^T$. Note: if an input sample gives $z = (0.6, 0.8)^T$ which class should it be classified to?

Exercise 5. Figure 4 shows a 3-layer, partially-connected feedforward neural network. The input units use linear activation functions and all hidden and output units use hyperbolic tangent sigmoid activation functions. The neural network is trained using stochastic backpropagation on the cost function $J = \frac{1}{2}\|t - z_1\|^2$, where z_1 is the network output corresponding to input pattern $\mathbf{x}$ selected from the training set and t is its corresponding target output.

i) Considering $\mathbf{x}^T = (0.1, 0.9)$ determine y_1, y_2 and z_1.
ii) Derive the backpropagation update rules for m_{10} and w_{21} and calculate the updated values for m_{10} and w_{21} for the next iteration using $\mathbf{x}^T = (0.1, 0.9)$, the weights in Figure 4, $\mathbf{t} = 0.5$ and learning rate $\eta = 0.25$.

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

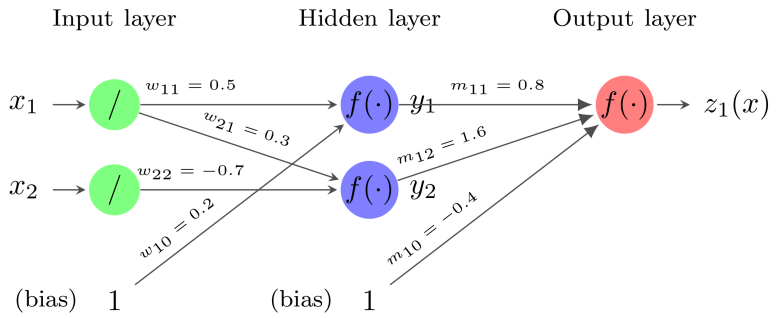


Figure 4: Diagram of a 3-layer partially-connected feedforward neural network.

Solution 5. The hyperbolic tangent sigmoid activation function is given by $f(x) = \frac{2}{1+e^{-2x}} - 1$ with $f(x) \in (-1, 1)$ and $f'(x) = \frac{df(x)}{dx} = \frac{4e^{-2x}}{(1+e^{-2x})^2}$ for any $x \in (-\infty, +\infty)$.

i) For $\mathbf{x}^T = (0.1, 0.9)$, it holds that

$$\begin{aligned} y_1 &= f(0.5 \times x_1 + 0 \times x_2 + 1 \times 0.2) = f(0.25) = 0.2449, \\ y_2 &= f(0.3 \times x_1 - 0.7 \times x_2) = f(-0.6) = -0.5370, \\ z_1 &= f(0.8 \times y_1 + 1.6 \times y_2 - 1 \times 0.4) = f(-1.0633) = -0.7869. \end{aligned}$$

ii) Considering that $J(z_1) = \frac{1}{2}(\mathbf{t} - z_1)^2$, we have that

$$\frac{\partial J}{\partial m_{10}} = \frac{\partial J}{\partial f(net)} \cdot \frac{\partial f(net)}{\partial net} \cdot \frac{\partial net}{\partial m_{10}}$$

where $f(net) = \frac{2}{1+e^{-2net}} - 1 (= z_1)$ and $net = m_{11}y_1 + m_{12}y_2 + m_{10}$. Thus,

$$\begin{aligned} \frac{\partial J}{\partial m_{10}} &= \frac{\partial J}{\partial f(net)} \cdot \frac{\partial f(net)}{\partial net} \cdot \frac{\partial net}{\partial m_{10}} \\ &= \frac{\partial}{\partial z}\left(\frac{1}{2}(t - z_1)^2\right) \cdot \frac{\partial}{\partial net}\left(\frac{2}{1+e^{-2net}} - 1\right) \cdot \frac{\partial}{\partial m_{10}}(m_{11}y_1 + m_{12}y_2 + m_{10}) \\ &= (z_1 - t) \cdot \frac{4e^{-2net}}{(1+e^{-2net})^2} \cdot 1 \end{aligned}$$

The gradient descent update rule is given by: $m_{10} \leftarrow m_{10} + \Delta m_{10}$ where $\Delta m_{10} = -\eta \frac{\partial J}{\partial m_{10}}$. Thus,

$$m_{10} \leftarrow m_{10} + \Delta m_{10} = m_{10} - \eta \cdot (z_1 - t) \cdot \frac{4e^{-2net}}{(1+e^{-2net})^2}.$$

Considering that $x^T = (0.1, 0.9), t = 0.5, \eta = 0.25$ and $z_1 = -0.7869, net = -1.0633$ from part i), we have that

$$m_{10} = -0.4 - 0.25 \cdot (-0.7869 - 0.5) \cdot \frac{4e^{-2\cdot(-1.0633)}}{(1+e^{-2\cdot(-1.0633)})^2} = -0.2775.$$

Similarly, for w_{21}, we have that:

$$\begin{aligned} \frac{\partial J}{\partial w_{21}} &= \underbrace{\frac{\partial J}{\partial f(net)} \cdot \frac{\partial f(net)}{\partial net}}_{\text{we already know that}} \cdot \frac{\partial net}{\partial y_2} \cdot \frac{\partial y_2}{\partial a} \cdot \frac{\partial a}{\partial w_{21}} \\ &= \underbrace{(z_1 - t) \cdot \frac{4e^{-2net}}{(1+e^{-2net})^2}}_{\text{as above}} \cdot \frac{\partial}{\partial y_2}(m_{11}y_1 + m_{12}y_2 + m_{10}) \cdot \frac{\partial}{\partial a}\frac{4e^{-2a}}{(1+e^{-2a})^2} \cdot \frac{\partial}{\partial w_{21}}(w_{21}x_1 + w_{22}x_2) \\ &= (z_1 - t) \cdot \frac{4e^{-2net}}{(1+e^{-2net})^2} \cdot m_{12} \cdot \frac{4e^{-2a}}{(1+e^{-2a})^2} \cdot x_1. \end{aligned}$$

Thus, using the gradient descent update rule, we have for $x^T = (0.1, 0.9)$ that

$$w_{21} \leftarrow w_{21} + \Delta w_{21} = w_{21} - \eta \cdot (z_1 - = t) \cdot \frac{4e^{-2net}}{(1+e^{-2net})^2}$$
$$= w_{21} - \eta \cdot (z_1 - t) \cdot \frac{4e^{-2net}}{(1+e^{-2net})^2} \cdot m_{12} \cdot \frac{4e^{-2a}}{(1+e^{-2a})^2} \cdot x_1$$
$$= 0.3 - 0.25 \cdot (-0.7869 - 0.5) \cdot \frac{4e^{-2\cdot(-1.0633)}}{(1+e^{-2\cdot(-1.0633)})^2} \cdot 1.6 \cdot \frac{4e^{-2\cdot(-0.6)}}{(1+e^{-2\cdot(-0.6)})^2} \cdot 0.1 = 0.3140.$$

iii) With the updated m_{10} and w_{21}, we have for $\mathbf{x} = (0.1, 0.9)^T$ that

$$y_1 = f(0.5 \times x_1 + 0 \times x_2 + 1 \times 0.2) = f(0.25) = 0.2449,$$
$$y_2 = f(0.3140 \times x_1 - 0.7 \times x_2) = f(-0.5986) = -0.5361,$$
$$z_1 = f(0.8 \times y_1 + 1.6 \times y_2 - 1 \times 0.2775) = f(-0.9393) = -0.7349.$$

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

Exercise 6. Consider a 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 and use Gaussian function with $\sigma_j = \frac{\rho_{\max}}{\sqrt{2n_h}}$ in the hidden units 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.

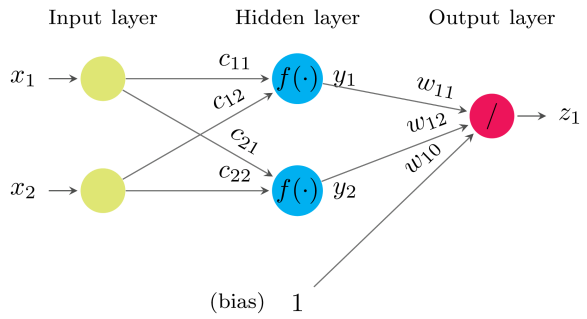
p	x_1	x_2
1	0.5	-0.1
2	-0.2	1.2
3	0.8	0.3
4	1.8	0.6

Table 2: Input patterns for the XOR classifier.

i) Draw a diagram showing the structure of an RBF neural network solving the XOR problem.
ii) 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.
iii) Determine the output weights using the least square method. Compute the outputs at the output unit(s) and list them in a table. Show that the XOR problem is solved.
iv) Determine the class for the input patterns in Table 2.

Solution 6.

i) A 2-2-1 RBF neural network for the XOR problem is shown in the figure below



ii) Given that $\mathbf{c}_1 = (0, 0)^T$ (pattern 1), $\mathbf{c}_2 = (1, 1)^T$ (pattern 4) are used as the centres, we have that

$$\rho_{\max} = \|\mathbf{c}_1 - \mathbf{c}_2\| = \sqrt{(0-1)^2 + (0-1)^2} = \sqrt{2}.$$

Since the number, n_h, of units/nodes in the hidden layer is $n_h = 2$, we also have that

$$\sigma_1 = \sigma_2 = \frac{\rho_{\max}}{\sqrt{2n_h}} = \frac{\sqrt{2}}{\sqrt{2 \cdot 2}} = \frac{1}{\sqrt{2}}.$$

Thus,

$$y_j(\mathbf{x}_p) = \exp\left\{-\frac{\|\mathbf{x}_p - \mathbf{c}_j\|^2}{2\sigma_j^2}\right\}, \qquad \text{for } j = 1, 2.$$

For the patterns $\mathbf{x}_p, p = 1, 2, 3, 4$ in Table 2, the above equation yields:

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

Either by plotting $y_2(\mathbf{x})$ against $y_1(\mathbf{x})$ or by observing that both feature vectors of the $t = 1$ class have collapsed to the same point that is not on the same line with the two feature vectors of the $t = 0$ class (why?), it can be seen that the patterns corresponding to the classes at the hidden units are linearly separable.

iii) The output neuron is intended to act as a dichotomizer, so we should use $t = 1$ for class 1, and $t = 0$ for class 0. Recalling that the activation function in the output units is a *linear function*, the output equation of the RBF neural network is:

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

There are 3 weights to be determined: w_{10} (bias), w_{11} and w_{12}. We can calculate the output for all input patterns, in matrix notation, as $\mathbf{Z} = \mathbf{Y}\mathbf{w} = \mathbf{T}$ which gives:

$$\begin{pmatrix} 1 & 1 & 0.1353 \\ 1 & 0.3679 & 0.3679 \\ 1 & 0.3679 & 0.3679 \\ 1 & 0.1353 & 1 \end{pmatrix} \begin{pmatrix} w_{10} \\ w_{11} \\ w_{12} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \Rightarrow \begin{pmatrix} w_{10} \\ w_{11} \\ w_{12} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0.1353 \\ 1 & 0.3679 & 0.3679 \\ 1 & 0.3679 & 0.3679 \\ 1 & 0.1353 & 1 \end{pmatrix}^{\dagger} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

where $\dagger$ denotes the pseudo-inverse of $\mathbf{Y}$. Using the command `pinv` in MATLAB, we find that: $w_{10} = 2.8413, w_{11} = w_{12} = -2.5027$. Thus, the output equation of the RBF neural network is:

$$z_1(\mathbf{x}) = -2.5027 y_1(\mathbf{x}) - 2.5027 y_2(\mathbf{x}) + 2.8413.$$

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

p	x_1	x_2	$y_1(\mathbf{x})$	$y_2(\mathbf{x})$	$z_1(\mathbf{x})$
1	0	0	1	0.1353	0
2	0	1	0.3679	0.3679	1
3	1	0	0.3679	0.3679	1
4	1	1	0.1353	1	0

iv) Putting the input patterns to the RBF classifier, we obtain the result in the following table. Note, that we can produce categorical outputs by applying the *heaviside function*, $H(z_1(\mathbf{x})) = \frac{1}{2} \cdot [\text{sign}(z_1(x) - 0.5) + 1]$, to the output.

p	x_1	x_2	$y_1(\mathbf{x})$	$y_2(\mathbf{x})$	$z_1(\mathbf{x})$	Class: $H(z_1(\mathbf{x}))$
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

Exercise 7. Consider a classifier which is able to separate two classes, such that x belongs to class 1 if $g(x) \geq x/2$ and x belongs to class 2 if $g(x) < x/2$, where $g(x)$ is an unknown function for which we wo only know the input-output mappings shown in Table 3

.

p	1	2	3	4	5	6	7	8	9	10	11	12
x	0.05	0.20	0.25	0.30	0.40	0.43	0.48	0.60	0.70	0.80	0.90	0.95
z	0.0863	0.2662	0.2362	0.1687	0.1260	0.1756	0.3290	0.6694	0.4573	0.3320	0.4063	0.3535

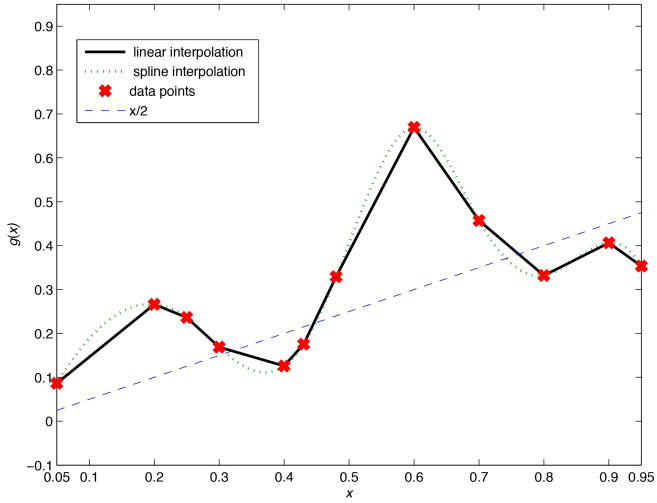
Table 3: Input-output mappings for function $g(x)$.

i) Sketch the function $g(x)$ and determine the possible class of the following points $x = 0.1, 0.35, 0.55, 0.75, 0.9$.
ii) If an RBF neural network is employed to implement the classifier, determine its number of inputs and outputs.
iii) An RBF neural network with three hidden units is employed to implement the classifier. All hidden units use Gaussian function with $\sigma = 0.1$ and all output units use linear function. Given that $c_1 = 0.2, c_2 = 0.6$, and $c_3 = 0.9$, use the least squares method to determine the output weights of the RBF neural network and provide its equation.
iv) 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. All hidden units use Gaussian function with $\sigma = 2\rho_{\text{avg}}$ and all output use linear function. Based on the found centres, use the least squares method to find the output weights of the RBF neural network. Draw the diagram of the designed network and show its weights.

Solution 7.

i)

x	Class
0.10	1
0.35	2
0.55	1
0.75	1 (hard to tell)
0.90	2



ii) A single-input-single-output RBF neural network which takes x as an input and z as output for this classification problem.
iii) The equation of the output z is: $z(x) = w_{10} + w_{11}y_1(x) + w_{12}y_2(x) + w_{13}y_3(x)$ where

$$y_1(x) = \exp\left\{-\frac{(x-0.2)^2}{2\times 0.1^2}\right\},\ y_2(x) = \exp\left\{-\frac{(x-0.6)^2}{2\times 0.1^2}\right\},\ \text{and}\ y_3(x) = \exp\left\{-\frac{(x-0.9)^2}{2\times 0.1^2}\right\}.$$

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

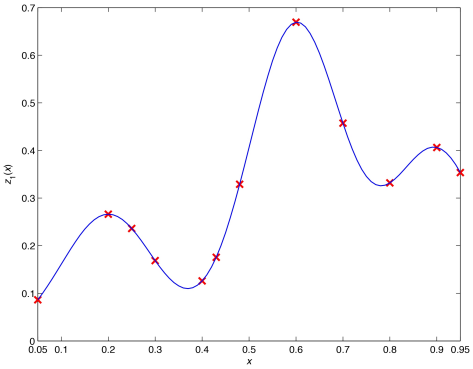
We can calculate the output for all input patterns, in matrix notation, as: $\mathbf{Z} = \mathbf{wY} = \mathbf{T}$, where $\mathbf{w} = (w_{10}, w_{11}, w_{12}, w_{13})$, and

$$\mathbf{Y} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.3247 & 1 & 0.8825 & 0.6065 & 0.1353 & 0.0710 & 0.0198 & 0.0003 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0003 & 0.0022 & 0.0111 & 0.1353 & 0.2357 & 0.4868 & 1.0000 & 0.6065 & 0.1353 & 0.0111 & 0.0022 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0111 & 0.1353 & 0.6065 & 1.0000 & 0.8825 \end{pmatrix},$$

$\mathbf{T} = (0.0863, 0.2662, 0.2362, 0.1687, 0.1260, 0.1756, 0.3290, 0.6694, 0.4573, 0.3320, 0.4063, 0.3535)$.

Thus, using MATLAB, we can solve the above equation by calculating the pseudo-inverse of $\mathbf{Y}$ as $\mathbf{w} = \mathbf{TY}^{\dagger}$, which yields: $\mathbf{w} = (0, 0.2660, 0.6649, 0.3990)$. Thus, the output equation of the RBF neural network is

$$
\begin{aligned}
z(x) &= 0.2660y_1(x) + 0.6649y_2(x) + 0.3990y_3(x) \\
&= 0.2660 \exp\left\{-\frac{(x-0.2)^2}{2\times 0.1^2}\right\} \\
&+ 0.6649 \exp\left\{-\frac{(x-0.6)^2}{2\times 0.1^2}\right\} \\
&+ 0.3990 \exp\left\{-\frac{(x-0.9)^2}{2\times 0.1^2}\right\}.
\end{aligned}
$$



iv) The centre of each cluster is determined as the averages of all points in that cluster:

$$
\begin{aligned}
S_1 = \{x_1, x_2, x_3\} \Rightarrow c_1 &= \frac{0.05 + 0.2 + 0.25}{3} = 0.1667, & S_3 = \{x_6, x_7, x_8, x_9\} \Rightarrow c_3 &= 0.5525, \\
S_2 = \{x_4, x_5\} \Rightarrow c_2 &= \frac{0.3 + 0.4}{2} = 0.35, & S_4 = \{x_{10}, x_{11}, x_{12}\} \Rightarrow c_4 &= 0.8833.
\end{aligned}
$$

The output equation of the RBF neural network is:

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

where $y_i(x) = e^{-\frac{(x-c_i)^2}{2\times\sigma^2}}$ for $i = 1, 2, 3, 4$ and

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

Proceeding as in part iii), i.e., by calculating $y_i(x)$ for all $i = 1, 2, 3, 4$ and x in Table 3, and by calculating the pseudoinverse of the resulting matrix $\mathbf{Y}$, we find that

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{10} \end{pmatrix} = \left(\mathbf{Y}^{\dagger} z_1(x)\right)^T = \begin{pmatrix} 129.30 & -235.37 & 129.73 & -4.68 & -11.99 \end{pmatrix}.$$

Thus, the output equation of the RBF neural network is:

$$z_1(x) = 129.30y_1(x) - 235.37y_2(x) + 129.73y_3(x) - 4.68y_4(x) - 11.99.$$

Note: Can you draw the resulting RBF neural networks in questions iii) and iv)? Which of the two performs better?

Exercise 8. A nonlinear classifier with multiple inputs and a single output is implemented by 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.

Solution 8. The following procedure is employed to redesign the classifier using an RBF neural network.

Step	Description
1	Generate a set of random input patterns $\mathbf{x}_p \in \mathbb{R}^{d\times p}$, $p = 1, \ldots, n$, of which each entry is in the range of -1 and 1. Apply the input patterns to the feedforward neural network and collect the output as $\mathbf{t}_p \in \mathbb{R}^p$.
2	Form the dataset with the obtained input patterns and $\mathbf{x}_p$ and output classes $\mathbf{t}_p$.
3	Divide the dataset into training, test and validation sets, say, in a ration of $70\% - 15\% - 15\%$.
4	Choose the activation functions for the hidden and output units.
5	Determine the number of hidden units and centres by some algorithms.
6	If linear activation functions are used in the output units, employ the whole dataset to determine the output weights using the pseudo-inverse method. If nonlinear activation functions are used in the output units, employ stochastic or batch backpropagation to learn the parameters (output weights and parameters of activation functions, and even the centres) using the training set. A possible training-stopping criterion is when the MSE for the validation set start to increase.
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.

7CCSMPNN: Pattern Recognition, Neural Networks and Deep Learning

KING'S College LONDON

Tutorial - 5: Deep discriminative neural networks

Instructors: David Watson, Stefanos Leonardos

Term: Spring 2025 – 10-Feb

Exercise 1. Give brief definitions of the following terms

i) feature map
ii) sub-sampling
iii) adversarial example
iv) data augmentation
v) transfer learning
vi) dropout
vii) regularization

Solution 1.

i) The output produced by a single mask (before or after the application of the activation function, or pooling) in a convolutional layer of a CNN. It shows the responses produced by identical neurons at different spatial locations.
ii) The process of reducing the width and height of a feature map by pooling (also the process of reducing the resolution of an image).
iii) A sample that has been manipulated so that it is classified differently to the original sample.
iv) A method of expanding the number of samples in a dataset by adding new samples that are the original samples after the application of class-preserving transformations.
v) A method of training a classifier that utilises the results of pre-training the classifier using another dataset.
vi) A method of reducing over-fitting that works by giving an activation of zero to a random sample of neurons in a layer at each iteration during training.
vii) Any method intended to reduce over-fitting and improve generalisation.

Exercise 2. Define what is meant by, and explain the causes of, the "vanishing gradient problem". Briefly describe four methods that can be used to reduce its effects.

Solution 2. When errors backpropagate through a neural network, they become smaller and smaller resulting in smaller and smaller weight updates. This is due to the error being multiplied at each layer of the network by derivatives or weights that are small and can result in little or no learning occurring in the earlier layers of the network which, in turn, causes these layer to fail to contribute to solving the task. Methods that reduce the likelihood of gradients vanishing are:

i) using activation functions, like ReLU, which have a smaller range for which the derivative is < 1.
ii) initialising the weights in ways that have been found, empirically, to reduce the effects of vanishing gradients.
iii) using variations of standard backpropagation that use momentum and/or an adaptive learning rate.
iv) using batch normalisation to keep the mean and standard deviation of each neuron close to 0 and 1 respectively.
v) using skip connections so the gradient can by-pass layers in the network where the gradient has vanished.

Exercise 3. Mathematically define the following activation functions

i) ReLU
ii) LReLU
iii) PReLU

Solution 3.

i) ReLU $\varphi(net_j) = net_j$, if $net_j \geq 0$ or 0, if $net_j < 0$.
ii) LReLU $\varphi(net_j) = net_j$, if $net_j \geq 0$ or $a \times net_j$, if $net_j < 0$ where a is a constant hyper-parameter.
iii) PReLU $\varphi(net_j) = net_j$, if $net_j \geq 0$ or $a_j \times net_j$, if $net_j < 0$ where a_j is a learnt parameter.

Exercise 4. The following array shows the output produced by a mask in a convolutional layer of a convolutional neural network (CNN)

$$\text{net}_j = \begin{bmatrix} 1 & 0.5 & 0.2 \\ -1 & -0.5 & -0.2 \\ 0.1 & -0.1 & 0 \end{bmatrix}.$$

Calculate the values produced by the application of the following activation functions

i) ReLU,
ii) LReLU when $a = 0.1$,
iii) tanh,
iv) heaviside function with threshold 0.1 for each neuron (define $H(0) = 0.5$).

Solution 4.

i) $\begin{bmatrix} 1 & 0.5 & 0.2 \\ 0 & 0 & 0 \\ 0.1 & 0 & 0 \end{bmatrix}$.

ii) $\begin{bmatrix} 1 & 0.5 & 0.2 \\ -0.1 & -0.05 & -0.02 \\ 0.1 & -0.01 & 0 \end{bmatrix}$.

iii) $\begin{bmatrix} 0.7616 & 0.4621 & 0.1974 \\ -0.7616 & -0.4621 & -0.1974 \\ 0.0997 & -0.0997 & 0 \end{bmatrix}$.

iv) $\begin{bmatrix} H(1-0.1) & H(0.5-0.1) & H(0.2-0.1) \\ H(-1-0.1) & H(-0.5-0.1) & H(-0.2-0.1) \\ H(0.1-0.1) & H(-0.1-0.1) & H(0-0.1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0.5 & 0 & 0 \end{bmatrix}$.

Exercise 5. The following arrays show the output produced by a convolutional layer to all 4 samples in a batch

$$X_1 = \begin{bmatrix} 1 & 0.5 & 0.2 \\ -1 & -0.5 & -0.2 \\ 0.1 & -0.1 & 0 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 1 & -1 & 0.1 \\ 0.5 & -0.5 & -0.1 \\ 0.2 & -0.2 & 0 \end{bmatrix}, \quad X_3 = \begin{bmatrix} 0.5 & -0.5 & -0.1 \\ 0 & -0.4 & 0 \\ 0.5 & 0.5 & 0.2 \end{bmatrix}, \quad X_4 = \begin{bmatrix} 0.2 & 1 & -0.2 \\ -1 & -0.6 & -0.1 \\ 0.1 & 0 & 0.1 \end{bmatrix}.$$

Calculate the corresponding outputs produced after the application of batch normalisation, assuming the following parameter values $\beta = 0, \gamma = 1$, and $\epsilon = 0.1$ which are the same for all neurons.

Solution 5. Batch normalisation modifies the output of an individual neuron, x, to become:

$$BN(x) = \beta + \gamma \frac{x - \mathbb{E}(x)}{\sqrt{\text{Var}(x) + \epsilon}}$$

where $\mathbb{E}(x)$ is the mean, and $\text{Var}(x)$ is the variance of x within the batch. For the top-left neuron, we have $\mathbb{E}(x) = \frac{1}{4}[1 + 1 + 0.5 + 0.2] = 0.675$, and $\text{Var}(x) = \frac{1}{4}\left[(1 - 0.675)^2 + (1 - 0.675)^2 + (0.5 - 0.675)^2 + (0.2 - 0.675)^2\right] = 0.1169$.

$$BN(x_1) = 0 + 1 \times \frac{1 - 0.675}{\sqrt{0.1169 + 0.1}} = 0.6979, \quad BN(x_2) = 0 + 1 \times \frac{1 - 0.675}{\sqrt{0.1169 + 0.1}} = 0.6979,$$

$$BN(x_3) = 0 + 1 \times \frac{0.5 - 0.675}{\sqrt{0.1169 + 0.1}} = -0.3758, \quad BN(x_4) = 0 + 1 \times \frac{0.2 - 0.675}{\sqrt{0.1169 + 0.1}} = -1.0200.$$

Applying the same method to the output of each neuron gives:

$$BN(X_1) = \begin{bmatrix} 0.6979 & 0.5872 & 0.5657 \\ -0.8652 & 0 & -0.3086 \\ -0.3509 & -0.3612 & -0.2294 \end{bmatrix}, \quad BN(X_2) = \begin{bmatrix} 0.6979 & -1.1744 & 0.2828 \\ 1.2112 & 0 & 0 \\ -0.0702 & -0.6019 & -0.2294 \end{bmatrix},$$

$$BN(X_3) = \begin{bmatrix} -0.3758 & -0.5872 & -0.2828 \\ 0.5191 & 0.3086 & 0.3086 \\ 0.7720 & 1.0835 & 0.3824 \end{bmatrix}, \quad BN(X_4) = \begin{bmatrix} -1.0200 & 1.1744 & -0.5657 \\ -0.8652 & -0.3086 & 0 \\ -0.3509 & -0.1204 & 0.0765 \end{bmatrix}.$$

Remarks:

i) The top-left neuron's outputs were originally all positive, but after batch normalisation have been scaled to be both positive and negative; the top-middle neuron's outputs were originally well spread out in the range -1 to +1 and have not been changed much after batch normalisation; the top-right neuron's outputs were originally all similar, but after batch normalisation have been scaled to cover a wider range of values.
ii) Often, the batch normalisation parameters β and γ are not pre-defined hyper-parameters but are learnt, via back-propagation, at the same time as the weights of the network.
iii) For convolutional layers, rather than performing batch normalisation separately at each location in each feature map, it is common to apply the same normalisation to all locations in a feature map by calculating the mean and variance across all locations and all samples in the batch: so that different elements of the same feature map, at different locations, are normalised in the same way. However, for the purposes of the tutorial and exam questions, we will use the formulation used above.
iv) Other forms of normalisation, such as layer normalisation, group normalisation, and instance normalisation, use the same method but calculate the mean and variance across different dimensions.

Exercise 6. The following arrays show the feature maps that provide the input to a convolutional layer of a CNN

$$X_1 = \begin{bmatrix} 0.2 & 1 & 0 \\ -1 & 0 & -0.1 \\ 0.1 & 0 & 0.1 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 1 & 0.5 & 0.2 \\ -1 & -0.5 & -0.2 \\ 0.1 & -0.1 & 0 \end{bmatrix}.$$

If a mask, H, has two channels defined as

$$H_1 = \begin{bmatrix} 1 & -0.1 \\ 1 & -0.1 \end{bmatrix}, \quad H_2 = \begin{bmatrix} 0.5 & 0.5 \\ -0.5 & -0.5 \end{bmatrix},$$

calculate the output produced by mask H when using

i) padding $= 0$, stride $= 1$,
ii) padding $= 1$, stride $= 1$,
iii) padding $= 1$, stride $= 2$,
iv) padding $= 0$, stride $= 1$, dilation $= 2$.

Solution 6.

i) We first calculate the convolutions $X_1 \star H_1$ and $X_2 \star H_2$:

$$X_1 \star H_1 = \begin{bmatrix} (0.2 \times 1) + (1 \times -0.1) + (-1 \times 1) + (0 \times -0.1) & (1 \times 1) + (0 \times -0.1) + (0 \times 1) + (-0.1 \times -0.1) \\ (-1 \times 1) + (0 \times -0.1) + (0.1 \times 1) + (0 \times -0.1) & (0 \times 1) + (-0.1 \times -0.1) + (0 \times 1) + (0.1 \times -0.1) \end{bmatrix}$$

$$X_2 \star H_2 = \begin{bmatrix} (1 \times 0.5) + (0.5 \times 0.5) + (-1 \times -0.5) + (-0.5 \times -0.5) & (0.5 \times 0.5) + (0.2 \times 0.5) + (-0.5 \times -0.5) + (-0.2 \times -0.5) \\ (-1 \times 0.5) + (-0.5 \times 0.5) + (0.1 \times -0.5) + (-0.1 \times -0.5) & (-0.5 \times 0.5) + (-0.2 \times 0.5) + (-0.1 \times -0.5) + (0 \times -0.5) \end{bmatrix}$$

Thus, $X_1 \star H_1 + X_2 \star H_2 = \begin{bmatrix} -0.9 & 1.01 \\ -0.9 & 0 \end{bmatrix} + \begin{bmatrix} 1.5 & 0.7 \\ -0.75 & -0.3 \end{bmatrix} = \begin{bmatrix} 0.6 & 1.71 \\ -1.65 & -0.3 \end{bmatrix}$.

ii) $X_1^{\text{pad}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2 & 1 & 0 & 0 \\ 0 & -1 & 0 & -0.1 & 0 \\ 0 & 0.1 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, X_2^{\text{pad}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0.5 & 0.2 & 0 \\ 0 & -1 & -0.5 & -0.2 & 0 \\ 0 & 0.1 & -0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$. Thus, $X_1^{\text{pad}} \star H_1 + X_2^{\text{pad}} \star H_2 =$

$$= \begin{bmatrix} -0.02 & 0.1 & 1 & 0 \\ 0.08 & -0.9 & 1.01 & -0.1 \\ 0.09 & -0.9 & 0 & 0 \\ -0.01 & 0.1 & -0.01 & 0.1 \end{bmatrix} + \begin{bmatrix} -0.5 & -0.75 & -0.35 & -0.1 \\ 1 & 1.5 & 0.7 & 0.2 \\ -0.55 & -0.75 & -0.3 & -0.1 \\ 0.05 & -0 & -0.05 & 0 \end{bmatrix} = \begin{bmatrix} -0.52 & -0.65 & 0.65 & -0.1 \\ 1.08 & 0.6 & 1.71 & 0.1 \\ -0.46 & -1.65 & -0.3 & -0.1 \\ 0.04 & 0.1 & -0.06 & 0.1 \end{bmatrix}.$$

iii) Using the answer to previous part: $\begin{bmatrix} -0.52 & 0.65 \\ -0.46 & -0.30 \end{bmatrix}$.

iv) $H_1' = \begin{bmatrix} 1 & 0 & -0.1 \\ 0 & 0 & 0 \\ 1 & 0 & -0.1 \end{bmatrix}, H_2' = \begin{bmatrix} 0.5 & 0 & 0.5 \\ 0 & 0 & 0 \\ -0.5 & 0 & -0.5 \end{bmatrix}$. Thus, $X_1 \star H_1' + X_2 \star H_2' = \begin{bmatrix} 0.29 \end{bmatrix} + \begin{bmatrix} 0.55 \end{bmatrix} = \begin{bmatrix} 0.84 \end{bmatrix}$.

Exercise 7. The following arrays show the feature maps that provide the input to a convolutional layer of a CNN

$$X_1 = \begin{bmatrix} 0.2 & 1 & 0 \\ -1 & 0 & -0.1 \\ 0.1 & 0 & 0.1 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 1 & 0.5 & 0.2 \\ -1 & -0.5 & -0.2 \\ 0.1 & -0.1 & 0 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 0.5 & -0.5 & -0.1 \\ 0 & -0.4 & 0 \\ 0.5 & 0.5 & 0.2 \end{bmatrix}.$$

Calculate the output produced by 1×1 convolution when the 3 channels of the 1×1 mask are: $[1, -1, 0.5]$.

Solution 7. $Y =$

$$= \begin{bmatrix} (0.2 \times 1) + (1 \times -1) + (0.5 \times 0.5) & (1 \times 1) + (0.5 \times -1) + (-0.5 \times 0.5) & (0 \times 1) + (0.2 \times -1) + (-0.1 \times 0.5) \\ (-1 \times 1) + (-1 \times -1) + (0 \times 0.5) & (0 \times 1) + (-0.5 \times -1) + (-0.4 \times 0.5) & (-0.1 \times 1) + (-0.2 \times -1) + (0 \times 0.5) \\ (0.1 \times 1) + (0.1 \times -1) + (0.5 \times 0.5) & (0 \times 1) + (-0.1 \times -1) + (0.5 \times 0.5) & (0.1 \times 1) + (0 \times -1) + (0.2 \times 0.5) \end{bmatrix}$$

$$= \begin{bmatrix} -0.55 & 0.25 & -0.25 \\ 0 & 0.3 & 0.1 \\ 0.25 & 0.35 & 0.2 \end{bmatrix}.$$

Exercise 8. The following array shows the input to a pooling layer of a CNN: $X_1 = \begin{bmatrix} 0.2 & 1 & 0 & 0.4 \\ -1 & 0 & -0.1 & -0.1 \\ 0.1 & 0 & -1 & -0.5 \\ 0.4 & -0.7 & -0.5 & 1 \end{bmatrix}$.

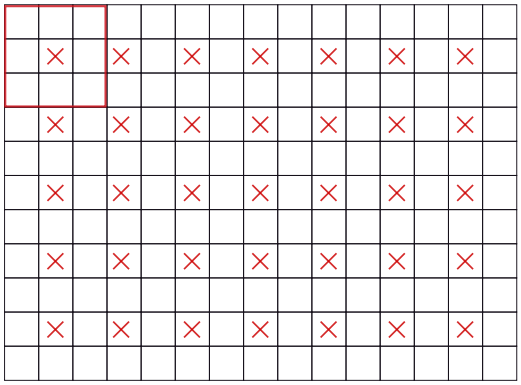
Calculate the output produced by the pooling layer when using

i) average pooling with a pooling region of 2×2 and stride $= 2$,
ii) max pooling with a pooling region of 2×2 and stride $= 2$,
iii) max pooling with a pooling region of 3×3 and stride $= 1$.

Solution 8. i) $\begin{bmatrix} (0.2+1-1+0)/4 & (0+0.4-0.1-0.1)/4 \\ (0.1+0+0.4-0.7)/4 & (-1-0.5-0.5+1)/4 \end{bmatrix} = \begin{bmatrix} 0.05 & 0.05 \\ -0.05 & -0.25 \end{bmatrix}$, ii) $\begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}$, iii) $\begin{bmatrix} 1 & 1 \\ 0.4 & 1 \end{bmatrix}$.

Exercise 9. The input to a convolutional layer of a CNN consists of 6 feature maps each of which has a height of 11 and width of 15, i.e., input is $11 \times 15 \times 6$. What size is the output produced by a single mask with 6 channels,a width of 3 and a height of 3, i.e., $3 \times 3 \times 6$ when using a stride of 2 and padding of 0.

Solution 9. The output size will be $5 \times 7 \times 1$:



In general

$$\boxed{\text{output}_{\text{dim}} = 1 + \frac{(\text{input}_{\text{dim}} - \text{mask}_{\text{dim}} + 2 \times \text{padding})}{\text{stride}}}.$$

So for this example:

$$\text{output}_{\text{width}} = 1 + \frac{(15 - 3 + 2 \times 0)}{2} = 1 + 6 = 7,$$
$$\text{output}_{\text{height}} = 1 + \frac{(11 - 3 + 2 \times 0)}{2} = 1 + 4 = 5.$$

Exercise 10. A CNN processes an image of size $200 \times 200 \times 3$ using the following sequence of layers

1) convolution with 40 masks of size $5 \times 5 \times 3$ with stride $= 1$, and padding $= 0$,
2) pooling with 2×2 pooling regions and stride $= 2$,
3) convolution with 80 masks of size 4×4 withs stride $= 2$, and padding $= 1$,
4) 1×1 convolution with 20 masks.

What is the size of the output once it has been flattened?

Solution 10. Using $\text{output}_{\text{dim}} = 1 + \frac{(\text{input}_{\text{dim}} - \text{mask}_{\text{dim}} + 2 \times \text{padding})}{\text{stride}}$:

1) – $\text{output}_{\text{width}} = \text{output}_{\text{height}} = 1 + \frac{200-5}{1} = 196$.
   – number of channels $= 40$ (as there are 40 masks)
   – so size of output is: $\boxed{196 \times 196 \times 40.}$

2) – $\text{output}_{\text{width}} = \text{output}_{\text{height}} = 1 + \frac{196-2}{2} = 98$.
   – number of channels $= 40$ (as pooling does not change this)
   – so size of output is: $\boxed{98 \times 98 \times 40.}$

3) – $\text{output}_{\text{width}} = \text{output}_{\text{height}} = 1 + \frac{98-4+2}{2} = 49$.
   – number of channels $= 80$ (as there are 80 masks)
   – so size of output is: $\boxed{49 \times 49 \times 80.}$

4) – $\text{output}_{\text{width}} = \text{output}_{\text{height}} = 49$ (as 1×1 convolution does not change this).
   – number of channels $= 20$ (as there are 20 masks)
   – so size of output is: $\boxed{49 \times 49 \times 20.}$

After flattening, the length of the feature vector is: $49 \times 49 \times 20 = \boxed{48020.}$

Exercise 11. The following images show exemplars from two datasets.

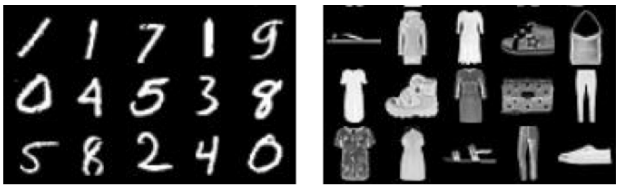


Figure 1: MNIST (left) and FashionMNIST (right).

Each dataset is to be expanded using data augmentation. Which of the following transformations are appropriate:

i) rescaling
ii) horizontal flip
iii) rotation
iv) cropping.

Solution 11.

	MNIST	FashionMNIST
i) rescaling	yes	yes
ii) horizontal flip	no	yes
iii) rotations	*small rotations would be ok*	yes (*if we want the classifier to recognise up-side-down clothes*)
iv) cropping	yes	yes (*assuming objects remain recognisable after crop*)

7CCSMPNN: Pattern Recognition, Neural Networks and Deep Learning

KING'S College LONDON

Tutorial - 6: Generative adversarial networks (GANs)

Instructors: David Watson, Stefanos Leonardos *Term: Spring 2025 – 24-Feb*

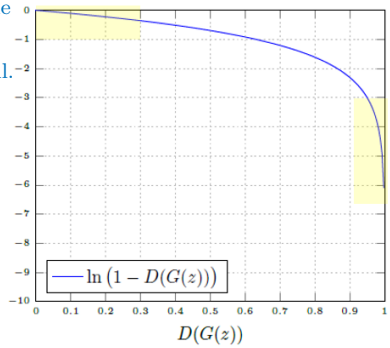
Exercise 1. Vanishing gradient is an issue when training Generative Adversarial Networks (GANs). In the literature, when training the generator, it is recommended to consider the alternative cost function: $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[-\log\left(D(G(\mathbf{z}))\right)\right]$.

i) What is the advantage of using this alternative cost function over the original one, i.e., $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[\log\left(1-D(G(\mathbf{z}))\right)\right]$?
ii) Write the pseudo-code for training a GAN with this alternative cost function.

Solution 1.

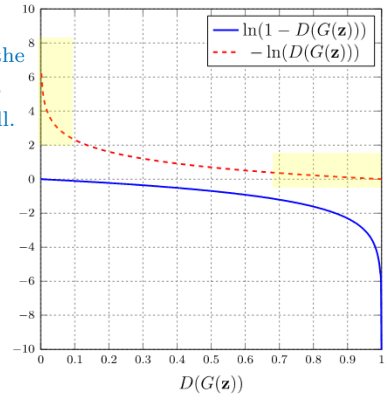
i) The alternative cost helps to alleviate the vanishing gradient problem when it is mostly needed as shown in the figures below.

Flat: When the generator does NOT work well.



Steep: When the generator works well.

Steep: When the generator does NOT work well.



Flat: When the generator works well.

ii) The pseudo-code for training a GAN with the alternative cost function,

$$V(D,G)=\underbrace{\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log\left(D(\mathbf{x})\right)]}_{\text{for real samples}}+\underbrace{\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[-\log\left(D(\mathbf{z})\right)]}_{\text{for fake samples}}$$

is provided below. The number k of update steps of the discriminator in every iteration is a hyperparameter.

Algorithm 1 Minibatch stochastic GD training of GANs

Initialise: number of steps k, size of minibatch m.
for number of iterations **do**
  for k steps **do**
    • sample minibatch of m noise samples $\{\mathbf{z}_1,\dots,\mathbf{z}_m\}$ from noise prior $p_{\mathbf{z}}(\mathbf{z})$.
    • sample minibatch of m examples $\{\mathbf{x}_1,\dots,\mathbf{x}_m\}$ from data distribution $p_{\text{data}}(\mathbf{x})$.
    • update the **discriminator** using stochastic gradient **ascent** with:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log\left(D(\mathbf{x}_i)\right)-\log\left(D(G(\mathbf{z}_i))\right)\right].$$

  end for
  • sample minibatch of m noise samples $\{\mathbf{z}_1,\dots,\mathbf{z}_m\}$ from noise distribution $p_{\mathbf{z}}(\mathbf{z})$.
  • update the **generator** using stochastic gradient **descent** with:

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\left[-\log\left(D(G(\mathbf{z}_i))\right)\right].$$

end for
Remark: the gradient-based updates can use **any** gradient-based learning rule.

Exercise 2. The training of GANs can be formulated as an optimisation problem as shown below

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1-D(G(\mathbf{z})))].$$

i) Given a generator, G, with output distribution $p_{\text{gen}}(\mathbf{x})$, find the value of the optimal discriminator $D^*(\mathbf{x})$, for each $\mathbf{x} \in \mathbf{X}$. Assume that $p_{\text{data}}(\mathbf{x}), p_{\text{gen}}(\mathbf{x}) > 0$ for all $\mathbf{x} \in \mathbf{X}$.
ii) What is the output distribution, $g^*_{\text{gen}}(\mathbf{x})$, of an optimal generator G^*. Find $D^*(\mathbf{x})$, for $\mathbf{x} \in \mathbf{X}$, against G^*.
iii) Determine the value of the loss function, $V(D^*, G^*)$, when both generator and discriminator are optimal.

Solution 2.

i) Recall that the generator induces a distribution, $p_{\text{gen}(\mathbf{x})}$, of fake (or generated) samples in the sample space $\mathbf{X}$. Thus, when $\mathbf{x}$ is a discrete variable[1], i.e., $\mathbf{x} \in \mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we have that

$$\begin{aligned} V(D,G) &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1-D(G(\mathbf{z})))] \\ &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{x}\sim p_{\text{gen}}(\mathbf{x})}[\log(1-D(\mathbf{x}))] \\ &= \sum_{\mathbf{x}\in\mathbf{X}} p_{\text{data}}(\mathbf{x})\log(D(\mathbf{x})) + \sum_{\mathbf{x}\in\mathbf{X}} p_{\text{gen}}(\mathbf{x})\log(1-D(\mathbf{x})) \\ &= \sum_{\mathbf{x}\in\mathbf{X}} \left[p_{\text{data}}(\mathbf{x})\log(D(\mathbf{x})) + p_{\text{gen}}(\mathbf{x})\log(1-D(\mathbf{x}))\right]. \end{aligned}$$

The optimal D is now obtained by setting the partial derivative of $V(D,G)$ with respect to D equal to 0 for every individual $\mathbf{x} \in \mathbf{X}$ (why?). Thus,

$$\begin{aligned} \frac{\partial}{\partial D(\mathbf{x})} V(D,G) &= \frac{\partial}{\partial D(\mathbf{x})}\left[p_{\text{data}}(\mathbf{x})\log(D(\mathbf{x})) + p_{\text{gen}}(\mathbf{x})\log(1-D(\mathbf{x}))\right] \\ &= p_{\text{data}}(\mathbf{x})\cdot\frac{1}{D(\mathbf{x})} - p_{\text{gen}}(\mathbf{x})\cdot\frac{1}{1-D(\mathbf{x})} \\ &= \frac{1}{D(\mathbf{x})(1-D(\mathbf{x}))}\cdot\left[p_{\text{data}}(\mathbf{x}) - D(\mathbf{x})(p_{\text{data}}(\mathbf{x}) + p_{\text{gen}}(\mathbf{x}))\right]. \end{aligned}$$

Setting $\frac{\partial}{\partial D(\mathbf{x})}V(D,G) = 0$, we obtain the condition: $\boxed{D^*(\mathbf{x}) = \frac{p_{\text{data}(\mathbf{x})}}{p_{\text{data}(\mathbf{x})} + p_{\text{gen}(\mathbf{x})}}}$. The assumption $p_{\text{data}}(\mathbf{x})$, $p_{\text{gen}}(\mathbf{x}) > 0$ for all $\mathbf{x} \in \mathbf{X}$, ensures that there are no points that are never visited during training, and which can, thus, have undefined behavior.

ii) The optimal output distribution, $p^*_{\text{gen}}(\mathbf{x})$ is $p^*_{\text{gen}}(\mathbf{x}) = p_{\text{data}}(\mathbf{x})$ for $\mathbf{x} \in \mathbf{X}$. It happens when the generator reproduces the real data distribution. Using i), we find the value of the optimal discriminator, D^* is

$$D^*(\mathbf{x}) = \frac{p_{\text{data}(\mathbf{x})}}{p_{\text{data}(\mathbf{x})} + p_{\text{gen}(\mathbf{x})}} = \frac{p_{\text{data}(\mathbf{x})}}{2\cdot p_{\text{data}(\mathbf{x})}} = \frac{1}{2}, \quad \text{for all } \mathbf{x} \in \mathbf{X}.$$

iii) When both generator and discriminator are optimal, we have that

$$\begin{aligned} V(D^*,G^*) &= \sum_{\mathbf{x}\in\mathbf{X}} \left[p_{\text{data}}(\mathbf{x})\log D^*(\mathbf{x}) + p^*_{\text{gen}}(\mathbf{x})\log(1-D^*(\mathbf{x}))\right] \\ &= \sum_{\mathbf{x}\in\mathbf{X}} \left[p_{\text{data}}(\mathbf{x})\log\left(\frac{1}{2}\right) + p_{\text{data}}(\mathbf{x})\log\left(1-\frac{1}{2}\right)\right] \\ &= 2\log\left(\frac{1}{2}\right)\underbrace{\sum_{\mathbf{x}\in\mathbf{X}} p_{\text{data}}(\mathbf{x})}_{=1} \\ &= -2\log 2. \end{aligned}$$

Exercise 3. When training a GAN, we consider a dataset consisting of real, $\mathbf{X}_{\text{real}} = \{\mathbf{x}_1, \mathbf{x}_2\}$, and generated samples, $\mathbf{X}_{\text{fake}} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2\}$, where: $\mathbf{x}_1 = \begin{pmatrix}1\\2\end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix}3\\4\end{pmatrix}, \tilde{\mathbf{x}}_1 = \begin{pmatrix}5\\6\end{pmatrix}, \tilde{\mathbf{x}}_2 = \begin{pmatrix}7\\8\end{pmatrix}$. The discriminator is given by

$$D(\mathbf{x}) = \frac{1}{1+e^{-(\theta_{d_1}x_1 - \theta_{d_2}x_2 - 2)}},$$

[1] The result is the same when $\mathbf{x}$ is a continuous variable. The main difference is that there will be integrals rather than summations in the calculations.

where $\theta_{d_1} = 0.1, \theta_{d_2} = 0.2$ are the parameters of the discriminator, and $\mathbf{x} = (x_1, x_2)^T$. Assume that each sample from the (real and fake) dataset has equal probability to be selected.

i) Given the datasets $\mathbf{X}_{\text{real}}$, and $\mathbf{X}_{\text{fake}}$ compute

$$V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))].$$

ii) Assuming that all real and fake samples are selected in the (mini)batch and that $k = 1$ for GAN training, compute

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(D\left(\mathbf{x}_i\right)\right) + \log\left(1 - D\left(G\left(\mathbf{z}_i\right)\right)\right)\right],$$

and determine the updated θ_{d_1} and θ_{d_2} for the next iteration using learning rate $\eta = 0.02$.

Solution 3.

i) In this case, we have two samples in each dataset, so our sample is discrete. Thus, to calculate the expectations in the definition of $V(D,G)$, we will use summations rather than integrals as in the previous exercise. To calculate $V(D,G)$, it will also be helpful to evaluate $D(\mathbf{x}_i)$ for all $\mathbf{x}_i = (x_{i1}, x_{i2})^T \in \mathbf{X}_{\text{real}}$ and all $\tilde{\mathbf{x}}_i = (\tilde{x}_{i1}, \tilde{x}_{i2})^T \in \mathbf{X}_{\text{fake}}$:

$$\begin{aligned} D(\mathbf{x}_1) &= \frac{1}{1 + e^{-(\theta_{d_1}\cdot x_{11} - \theta_{d_2}\cdot x_{12} - 2)}} = \frac{1}{1 + e^{-(0.1\cdot 1 - 0.2\cdot 2 - 2)}} = 0.0911, \\ D(\mathbf{x}_2) &= 0.0759, \\ D(\tilde{\mathbf{x}}_1) &= 0.0630, \\ D(\tilde{\mathbf{x}}_2) &= 0.0522. \end{aligned} \tag{1}$$

- For the first term, $\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(D(\mathbf{x}))]$ in $V(D,G)$, we have that

$$\begin{aligned} \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(D(\mathbf{x}))] &= \sum_{\mathbf{x}_i \in \mathbf{X}_{\text{real}}} p_{\text{data}}(\mathbf{x}_i)\log(D(\mathbf{x}_i)) \\ &= \underbrace{p_{\text{data}}(\mathbf{x}_1)}_{=1/2}\log(D(\mathbf{x}_1)) + \underbrace{p_{\text{data}}(\mathbf{x}_2)}_{=1/2}\log(D(\mathbf{x}_2)) && \text{(samples are equiprobable)} \\ &= \frac{1}{2}\log(0.0911) + \frac{1}{2}\log(0.0759) && \text{using (1)} \\ &= -1.1978 - 1.2894 = -2.4872. \end{aligned}$$

- For the second term, $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$ in $V(D,G)$, we have that

$$\begin{aligned} \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] = \mathbb{E}_{\tilde{\mathbf{x}}\sim p_{\text{gen}}(\tilde{\mathbf{x}})}[\log(1 - D(\tilde{\mathbf{x}}))] &= \sum_{\mathbf{x}_i \in \mathbf{X}_{\text{fake}}} p_{\text{gen}}(\tilde{\mathbf{x}}_i)\log(1 - D(\tilde{\mathbf{x}}_i)) \\ &= \underbrace{p_{\text{gen}}(\tilde{\mathbf{x}}_1)}_{=1/2}\log(1 - D(\tilde{\mathbf{x}}_1)) + \underbrace{p_{\text{gen}}(\tilde{\mathbf{x}}_2)}_{=1/2}\log(1 - D(\tilde{\mathbf{x}}_2)) && \text{(samples are equiprobable)} \\ &= \frac{1}{2}\log(1 - 0.0630) + \frac{1}{2}\log(1 - 0.0522) && \text{using (1)} \\ &= -0.0325 - 0.0268 = -0.0593. \end{aligned}$$

Putting the two expressions together, we have that

$$V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] = -2.4872 - 0.0593 = -2.5465.$$

ii) To update the parameters of the generator, $\theta_{d_1}, \theta_{d_2}$ using the GAN training algorithm (see Algorithm 1 but with the original cost function as in the lecture slides) we first need to calculate the gradient of the loss function, $V(D,G)$, with respect to θ_d. We have that:

$$\nabla_{\theta_d} V(D,G) = \begin{pmatrix} \frac{\partial}{\partial\theta_{d_1}} V(D,G) \\ \frac{\partial}{\partial\theta_{d_2}} V(D,G) \end{pmatrix}.$$

Considering the first partial derivative with respect to θ_{d_1}, we have that

$$\begin{aligned}
\frac{\partial}{\partial\theta_{d_1}}V(D,G) &= \frac{\partial}{\partial\theta_{d_1}}\left(\frac{1}{m}\sum_{i=1}^{m}\left[\log\left(D(\mathbf{x}_i)\right)+\log\left(1-D(G(\mathbf{z}_i))\right)\right]\right)\\
&= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial\theta_{d_1}}\left[\log\left(D(\mathbf{x}_i)\right)+\log\left(1-D(G(\mathbf{z}_i))\right)\right]\\
&= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial}{\partial\theta_{d_1}}\log\left(D(\mathbf{x}_i)\right)+\frac{\partial}{\partial\theta_{d_1}}\log\left(1-D(G(\mathbf{z}_i))\right)\right]\\
&= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial}{\partial\theta_{d_1}}\log\left(D(\mathbf{x}_i)\right)+\frac{\partial}{\partial\theta_{d_1}}\log\left(1-D(\tilde{\mathbf{x}}_i)\right)\right]\\
&= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{1}{D(\mathbf{x}_i)}\cdot\frac{\partial}{\partial\theta_{d_1}}D(\mathbf{x}_i)-\frac{1}{1-D(\tilde{\mathbf{x}}_i)}\cdot\frac{\partial}{\partial\theta_{d_1}}D(\tilde{\mathbf{x}}_i)\right] \qquad (2)
\end{aligned}$$

To proceed, we need to calculate the partial derivative of $D(\mathbf{x})$ with respect to θ_{d_1}. We have that

$$\begin{aligned}
\frac{\partial}{\partial\theta_{d_1}}D(\mathbf{x}) &= \frac{\partial}{\partial\theta_{d_1}}\left(\frac{1}{1+e^{-(\theta_{d_1}x_1-\theta_{d_2}x_2-2)}}\right)=\frac{x_1e^{-(\theta_{d_1}x_1-\theta_{d_2}x_2-2)}}{\left(1+e^{-(\theta_{d_1}x_1-\theta_{d_2}x_2-2)}\right)^2}\\
&= x_1\cdot\frac{1}{1+e^{-(\theta_{d_1}x_1-\theta_{d_2}x_2-2)}}\cdot\frac{e^{-(\theta_{d_1}x_1-\theta_{d_2}x_2-2)}}{1+e^{-(\theta_{d_1}x_1-\theta_{d_2}x_2-2)}}\\
&= x_1\cdot D(\mathbf{x})\cdot\left(1-\frac{1}{1+e^{-(\theta_{d_1}x_1-\theta_{d_2}x_2-2)}}\right)\\
&= x_1\cdot D(\mathbf{x})\cdot\left(1-D(\mathbf{x})\right). \qquad (3)
\end{aligned}$$

Thus, substituting (3) back in (2), we have that

$$\begin{aligned}
\frac{\partial}{\partial\theta_{d_1}}V(D,G) &= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{1}{D(\mathbf{x}_i)}\cdot\frac{\partial}{\partial\theta_{d_1}}D(\mathbf{x}_i)-\frac{1}{1-D(\tilde{\mathbf{x}}_i)}\cdot\frac{\partial}{\partial\theta_{d_1}}D(\tilde{\mathbf{x}}_i)\right]\\
&= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{x_{i1}\cdot D(\mathbf{x}_i)\cdot(1-D(\mathbf{x}_i))}{D(\mathbf{x}_i)}-\frac{\tilde{x}_{i1}\cdot D(\tilde{\mathbf{x}}_i)\cdot(1-D(\tilde{\mathbf{x}}_i))}{1-D(\tilde{\mathbf{x}}_i)}\right]\\
&= \frac{1}{m}\sum_{i=1}^{m}\left[x_{i1}(1-D(\mathbf{x}_i))-\tilde{x}_{i1}D(\tilde{\mathbf{x}}_i)\right]
\end{aligned}$$

Since all samples are selected in the minibatch, we have that $m=2$. Thus, using (1), we have that

$$\begin{aligned}
\frac{\partial}{\partial\theta_{d_1}}V(D,G) &= \frac{1}{m}\sum_{i=1}^{m}\left[x_{i1}(1-D(\mathbf{x}_i))-\tilde{x}_{i1}D(\tilde{\mathbf{x}}_i)\right]\\
&= \frac{1}{2}\sum_{i=1}^{2}\left[x_{i1}(1-D(\mathbf{x}_i))-\tilde{x}_{i1}D(\tilde{\mathbf{x}}_i)\right]\\
&= \frac{1}{2}\left[1\cdot(1-0.0911)+3\cdot(1-0.0759)-5\cdot0.0630-7\cdot0.0522\right]\\
&= 1.5007.
\end{aligned}$$

Repeating the above process for $\frac{\partial}{\partial\theta_{d_2}}V(D,G)$, we find that: $\frac{\partial}{\partial\theta_{d_2}}V(D,G)=-2.3596$. Thus, the updated $\theta_{d_1},\theta_{d_2}$ usng the gradient ascent rule with $\eta=0.02$ are

$$\begin{aligned}
\begin{pmatrix}\theta_{d_1}\\ \theta_{d_2}\end{pmatrix}^{\text{new}} &= \begin{pmatrix}\theta_{d_1}\\ \theta_{d_2}\end{pmatrix}^{\text{old}}+\eta\cdot\nabla_{\theta_d}V(D,G)\\
&= \begin{pmatrix}\theta_{d_1}\\ \theta_{d_2}\end{pmatrix}^{\text{old}}+\eta\cdot\begin{pmatrix}\frac{\partial}{\partial\theta_{d_1}}V(D,G)\\ \frac{\partial}{\partial\theta_{d_2}}V(D,G)\end{pmatrix}\\
&= \begin{pmatrix}0.1\\ 0.2\end{pmatrix}+0.02\cdot\begin{pmatrix}1.5007\\ -2.3596\end{pmatrix}=\begin{pmatrix}0.13001\\ 0.15281\end{pmatrix}.
\end{aligned}$$

Thus, the new updated parameters are $\theta_{d_1}=0.13001,\theta_{d_2}=0.15281$.

7CCSMPNN: Pattern Recognition, Neural Networks and Deep Learning

KING'S College LONDON

Tutorial - 7: Feature extraction

Instructors: David Watson, Stefanos Leonardos *Term: Spring 2025 – 03-Mar*

Exercise 1. Give brief definitions of the following terms

i) feature engineering
ii) feature selection
iii) feature extraction
iv) dimensionality reduction
v) deep learning

Solution 1.

i) feature engineering: modifying measured values to make them suitable for classification.
ii) feature selection: choosing a subset of measured/possible features to use for classification.
iii) feature extraction: projecting the chosen feature vectors into a new feature space.
iv) dimensionality reduction: selecting/extracting feature vectors of lower dimensionality (length) than the original feature vectors.
v) deep learning: the process of training a neural network that has many layers (> 3, typically $\gg 3$), or performing multiple stages of (nonlinear) feature extraction prior to classification.

Exercise 2. List five methods that can be used to perform feature extraction.

Solution 2.

i) Principal Component Analysis (PCA)
ii) Whitening
iii) Linear Discriminant Analysis (LDA)
iv) Independent Component Analysis (ICA)
v) Random Projections
vi) Sparse Coding

Exercise 3. Write pseudo-code for the Karhunen-Loève Transform (KLT) method for performing Principal Component Analysis (PCA).

Solution 3.

Algorithm 1 Karhunen-Loève Transform (KLT)

```
for all data vectors: do
    subtract the mean
calculate the covariance matrix of the zero-mean data.
find the eigenvalues and eigenvectors of the covariance matrix.
for all eigenvalues: do
    order eigenvalues from large to small
    discard small eigenvalues and their respective vectors.
form a matrix (V̂) of the remaining eigenvectors.
project the zero-mean data onto the PCA subspace by multiplying by the transpose of V̂.
```

Exercise 4. Use the KLT method to project the following 3-dimensional data set onto the first 2 principal components: $\mathbf{x_1} = (1,2,1)^T, \mathbf{x_2} = (2,3,1)^T, \mathbf{x_3} = (3,5,1)^T, \mathbf{x_4} = (2,2,1)^T$. Hint: The MATLAB command `eig` can be used to find the eigenvectors and eigenvalues.

Solution 4. The mean of the data is $\mu^T = (2,3,1)$, hence, the zero-mean data is: $\mathbf{x}'_1 = (-1,-1,0)^T, \mathbf{x}'_2 = (0,0,0)^T, \mathbf{x}'_3 = (1,2,0)^T, \mathbf{x}'_4 = (0,-1,0)^T$. The covariance matrix is

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}'_i)(\mathbf{x}'_i)^T$$

$$= \frac{1}{4}\left[\begin{pmatrix}-1\\-1\\0\end{pmatrix}(-1,-1,0) + \begin{pmatrix}0\\0\\0\end{pmatrix}(0,0,0) + \begin{pmatrix}1\\2\\0\end{pmatrix}(1,2,0) + \begin{pmatrix}0\\-1\\0\end{pmatrix}(0,-1,0)\right]$$
$$= \frac{1}{4}\left[\begin{pmatrix}1&1&0\\1&1&0\\0&0&0\end{pmatrix} + \begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix} + \begin{pmatrix}1&2&0\\2&4&0\\0&0&0\end{pmatrix} + \begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix}\right] = \frac{1}{4}\begin{pmatrix}2&3&0\\3&6&0\\0&0&0\end{pmatrix}.$$

We calculate eigenvectors and eigenvalues of $\mathbf{C}$ using MATLAB command `eig`:

$$\mathbf{V} = \begin{pmatrix}0 & -0.8817 & 0.4719\\0 & 0.4719 & 0.8817\\1 & 0 & 0\end{pmatrix}, \qquad \mathbf{E} = \begin{pmatrix}0 & 0 & 0\\0 & 0.0986 & 0\\0 & 0 & 1.9015\end{pmatrix}.$$

We next choose the two eigenvectors corresponding to the two largest eigenvectors: $\hat{\mathbf{V}} = \begin{pmatrix}0.4719 & -0.8817\\0.8817 & 0.4719\\0 & 0\end{pmatrix}$. The projection of the data onto the subspace spanned by the first two principal components is given by:

$$\mathbf{y}_i = \hat{\mathbf{V}}^T(\mathbf{x}_i - \mu) = \begin{pmatrix}0.4719 & 0.8817 & 0\\-0.8817 & 0.4719 & 0\end{pmatrix}\mathbf{x}_i'.$$

Thus,

$\mathbf{x}_1'$	$\mathbf{y}_1'$	$\mathbf{x}_2'$	$\mathbf{y}_2$	$\mathbf{x}_3'$	$\mathbf{y}_3$	$\mathbf{x}_4'$	$\mathbf{y}_4$
$\begin{pmatrix}-1\\-1\\0\end{pmatrix}$	$\begin{pmatrix}-1.3536\\0.4098\end{pmatrix}$	$\begin{pmatrix}0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}1\\2\\0\end{pmatrix}$	$\begin{pmatrix}2.2353\\0.0621\end{pmatrix}$	$\begin{pmatrix}0\\-1\\0\end{pmatrix}$	$\begin{pmatrix}-0.8817\\-0.4719\end{pmatrix}$

Note: the new data, $\mathbf{y}$, has zero mean and the covariance matrix is: $\begin{pmatrix}1.9015 & 0\\0 & 0.0986\end{pmatrix}$. The eigenvalues of the original covariance matrix measure variance along each principal component.

Exercise 5. What is the proportion of variance explained by the first 2 principal components in the previous exercise?

Solution 5. The proportion of the variance is given by sum of eigenvalues for selected components divided by the sum of all eigenvalues. For the preceding question this is

$$\frac{1.9015 + 0.0986}{1.9015 + 0.0986 + 0} = 1.$$

Note: the first principal component alone would explain $\frac{1.9015}{1.9015 + 0.0986 + 0} = 0.95$ of the variance, so we could project data onto only the first PC without losing too much information.

Exercise 6. Use the KLT method to project the 2-dimensional data set, $\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}3\\5\end{pmatrix}, \begin{pmatrix}5\\4\end{pmatrix}, \begin{pmatrix}5\\6\end{pmatrix}, \begin{pmatrix}8\\7\end{pmatrix}, \begin{pmatrix}9\\7\end{pmatrix}$, onto the first principal component. (Hint: the MATLAB command `eig` can be used to find the eigenvectors and eigenvalues).

Solution 6. The mean of the data is $\mu = \begin{pmatrix}5\\5\end{pmatrix}$, hence, the zero-mean data is: $\begin{pmatrix}-5\\-4\end{pmatrix}, \begin{pmatrix}-2\\0\end{pmatrix}, \begin{pmatrix}0\\-1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}3\\2\end{pmatrix}, \begin{pmatrix}4\\2\end{pmatrix}$. The covariance matrix is

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$$
$$= \frac{1}{6}\left[\begin{pmatrix}-5\\-4\end{pmatrix}(-5,-4) + \begin{pmatrix}-2\\0\end{pmatrix}(-2,0) + \begin{pmatrix}0\\-1\end{pmatrix}(0,-1) + \begin{pmatrix}0\\1\end{pmatrix}(0,1) + \begin{pmatrix}3\\2\end{pmatrix}(3,2) + \begin{pmatrix}4\\2\end{pmatrix}(4,2)\right]$$
$$= \frac{1}{6}\left[\begin{pmatrix}25&20\\20&16\end{pmatrix} + \begin{pmatrix}4&0\\0&0\end{pmatrix} + \begin{pmatrix}0&0\\0&1\end{pmatrix} + \begin{pmatrix}0&0\\0&1\end{pmatrix} + \begin{pmatrix}9&6\\6&4\end{pmatrix} + \begin{pmatrix}16&8\\8&4\end{pmatrix}\right]$$
$$= \frac{1}{6}\begin{pmatrix}54&34\\34&26\end{pmatrix} = \begin{pmatrix}9&5.67\\5.67&4.33\end{pmatrix}.$$

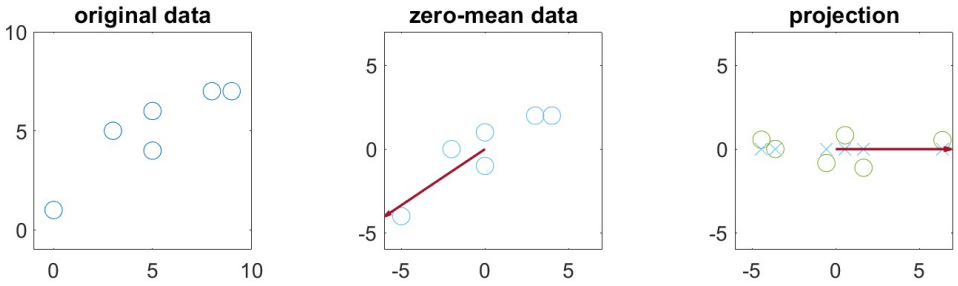
We calculate eigenvectors and eigenvalues of $\mathbf{C}$ using MATLAB command `eig`:

$$\mathbf{V} = \begin{pmatrix} 0.5564 & -0.8309 \\ -0.8309 & -0.5564 \end{pmatrix}, \mathbf{E} = \begin{pmatrix} 0.5384 & 0 \\ 0 & 12.7949 \end{pmatrix}.$$

We next choose the eigenvector corresponding to the largest eigenvector: $\hat{\mathbf{V}} = \begin{pmatrix} -0.8309 \\ -0.5564 \end{pmatrix}$. The projection of the data onto the subspace spanned by the first principal component is given by:

$$\mathbf{y}_i = \hat{\mathbf{V}}^T(\mathbf{x}_i - \mu) = (-0.8309, -0.5564)\left(\mathbf{x}_i - \begin{pmatrix} 5 \\ 5 \end{pmatrix}\right).$$

Therefore, the new dataset is: $6.3801, 1.6618, 0.5564, -0.5564, -3.6055, -4.4364$. The projection looks like this:



Exercise 7. Apply two epochs of a batch version of Oja's learning rule to the same data used in the previous exercise. Use a learning rate of 0.01, and an initial weight vector of $(-1, 0)$.

Solution 7. For the batch Oja's learning rule, weights are updated as: $\mathbf{w} \leftarrow \mathbf{w} + \eta \sum_i y(\mathbf{x}_i^t - y\mathbf{w})$. Here, $\eta = 0.01$.

- Epoch 1: initial $\mathbf{w} = (-1, 0)$.

$\mathbf{x}^t$	$y = \mathbf{wx}$	$\mathbf{x}^t - y\mathbf{w}$	$\eta y(\mathbf{x}^t - y\mathbf{w})$
$(-5, -4)$	5	$(0, -4)$	$(0, -0.20)$
$(-2, 0)$	2	$(0, 0)$	$(0, 0)$
$(0, -1)$	0	$(0, -1)$	$(0, 0)$
$(0, 1)$	0	$(0, 1)$	$(0, 0)$
$(3, 2)$	-3	$(0, 2)$	$(0, -0.06)$
$(4, 2)$	-4	$(0, 2)$	$(0, -0.08)$
		total weight change	$(0, -0.34)$
		$\mathbf{w}$	$(-1, -0.34)$

- Epoch 2: $\mathbf{w} = (-1, -0.34)$.

$\mathbf{x}^t$	$y = \mathbf{wx}$	$\mathbf{x}^T - y\mathbf{w}$	$\eta y(\mathbf{x}^T - y\mathbf{w})$
$(-5, -4)$	6.36	$(1.36, -1.84)$	$(0.087, -0.117)$
$(-2, 0)$	2	$(0, 0.68)$	$(0, 0.014)$
$(0, -1)$	0.34	$(0.34, -0.88)$	$(0.001, -0.003)$
$(0, 1)$	-0.34	$(-0.34, 0.88)$	$(0.001, -0.003)$
$(3, 2)$	-3.68	$(-0.68, 0.75)$	$(0.025, -0.028)$
$(4, 2)$	-4.68	$(-0.68, 0.41)$	$(0.318, -0.019)$
		total weight change	$(0.146, -0.156)$
		$\mathbf{w}$	$(-0.854, -0.496)$

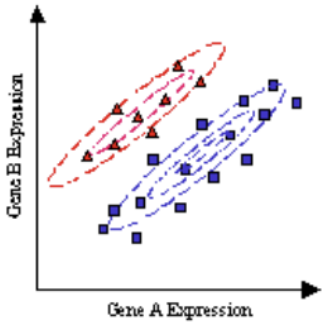
Repeating the above process, we have that

- after 3 epochs: $\mathbf{w} = (-0.8468, -0.5453)$.
- after 4 epochs: $\mathbf{w} = (-0.8302, -0.5505)$.
- after 5 epochs: $\mathbf{w} = (-0.8333, -0.5565)$.
- after 6 epochs: $\mathbf{w} = (-0.8302, -0.5556)$.

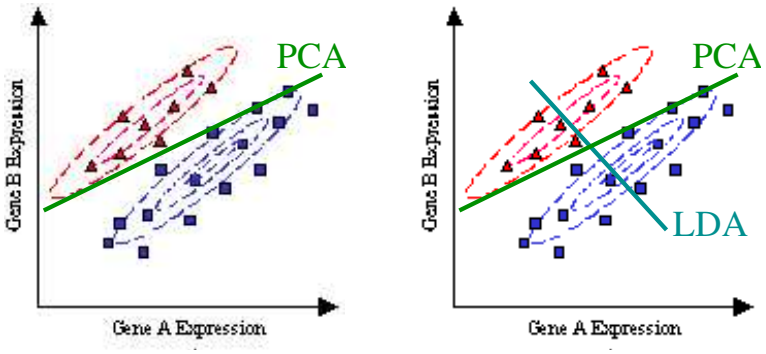
Compare the result for previous question when calculating first PC via the Karhunen-Loève Transform.

Exercise 8. The graph below shows a 2-dimensional data set in which exemplars come from two classes. Exemplars from the two classes are plotted using triangular and square markers, respectively.
Draw the approximate direction of

i) the first principal component of this data,
ii) the axis onto which the data will be projected using Linear Discriminant Analysis (LDA).



Solution 8.



Exercise 9. Briefly describe the optimisation method performed by Fisher's LDA to find a projection of the original data onto a subspace.

Solution 9. LDA searches for a discriminative subspace in which exemplars belonging to the same class are as close together as possible while patterns belonging to different classes are as far apart as possible.

Exercise 10. For the data shown below use Fisher's LDA method to determine which of the following projection

feature vector $\mathbf{x}^T$	$(1,2)$	$(2,1)$	$(3,3)$	$(6,5)$	$(7,8)$
class	1	1	1	2	2

weights is more effective at performing LDA: i) $\mathbf{w}^T = (-1, 5)$, ii) $\mathbf{w}^T = (2, -3)$.

Solution 10. Fisher's LDA maximises the cost function $J(\mathbf{w}) = \frac{sb}{sw}$, where:

- between class scatter: $sb = |\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)|^2$,
- within class scatter: $sw = \sum_{\mathbf{x}\in\omega_1}(\mathbf{w}^T(\mathbf{x} - \mathbf{m}_1))^2 + \sum_{\mathbf{x}\in\omega_2}(\mathbf{w}^T(\mathbf{x} - \mathbf{m}_2))^2$,
- sample means: $\mathbf{m}_i = \frac{1}{n_i}\sum_{x\in\omega_i} x, \quad \text{for } i = 1, 2.$

The sample means for classes 1 and 2 are: $\mathbf{m}_1^T = \frac{1}{3}\left((1,2) + (2,1) + (3,3)\right) = (2,2), \mathbf{m}_2^T = \frac{1}{2}\left((6,5) + (7,8)\right) = (6.5, 6.5)$.

- $\mathbf{w}^T = [-1, 5]$:

$$sb = \left|\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)\right|^2 = \left|(-1,5) \times \left[\begin{pmatrix}2\\2\end{pmatrix} - \begin{pmatrix}6.5\\6.5\end{pmatrix}\right]\right|^2 = \left|(-1,5) \times \begin{pmatrix}-4.5\\-4.5\end{pmatrix}\right|^2 = |-18|^2 = 324,$$

$$\begin{aligned} sw &= \sum_{\mathbf{x}\in\omega_1}(\mathbf{w}^T(\mathbf{x} - \mathbf{m}_1))^2 + \sum_{\mathbf{x}\in\omega_2}(\mathbf{w}^T(\mathbf{x} - \mathbf{m}_2))^2 \\ &= \left((-1,5) \times \left[\begin{pmatrix}1\\2\end{pmatrix} - \begin{pmatrix}2\\2\end{pmatrix}\right]\right)^2 + \left((-1,5) \times \left[\begin{pmatrix}2\\1\end{pmatrix} - \begin{pmatrix}2\\2\end{pmatrix}\right]\right)^2 + \left((-1,5) \times \left[\begin{pmatrix}3\\3\end{pmatrix} - \begin{pmatrix}2\\2\end{pmatrix}\right]\right)^2 + \\ &+ \left((-1,5) \times \left[\begin{pmatrix}6\\5\end{pmatrix} - \begin{pmatrix}6.5\\6.5\end{pmatrix}\right]\right)^2 + \left((-1,5) \times \left[\begin{pmatrix}7\\8\end{pmatrix} - \begin{pmatrix}6.5\\6.5\end{pmatrix}\right]\right)^2 = 140, \end{aligned}$$

$$J(\mathbf{w}) = \frac{sb}{sw} = \frac{324}{140} = 2.3143.$$

- $\mathbf{w}^T = [2, -3]$:

$$sb = |\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)|^2 = \left|(2,-3) \times \left[\begin{pmatrix}2\\2\end{pmatrix} - \begin{pmatrix}6.5\\6.5\end{pmatrix}\right]\right|^2 = \left|(2,-3) \times \begin{pmatrix}-4.5\\-4.5\end{pmatrix}\right|^2 = |4.5|^2 = 20.25,$$

$$\begin{aligned} sw &= \sum_{\mathbf{x}\in\omega_1}(\mathbf{w}^T(\mathbf{x} - \mathbf{m}_1))^2 + \sum_{\mathbf{x}\in\omega_2}(\mathbf{w}^T(\mathbf{x} - \mathbf{m}_2))^2 \\ &= \left((2,-3) \times \left[\begin{pmatrix}1\\2\end{pmatrix} - \begin{pmatrix}2\\2\end{pmatrix}\right]\right)^2 + \left((2,-3) \times \left[\begin{pmatrix}2\\1\end{pmatrix} - \begin{pmatrix}2\\2\end{pmatrix}\right]\right)^2 + \left((2,-3) \times \left[\begin{pmatrix}3\\3\end{pmatrix} - \begin{pmatrix}2\\2\end{pmatrix}\right]\right)^2 + \\ &+ \left((2,-3) \times \left[\begin{pmatrix}6\\5\end{pmatrix} - \begin{pmatrix}6.5\\6.5\end{pmatrix}\right]\right)^2 + \left((2,-3) \times \left[\begin{pmatrix}7\\8\end{pmatrix} - \begin{pmatrix}6.5\\6.5\end{pmatrix}\right]\right)^2 = 38.5, \end{aligned}$$

$$J(\mathbf{w}) = \frac{sb}{sw} = \frac{20.5}{38.5} = 0.526.$$

Since $J(\mathbf{w})$ given by $\mathbf{w}^T = (-1, 5)$ is higher, we conclude that it is a more effective projection weight. The projection of the data into the new feature space defined by the two projection weights is shown in the table. It can be seen,

		$y = \mathbf{w}^T\mathbf{x}$	
class	feature vector $\mathbf{x}^T$	$\mathbf{w}^T = (-1, 5)$	$\mathbf{w}^T = (2, -3)$
1	$(1, 2)$	$(-1, 5) \times (1, 2)^T = 9$	$(2, -3) \times (1, 2)^T = -4$
1	$(2, 1)$	$(-1, 5) \times (2, 1)^T = 3$	$(2, -3) \times (2, 1)^T = 1$
1	$(3, 3)$	$(-1, 5) \times (3, 3)^T = 12$	$(2, -3) \times (3, 3)^T = -3$
2	$(6, 5)$	$(-1, 5) \times (6, 5)^T = 19$	$(2, -3) \times (6, 5)^T = -3$
2	$(7, 8)$	$(-1, 5) \times (7, 8)^T = 33$	$(2, -3) \times (7, 8)^T = -10$

after projection by $\mathbf{w}^T = (-1, 5)$ the data is linearly separable, while after projection by $\mathbf{w}^T = (2, -3)$ it is not.

Exercise 11. An Extreme Learning Machine consists of a hidden layer with six neurons, and an output layer with one neuron. The weights to the hidden neurons have been assigned the following random values

$$\mathbf{V} = \begin{pmatrix} -0.62 & 0.44 & -0.91 \\ -0.81 & -0.09 & 0.02 \\ 0.74 & -0.91 & -0.60 \\ -0.82 & -0.92 & 0.71 \\ -.26 & 0.68 & 0.15 \\ 0.80 & -0.94 & -0.83 \end{pmatrix}.$$

The weights to the output neuron ar: $\mathbf{w}^T = (0, 0, 0, -1, 0, 0, 2)$. All weights are defined using augmented vector notation. Hidden neurons are *linear threshold* units, while the output neuron is linear. Calculate the response of the output neuron to each of the following input vectors: $\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

Solution 11. If we place all the augmented input patterns into a matrix we have the following dataset:

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

The response of each hidden neuron to a single exemplar is defined as $y = H(\mathbf{vx})$, where H is the heaviside function. The response of all six hidden neurons to all four input patterns is given by $\mathbf{Y} = H(\mathbf{VX})$, so that:

$$\mathbf{Y} = H\left(\begin{pmatrix} -0.62 & 0.44 & -0.91 \\ -0.81 & -0.09 & 0.02 \\ 0.74 & -0.91 & -0.60 \\ -0.82 & -0.92 & 0.71 \\ -0.26 & 0.68 & 0.15 \\ 0.80 & -0.94 & -0.83 \end{pmatrix}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}\right) = H\begin{pmatrix} -0.62 & -1.53 & -0.18 & -1.09 \\ -0.81 & -0.79 & -0.90 & -0.88 \\ 0.74 & 0.14 & -0.17 & -0.77 \\ -0.82 & -0.11 & -1.74 & -1.03 \\ -0.26 & -0.11 & 0.42 & 0.57 \\ 0.80 & -0.03 & -0.14 & -0.97 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Similarly, the response of the output neuron to a single exemplar is defined as $z = \mathbf{wy}$. The response of the output neuron to all four input patterns, is given by $\mathbf{Z} = \mathbf{wY} = (1, -1, 0, 0)$.

Exercise 12. Given a dictionary, $\mathbf{V}^T = \begin{pmatrix} 0.4 & 0.55 & 0.5 & -0.1 & -0.5 & 0.9 & 0.5 & 0.45 \\ -0.6 & -0.45 & -0.5 & 0.9 & -0.5 & 0.1 & 0.5 & 0.55 \end{pmatrix}$, what is the best sparse code for the signal, $\mathbf{x} = \begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix}$, out of the following two alternatives

i) $\mathbf{y_1}^T = (1, 0, 0, 0, 1, 0, 0, 0)$,
ii) $\mathbf{y_2}^T = (0, 0, 1, 0, 0, 0, -1, 0)$.

Assume that sparsity is measured as the count of elements that are non-zero.

Solution 12. Both alternatives are equally sparse (2 non-zero elements each), so the best will be the one with the lowest reconstruction error: $\|\mathbf{x} - \mathbf{V}^T\mathbf{y}\|_2$.

i) $\|\mathbf{x} - \mathbf{V}^T\mathbf{y_1}\|_2 = \left\|\begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} 0.4 - 0.5 \\ -0.6 - 0.5 \end{pmatrix}\right\|_2 = \left\|\begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} -0.1 \\ -1.1 \end{pmatrix}\right\|_2 = \left\|\begin{pmatrix} 0.05 \\ 0.15 \end{pmatrix}\right\|_2 = \sqrt{0.05^2 + 0.15^2} = 0.158.$

ii) $\|\mathbf{x} - \mathbf{V}^T\mathbf{y_1}\|_2 = \left\|\begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} 0.5 - 0.5 \\ -0.5 - 0.5 \end{pmatrix}\right\|_2 = \left\|\begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} 0 \\ -1 \end{pmatrix}\right\|_2 = \left\|\begin{pmatrix} -0.05 \\ 0.05 \end{pmatrix}\right\|_2 = \sqrt{0.05^2 + 0.05^2} = 0.071.$

Therefore, solution (ii) is the better sparse code.

Exercise 13. Repeat the previous exercise when the two alternatives are

i) $\mathbf{y_1}^T = (1, 0, 0, 0, 1, 0, 0, 0)$,
ii) $\mathbf{y_2}^T = (0, 0, 0, -1, 0, 0, 0, 0)$.

Solution 13. As we can see in the calculations below, the error this time is the same in both cases. Thus, we should prefer the sparser solution, which is solution (ii).

i) $\|\mathbf{x} - \mathbf{V}^T\mathbf{y_1}\|_2 = 0.158$. same as in the previous question.

ii) $\|\mathbf{x} - \mathbf{V}^T\mathbf{y_1}\|_2 = \left\| \begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} 0.1 \\ -0.9 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} -0.15 \\ 0.05 \end{pmatrix} \right\|_2 = \sqrt{0.15^2 + 0.05^2} = 0.158.$

7CCSMPNN: Pattern Recognition, Neural Networks and Deep Learning

KING'S College LONDON

Tutorial - 8: Support vector machines

Instructors: David Watson, Stefanos Leonardos *Term: Spring 2025 – 10-Mar*

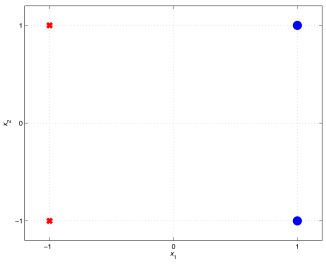
Exercise 1. A Support Vector Machine (SVM) classifier is employed to classify the following points

$$\text{class 1: } \mathbf{x_1} = \begin{pmatrix}1\\1\end{pmatrix}, \mathbf{x_2} = \begin{pmatrix}1\\-1\end{pmatrix}, \quad \text{class 2: } \mathbf{x_3} = \begin{pmatrix}-1\\1\end{pmatrix}, \mathbf{x_4} = \begin{pmatrix}-1\\-1\end{pmatrix}.$$

i) Determine if the two classes are linearly separable.
ii) Design an SVM classifier to correctly classify all given points.
iii) Identify the support vectors.

Solution 1.

i) By plotting the data it can be seen that the two classes are linearly separable. Thus, the best kernel function is the linear kernel function, i.e., we do not need to apply a transformation to make the data linearly separable.



ii) Define the label for Class 1 as $+1$ and Class 2 as -1 so that $y_1 = y_2 = 1$ and $y_3 = y_4 = -1$. The separating hyperplane is: $\mathbf{w}^T\mathbf{x} + w_0 = w_1x_1 + w_2x_2 + w_0 = 0$. To find it, we solve the following linear program:

$$\begin{aligned}\mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda}) &= -\sum_{i=1}^{4} \lambda_i(y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1) + \frac{1}{2}\|\mathbf{w}\|^2 \qquad \textbf{(Primal problem)}\\ &= -\lambda_1(w_1 + w_2 + w_0 - 1) - \lambda_2(w_1 - w_2 + w_0 - 1)\\ &\quad - \lambda_3(w_1 - w_2 - w_0 - 1) - \lambda_4(w_1 + w_2 - w_0 - 1) + \frac{1}{2}(w_1^2 + w_2^2)\end{aligned}$$

To solve the primal problem, we derive the following relations from the optimality conditions and the constraints:

Step 1: Find the partial derivatives of $\mathcal{L}$ with respect to $w_i, i = 1, 2$ and w_0 and set them equal to 0:

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

$$\frac{\partial\mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_2} = 0 \quad \Rightarrow \quad w_2 = \lambda_1 - \lambda_2 - \lambda_3 + \lambda_4 \tag{2}$$

$$\frac{\partial\mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \quad \Rightarrow \quad 0 = \lambda_1 + \lambda_2 - \lambda_3 - \lambda_4. \tag{3}$$

Step 2: Calculate $\lambda_i\big(y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1\big) = 0$ for all $i = 1, 2, 3, 4$ which leads to:

$$\lambda_1(w_1 + w_2 + w_0 - 1) = 0 \tag{4}$$

$$\lambda_2(w_1 - w_2 + w_0 - 1) = 0 \tag{5}$$

$$\lambda_3(w_1 - w_2 - w_0 - 1) = 0 \tag{6}$$

$$\lambda_4(w_1 + w_2 - w_0 - 1) = 0 \tag{7}$$

Step 3: $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$.

Thus, we have derived a system of 7 equations (3 in step 1 and 4 in step 2) in the 7 unknowns: $\lambda_1, \lambda_2, \lambda_3, \lambda_4, w_1, w_2, w_0$ plus the non-negativity constraints for $\lambda_i, i = 1, 2, 3, 4$. Our goal is to solve this system, which, in the general case, can be done by using a solver. In this case, this can be done by observing that, if we add all equations from step 2, i.e., (4) + (5) + (6) + (7), and substitute the relations from step 1, then we get

$$\begin{aligned} & w_1(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4) + w_2(\lambda_1 - \lambda_2 - \lambda_3 + \lambda_4) + w_0(\lambda_1 + \lambda_2 - \lambda_3 - \lambda_4) - (\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4) = 0, \\ \Rightarrow \quad & w_1 \cdot (w_1) + w_2 \cdot (w_2) + w_0 \cdot (0) - w_1 = 0, \\ \Rightarrow \quad & w_1(w_1 - 1) + w_2^2 = 0 \\ \Rightarrow \quad & \boxed{w_2 = \sqrt{w_1(1 - w_1)}}. \end{aligned}$$

To proceed, also consider the condition $y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1 \geq 0$ for $i = 1, 2, 3, 4$ leading to

$$w_1 + w_2 + w_0 - 1 \geq 0 \tag{8}$$
$$w_1 - w_2 + w_0 - 1 \geq 0 \tag{9}$$
$$w_1 - w_2 - w_0 - 1 \geq 0 \tag{10}$$
$$w_1 + w_2 - w_0 - 1 \geq 0 \tag{11}$$

Summing up these inequalities, i.e., (8)+(9)+(10)+(11), we get that: $4w_1 - 4 \geq 0$ which gives $\boxed{w_1 \geq 1}$. However, we see that if $w_1 > 1$, then $w_2 = \sqrt{-w_1(w_1 - 1)}$ is not possible. Thus, the only solution is $w_1 = 1$, and $w_2 = \sqrt{-w_1(w_1 - 1)} = \sqrt{-1 \times 0} = 0$. Thus, $\boxed{\mathbf{w} = (w_1, w_2) = (1, 0)}$. Substituting $w_1 = 1$ and $w_2 = 0$ in (8) and (11), we get that

$$w_0 \geq 0, \; -w_0 \geq 0 \quad \Rightarrow \quad \boxed{w_0 = 0}.$$

Thus, $\mathbf{w}^T\mathbf{x} = (1 \cdot x_1 + 0 \cdot x_2 + 0) = x_1$, which implies that the equation of the separating hyperplane is $x_1 = 0$ and the linear SVM classifier is:

$$\text{sign}(w_1x_1 + w_2x_2 + w_0) = \text{sign}(1 \cdot x_1 + 0 \cdot x_2 + 0) = \text{sign}(x_1) \qquad \text{(hard classifier)}$$

iii) As the hyperplane is $x_1 = 0$, it is a vertical line equidistant from all four samples. This implies that all $\mathbf{x}_1$ to $\mathbf{x}_4$ are the support vectors.
Alternatively, this can be seen by recalling that when $\mathbf{x}$ is a support vector, the value of $\mathbf{w}^T\mathbf{x} = x_1$ is 1 if $\mathbf{x}$ belongs to class '+1' or -1 if $\mathbf{x}$ belongs to the class '-1'.

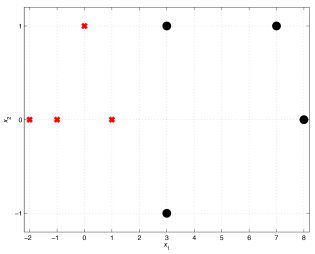
Exercise 2. An SVM classifier is employed to classify the following points

$$\text{class 1: } \mathbf{x_1} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \mathbf{x_2} = \begin{pmatrix} 3 \\ -1 \end{pmatrix}, \mathbf{x_3} = \begin{pmatrix} 7 \\ 1 \end{pmatrix}, \mathbf{x_4} = \begin{pmatrix} 8 \\ 0 \end{pmatrix}$$
$$\text{class 2: } \mathbf{x_5} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \mathbf{x_6} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \mathbf{x_7} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \mathbf{x_8} = \begin{pmatrix} -2 \\ 0 \end{pmatrix}.$$

i) Determine if the dataset is linearly separable.
ii) Identify the support vectors by inspection.
iii) Design an SVM classifier to correctly classify all given points. What is the margin?
iv) Classify the point $\mathbf{x}^T = (2.5, 0.5)$ using the designed SVM classifier.

Solution 2.

i) By plotting the data it can be seen that the dataset is linearly separable.



ii) 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:

$$\mathbf{x}_1 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} 3 \\ -1 \end{pmatrix}, \mathbf{x}_5 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

iii) Define the label for Class 1 as $+1$ and Class 2 as -1 so that $y_1 = y_2 = y_3 = y_4 = 1$ and $y_5 = y_6 = y_7 = y_8 = -1$. The separating hyperplane is $\mathbf{w}^T\mathbf{x} + w_0 = 0$ where $\mathbf{w} = \sum \lambda_i y_i \mathbf{x}_i$ with $\lambda_i = 0$ for all non-support vectors. Thus,

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

To proceed, recall that $y_i(\mathbf{w}^T\mathbf{x} + w_0) = 1$ when $\mathbf{x}$ is a support vector. Thus,

- $\mathbf{x} = \mathbf{x}_1$: $\left[\lambda_1 \begin{pmatrix} 3 \\ 1 \end{pmatrix} + \lambda_2 \begin{pmatrix} 3 \\ -1 \end{pmatrix} - \lambda_5 \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right]^T \begin{pmatrix} 3 \\ 1 \end{pmatrix} + w_0 = 1 \quad \Rightarrow \quad 10\lambda_1 + 8\lambda_2 - 3\lambda_5 + w_0 = 1.$
- $\mathbf{x} = \mathbf{x}_2$: $\left[\lambda_1 \begin{pmatrix} 3 \\ 1 \end{pmatrix} + \lambda_2 \begin{pmatrix} 3 \\ -1 \end{pmatrix} - \lambda_5 \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right]^T \begin{pmatrix} 3 \\ -1 \end{pmatrix} + w_0 = 1 \quad \Rightarrow \quad 8\lambda_1 + 10\lambda_2 - 3\lambda_5 + w_0 = 1.$
- $\mathbf{x} = \mathbf{x}_5$: $\left[\lambda_1 \begin{pmatrix} 3 \\ 1 \end{pmatrix} + \lambda_2 \begin{pmatrix} 3 \\ -1 \end{pmatrix} - \lambda_5 \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right]^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} + w_0 = -1 \quad \Rightarrow \quad 3\lambda_1 + 3\lambda_2 - \lambda_5 + w_0 = -1.$

Combining with the condition $\sum_{i=1}^{8} \lambda_i y_i = 0 \Rightarrow \lambda_1 + \lambda_2 - \lambda_5 = 0$, we have the following system of linear equations:

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

Solving this system, gives $\lambda_1 = \lambda_2 = 0.25, \lambda_5 = 0.5$ and $w_0 = -2$. As a result

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

Thus, the separating hyperplane is $\mathbf{w}^T\mathbf{x} + w_0 = x_1 - 2 = 0$, and the margin is $\frac{2}{\|\mathbf{w}\|} = \frac{2}{\sqrt{1^2+0^2}} = 2$.

Note: The above is only one way to solve for the separating hyperplane, i.e., for the values of $\mathbf{w}, w_0$. As we saw in the tutorial, using the optimality conditions, there are alternative ways to do this. For instance, using the equations $y_i(\mathbf{w}^T\mathbf{x} + w_0) = 1$ for the support vectors $\mathbf{x}_1, \mathbf{x}_2$, and $\mathbf{x}_5$, we have that

- $\mathbf{x} = \mathbf{x}_1$: $1 \cdot \left[(w_1, w_2) \cdot \begin{pmatrix} 3 \\ 1 \end{pmatrix} + w_0\right] = 1 \quad \Rightarrow 3w_1 + w_2 + w_0 = 1 \quad (1)$
- $\mathbf{x} = \mathbf{x}_2$: $1 \cdot \left[(w_1, w_2) \cdot \begin{pmatrix} 3 \\ -1 \end{pmatrix} + w_0\right] = 1 \quad \Rightarrow 3w_1 - w_2 + w_0 = 1 \quad (2)$
- $\mathbf{x} = \mathbf{x}_5$: $(-1) \cdot \left[(w_1, w_2) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} + w_0\right] = 1 \quad \Rightarrow w_1 + w_0 = -1 \quad (3)$

By taking (1) − (2), we find that

$$3w_1 + w_2 + w_0 - (3w_1 - w_2 + w_0) = 1 - 1 \Rightarrow \boxed{w_2 = 0}.$$

Thus, by taking (1) − (3) and using that $w_2 = 0$, we find that

$$3w_1 + 0 + w_0 - (w_1 + w_0) = 1 - (-1) \Rightarrow \boxed{w_1 = 1}.$$

Substituting $w_1 = 1$ back in (3) yields that $\boxed{w_0 = -2}$ as above.

iv) The hard SVM classifier is $\text{sign}(x_1 - 2)$. Thus, for the point $\mathbf{x} = \begin{pmatrix} 2.5 \\ 0.5 \end{pmatrix}$, we have that $\text{sign}(x_1 - 2) = \text{sign}(2.5 - 2) = 1$ which means that it is classified as Class 1.

Exercise 3. An SVM classifier is employed to classify the following points

$$\text{class 1: } \mathbf{x_1} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \mathbf{x_2} = \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \mathbf{x_3} = \begin{pmatrix} -2 \\ -2 \end{pmatrix}, \mathbf{x_4} = \begin{pmatrix} -2 \\ 2 \end{pmatrix}$$

$$\text{class 2: } \mathbf{x_5} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{x_6} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \mathbf{x_7} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \mathbf{x_8} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

i) Determine if the data set is linearly separable.

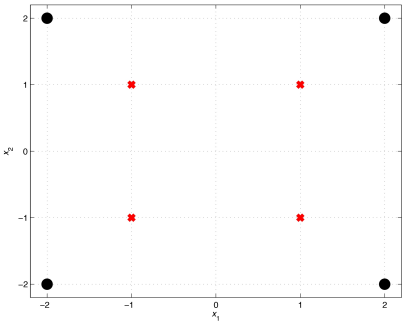
ii) Apply the feature mapping function $\Phi(\mathbf{x}) = \begin{pmatrix} 4 - \frac{x_2}{2} + |x_1 - x_2| \\ 4 - \frac{x_1}{2} + |x_1 - x_2| \end{pmatrix}$, if $\|\mathbf{x}\| > 2$, and $\Phi(\mathbf{x}) = \begin{pmatrix} x_1 - 2 \\ x_2 - 3 \end{pmatrix}$, otherwise, to the data set where $\mathbf{x}^T = (x_1, x_2)$. Determine if the data set is linearly separable in the new feature space.

iii) Identify the support vectors by inspection.

iv) Design a nonlinear SVM classifier to correctly classify all given points.

Solution 3.

i) By plotting the data it can be seen that the dataset is not linearly separable.

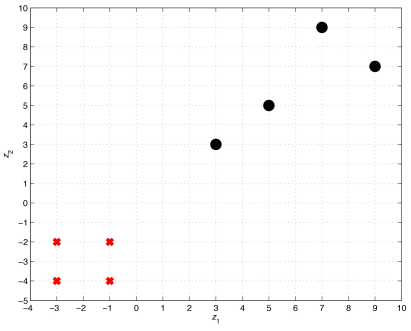


ii) Applying the feature mapping function to $\mathbf{x}$, we obtain the dataset in a new feature space, i.e., $\mathbf{z} = \Phi(\mathbf{x})$,

class 1: $\mathbf{z}_1 = \begin{pmatrix}3\\3\end{pmatrix}, \mathbf{z}_2 = \begin{pmatrix}9\\7\end{pmatrix}, \mathbf{z}_3 = \begin{pmatrix}5\\5\end{pmatrix}, \mathbf{z}_4 = \begin{pmatrix}9\\2\end{pmatrix}$

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

It can be seen from the next Figure that, after applying the feature mapping function, the dataset is linearly separable in the new feature space.



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

iv) Define the label for Class 1 as $+1$ and Class 2 as -1, so that $y_1 = y_2 = y_3 = y_4 = 1$ and $y_5 = y_6 = y_7 = y_8 = -1$. The separating hyperplane is $\mathbf{w}^T\mathbf{z} + w_0 = \mathbf{w}^T\Phi(\mathbf{x}) + w_0 = 0$ where $\mathbf{w} = \sum \lambda_i y_i \mathbf{x}_i$ with $\lambda_i = 0$ for all non-support vectors. Thus,

$$\mathbf{w} = \lambda_1 y_1 \mathbf{z}_1 + \lambda_5 y_5 \mathbf{z}_5 = \lambda_1 y_1 \Phi(\mathbf{x}_1) + \lambda_5 y_5 \Phi(\mathbf{x}_5) = \lambda_1 \begin{pmatrix}3\\3\end{pmatrix} - \lambda_5 \begin{pmatrix}-1\\-2\end{pmatrix}.$$

To proceed, recall that $y_i(\mathbf{w}^T\mathbf{z} + w_0) = 1$ when $\mathbf{z}$ is a support vector. Thus,

$$\cdot\ \mathbf{z} = \mathbf{z}_1: \quad \left[\lambda_1 \begin{pmatrix}3\\3\end{pmatrix} - \lambda_5 \begin{pmatrix}-1\\-2\end{pmatrix}\right]^T \begin{pmatrix}3\\3\end{pmatrix} + w_0 = 1 \quad \Rightarrow \quad 18\lambda_1 + 9\lambda_5 + w_0 = 1.$$

$$\cdot\ \mathbf{z} = \mathbf{z}_5: \quad \left[\lambda_1 \begin{pmatrix}3\\3\end{pmatrix} - \lambda_5 \begin{pmatrix}-1\\-2\end{pmatrix}\right]^T \begin{pmatrix}-1\\-2\end{pmatrix} + w_0 = -1 \quad \Rightarrow \quad -9\lambda_1 - 5\lambda_5 + w_0 = -1.$$

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

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

Solving this system, gives $\lambda_1 = \lambda_5 = 0.0488$ and $w_0 = -0.3171$. As a result

$$\mathbf{w} = 0.0488 \begin{pmatrix}3\\3\end{pmatrix} - 0.0488 \begin{pmatrix}-1\\-2\end{pmatrix} = \begin{pmatrix}0.1952\\0.2440\end{pmatrix}.$$

Thus, the separating hyperplane is:

$$\begin{aligned}
\text{in the new feature space:} \quad & 0 = \mathbf{w}^T\mathbf{z} + w_0 = 0.1952 z_1 + 0.2440 z_2 - 0.3171\\
\text{in the original feature space:} \quad & 0 = \mathbf{w}^T\Phi(\mathbf{x}) + w_0 = (0.1952, 0.2440)\Phi(\mathbf{x}) - 0.3171\\
\text{in the original feature space in kernel form:} \quad & 0 = 0.0488 K(\mathbf{x_1}, x) - 0.0488 K(\mathbf{x_5}, x) - 0.3171,
\end{aligned}$$

where $K(\mathbf{p}, \mathbf{x}) = \Phi(\mathbf{p})^T\Phi(\mathbf{x})$.

Exercise 4. Compare and discuss the advantages and disadvantages of the four approaches to using SVMs for multi-class problems, namely one against one, one against all, binary decision tree and binary coded approach.

Solution 4.

	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

	Advantages	Disadvantages
One against one	• Training is faster for each SVM since the training dataset is smaller compared with the whole dataset. • Can flexibly add or remove classes. For adding classes, existing SVMs are unaffected, only need trained SVMs for new classes. For removing classes, the SVMs for the corresponding classes can be simply removed.	• The number of SVMs is larger compared with other approaches. • Many SVMs have to be evaluated to make the final decision. • The class of some regions of the feature space cannot be determined.
One against all	• Number of SVMs to be trained is relatively few. • Fewer SVMs have to be evaluated to make the final decision.	• Training time may be longer as the whole dataset is used for training each SVM. • When adding/removing classes, all SVMs have to be re-trained. • The class of some regions of the feature space cannot be determined.
Binary decision tree	• Number of SVMs to be trained is relatively few. • When adding classes, only some SVMs need to be trained. The other existing SVMs can be re-used. • Number of SVMs that need to be evaluated to make a decision is further reduced. • Training time is slightly faster since not all SVMs use the whole training dataset. Smaller training sets are used as the algorithm descends the decision tree.	• When removing classes, all SVMs need to be re-trained. • The classification performance heavily depends on the SVMs in the upper level. Classification error will propagate.
Binary code	• Number of SVMs to be trained is further reduced. • Number of SVMs to be evaluated making a decision is further reduced.	• 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 classes if the number of classes is not exactly $2^{\text{Number of SVMs}}$.

Exercise 5. Given the class assignments in Table 1 for a multi-class SVM classifier using the *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

Solution 5.

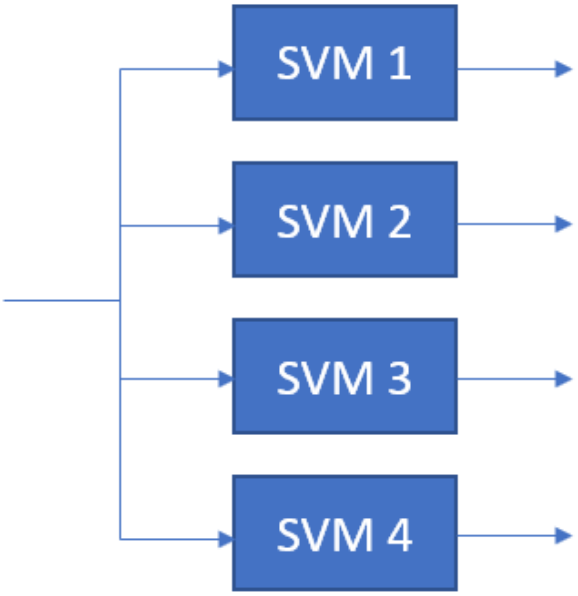
Code assignments:

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 there are combinations of SVM outputs that do not correspond to one of the classes as shown in the table below:

Class	SVM 1	SVM 2	SVM 3	SVM 4
?	+1	+1	−1	+1
?	+1	−1	+1	+1
?	+1	−1	−1	−1
?	−1	+1	+1	+1
?	−1	+1	−1	−1
?	−1	−1	+1	−1
?	−1	−1	−1	+1

The *block diagram* for the multi-class SVM designed above is shown below.



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:

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

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

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

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

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:

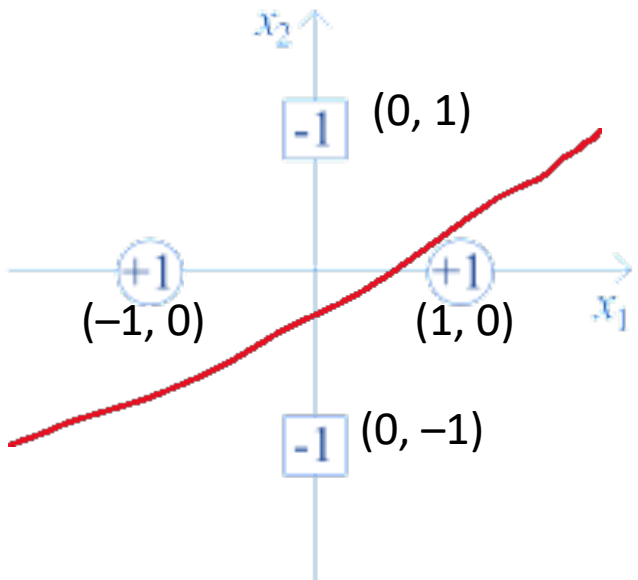
$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

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

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

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

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:

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

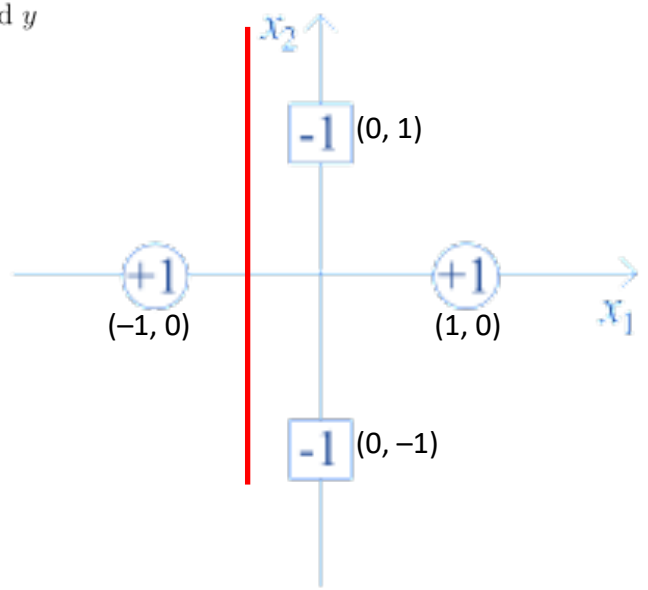
$$h_3(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > 0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_4(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > 0.5 \\ +1, & \text{otherwise} \end{cases}$$

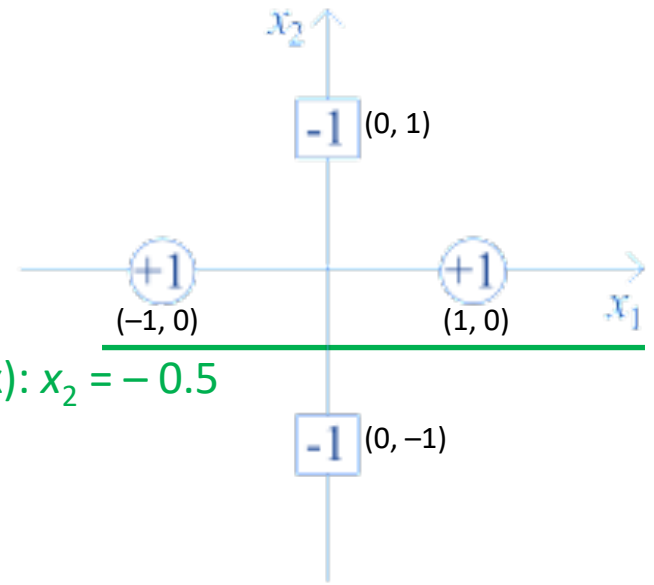
$$h_5(\mathbf{x}) = \begin{cases} +1, & \text{if } x_2 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_6(\mathbf{x}) = \begin{cases} -1, & \text{if } x_2 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

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

b. Determine the classification error for all weak classifiers.

$h_1(\mathbf{x})$: $x_1 = -0.5$





$h_5(\mathbf{x})$: $x_2 = -0.5$

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

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

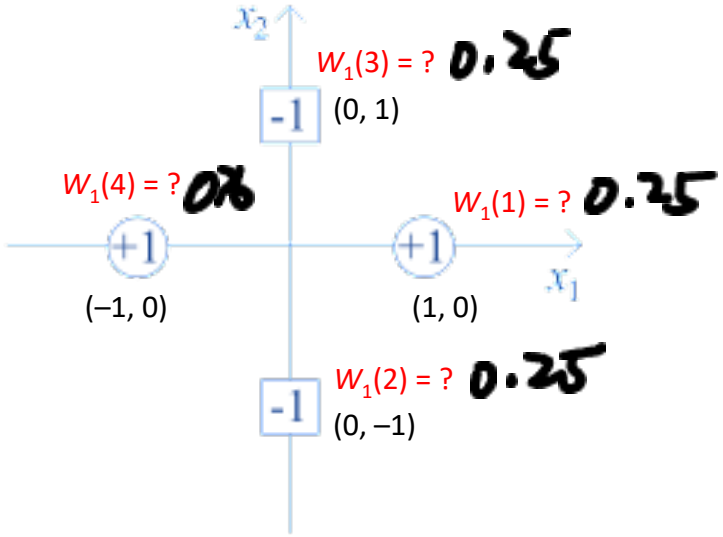
Q1. Consider the following dataset: $\{(\mathbf{x} = (1,0), y = +1), (\mathbf{x} = (-1,0), y = +1), (\mathbf{x} = (0,1), y = -1), (\mathbf{x} = (0,-1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

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

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

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

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



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

$k_{max} = ? \rightarrow$

Algorithm: Adaboost

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

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. $\hat{h}_k = \arg\min_{h_j \in H} E_j$ 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 $h_k \in H$ 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:

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

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

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

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

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.

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

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

Round 1:

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

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

3. **Determine ε_1:** $\varepsilon_1 =$

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

Algorithm: Adaboost

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

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

  $k \leftarrow 0$

  do $k \leftarrow k + 1$

    $\hat{h}_k = \arg\min_{h_j \in H} E_j$ where $E_j = \sum_{i=1, y_i \neq h_j(\mathbf{x}_i)} W_k(i)$ (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 $h_k \in H$ 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:

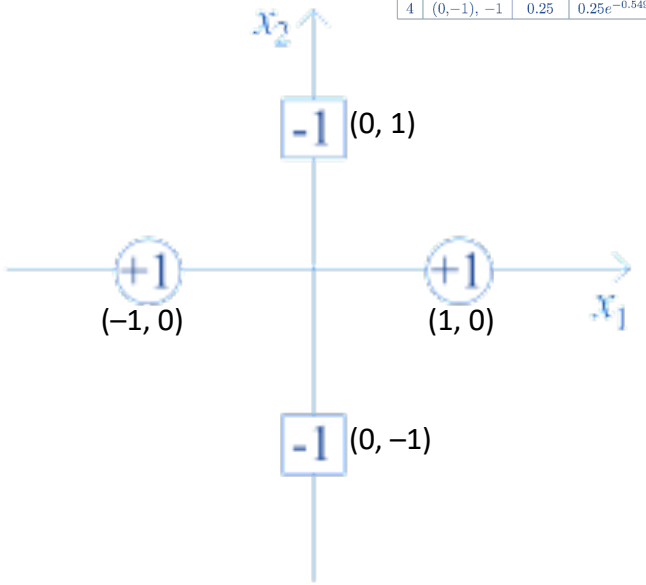
$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

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

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

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

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



We are at round 1: $k = 1$

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

4

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			$0.4330/0.8659 = 0.5001$
2	(−1,0), +1			$0.1443/0.8659 = 0.1666$
3	(0,1), −1			$0.1443/0.8659 = 0.1666$
4	(0,−1), −1		$0.25e^{-0.5493} = 0.1443$	$0.1443/0.8659 = 0.1666$

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

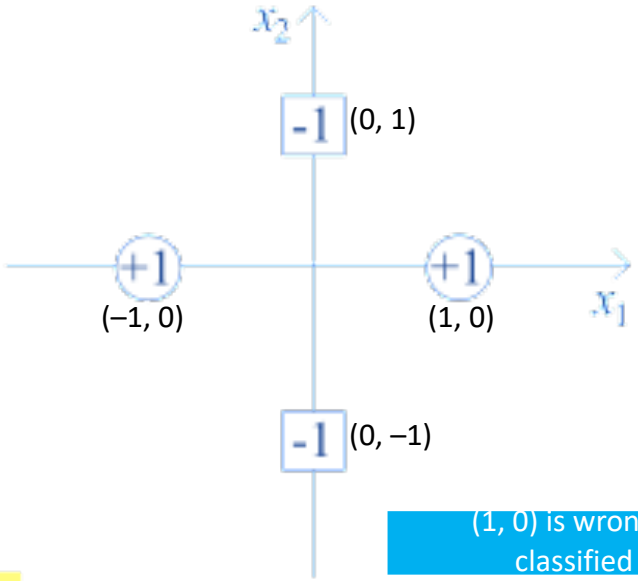
Q1. Consider the following dataset: $\{(\mathbf{x} = (1,0), y = +1), (\mathbf{x} = (-1,0), y = +1), (\mathbf{x} = (0,1), y = -1), (\mathbf{x} = (0,-1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

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

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

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

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



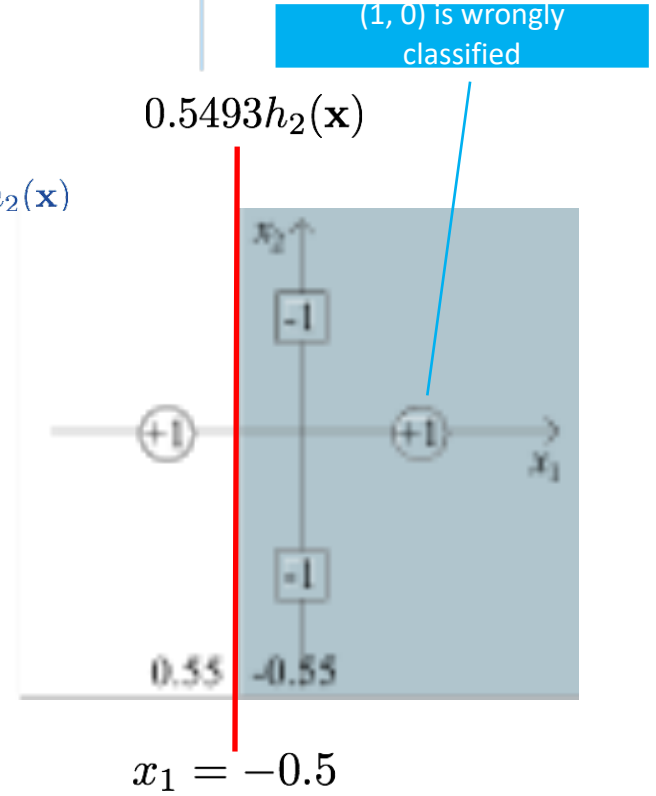
c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.

- **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: $\text{sgn}(0.5493 h_2(\mathbf{x}))$.

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.



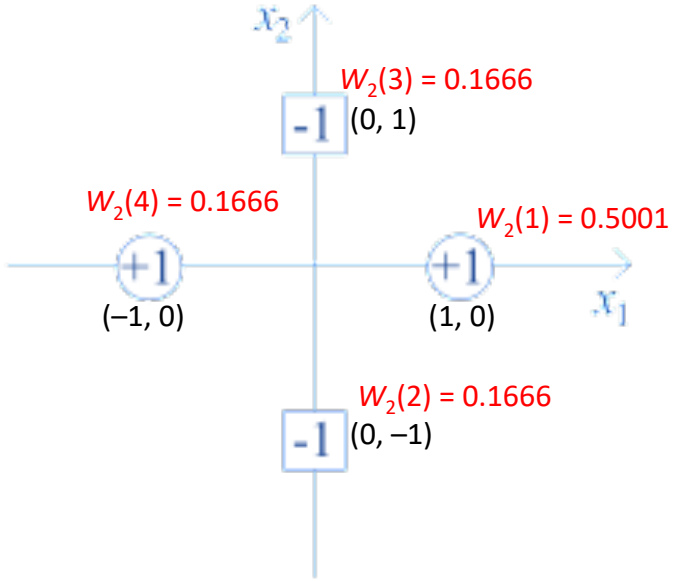
Q1. Consider the following dataset: $\{(\mathbf{x} = (1,0), y = +1), (\mathbf{x} = (-1,0), y = +1), (\mathbf{x} = (0,1), y = -1), (\mathbf{x} = (0,-1), y = -1)\}$ where $\mathbf{x} = (x_1, x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

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

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

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

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



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

Algorithm: Adaboost

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

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

  $k \leftarrow 0$

  do $k \leftarrow k + 1$ ← $k = 2$

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

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

    if $e_k > 0.5$

      $k_{max} = k - 1$

      Stop

    end

    $a_k = \frac{1}{2} \ln\left(\frac{1 - e_k}{e_k}\right)$

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

  until $k = k_{max}$

end

† Find the classifier $h_k \in H$ that minimises the error e_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:

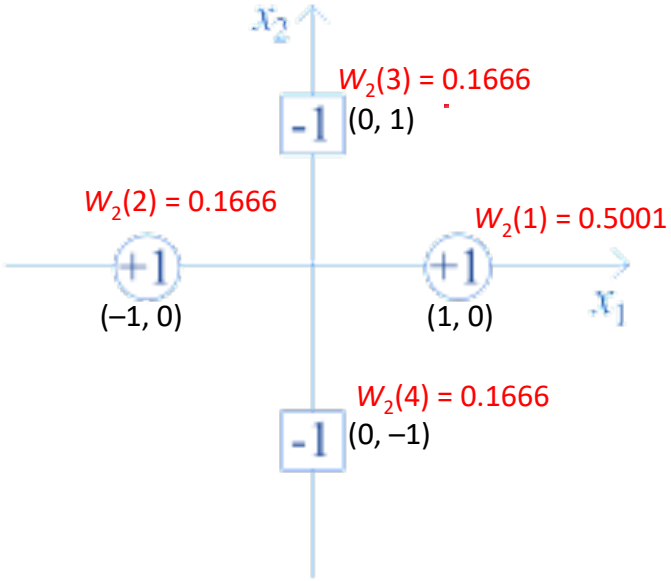
$$h_1(\mathbf{x}) = \begin{cases} +1, & \text{if } x_1 > -0.5 \\ -1, & \text{otherwise} \end{cases} \quad ; \quad h_2(\mathbf{x}) = \begin{cases} -1, & \text{if } x_1 > -0.5 \\ +1, & \text{otherwise} \end{cases}$$

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

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

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

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



Round 2: $k = 2$

- **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	= ?
$h_2(\mathbf{x})$	−1	+1	−1	−1	
$h_3(\mathbf{x})$	+1	−1	−1	−1	
$h_4(\mathbf{x})$	−1	+1	+1	+1	
$h_5(\mathbf{x})$	+1	+1	+1	−1	
$h_6(\mathbf{x})$	−1	−1	−1	+1	
$h_7(\mathbf{x})$	−1	−1	+1	−1	
$h_8(\mathbf{x})$	+1	+1	−1	+1	

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:

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

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

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

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

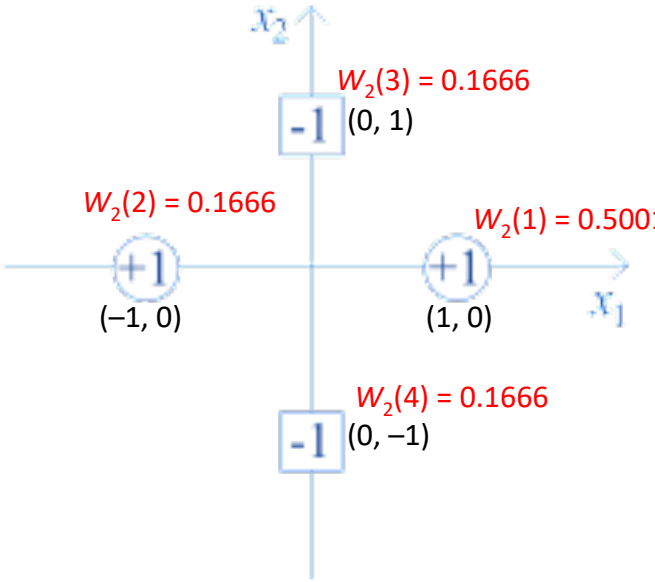
c. Find the minimal classifier using AdaBoost algorithm, which can reach zero training error.

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

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

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



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

Algorithm: Adaboost

Given $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i = \{-1, 1\}, i = 1, \ldots, n$
begin initialise k_{max}, $W_1(i) = 1/n, i = 1, \ldots, n$
  $k \leftarrow 0$
  do $k \leftarrow k + 1$
    $\hat{h}_k = \arg\min_{h_j \in H} E_j$ where $E_j = \sum_{i=1, y_i \neq h_j(\mathbf{x}_i)}^{n} W_k(i)$ (Weighted error rate)†
    $e_k =$ overall weighted error rate of classifier $\hat{h}_k$ (i.e., the minimum E_j)
    if $e_k > 0.5$
      $k_{max} = k - 1$
      Stop
    end
    $a_k = \frac{1}{2}\ln\left(\frac{1-e_k}{e_k}\right)$
    $W_{k+1}(i) = \frac{W_k(i)\, e^{-a_k y_i \hat{h}_k(\mathbf{x}_i)}}{Z_k}$, $i = 1, \ldots, n$ (Update $W_k(i)$)‡
  until $k = k_{max}$
end

† Find the classifier $h_k \in H$ that minimises the error e_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.

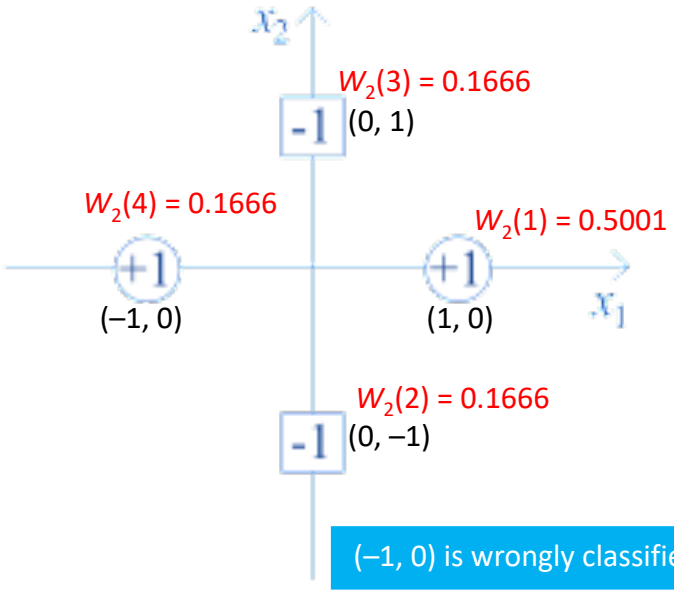
Q1. Consider the following dataset: $\{(\mathbf{x}=(1,0), y=+1), (\mathbf{x}=(-1,0), y=+1), (\mathbf{x}=(0,1), y=-1), (\mathbf{x}=(0,-1), y=-1)\}$ where $\mathbf{x}=(x_1,x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

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

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

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

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



(–1, 0) is wrongly classified

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

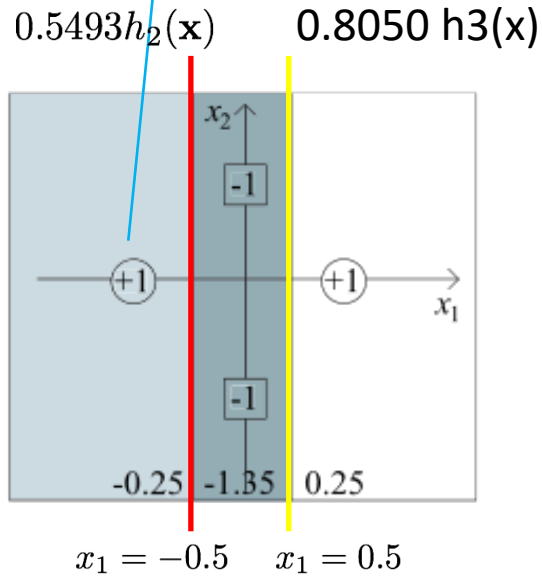
4

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

- **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	0.5001	$0.5001e^{-0.8050} = 0.2236$	$0.2236/0.7452 = 0.3000$
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	$0.1666e^{-0.8050} = 0.0745$	$0.0745/0.7452 = 0.1000$

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

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

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

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

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

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

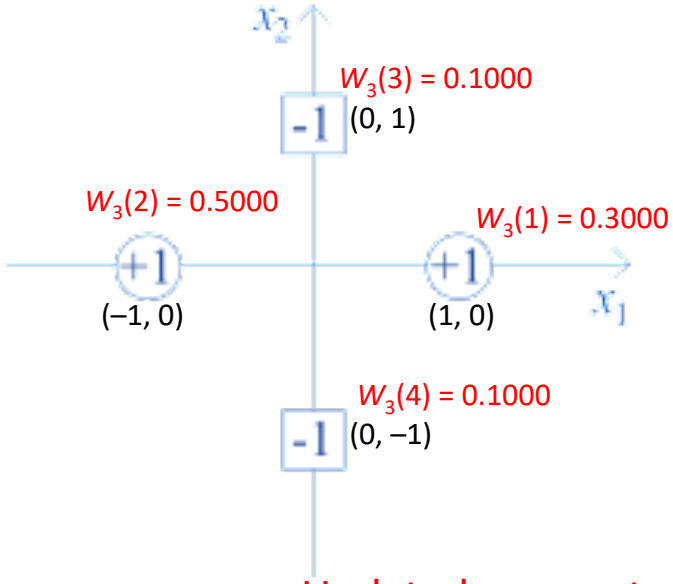
Round 3: $k = 3$

- **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	$0.5000 + 0.1000 \times 2 = 0.7000$
$h_2(\mathbf{x})$	−1	+1	−1	−1	0.3000
$h_3(\mathbf{x})$	+1	−1	−1	−1	0.5000
$h_4(\mathbf{x})$	−1	+1	+1	+1	$0.3000 + 0.1000 \times 2 = 0.5000$
$h_5(\mathbf{x})$	+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, $\hat{h}_3(\mathbf{x}) = h_5(\mathbf{x})$.



Updated error rate

Algorithm: Adaboost

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

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

  $k \leftarrow 0$

  do $k \leftarrow k + 1$

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

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

    if $\theta_k > 0.5$

      $k_{max} = k - 1$

      Stop

    end

    $a_k = \frac{1}{2} \ln\left(\frac{1-\theta_k}{\theta_k}\right)$

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

  until $k = k_{max}$

end

† Find the classifier $h_k \in H$ that minimises the error θ_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.

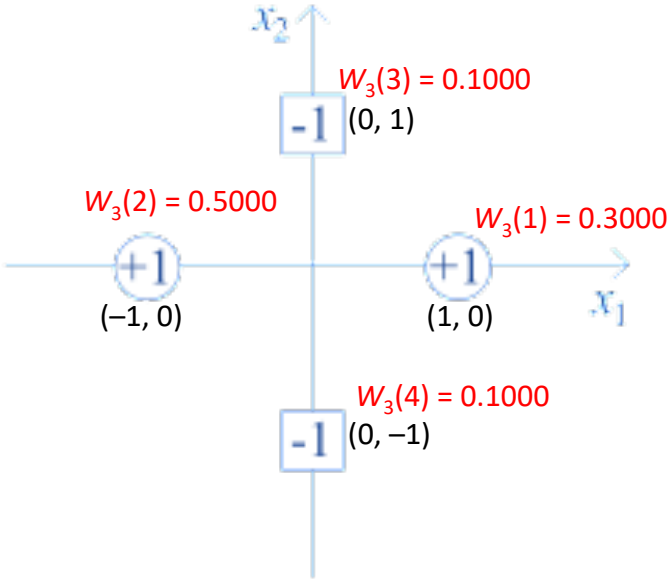
Q1. Consider the following dataset: $\{(\mathbf{x}=(1,0), y=+1), (\mathbf{x}=(-1,0), y=+1), (\mathbf{x}=(0,1), y=-1), (\mathbf{x}=(0,-1), y=-1)\}$ where $\mathbf{x}=(x_1,x_2)$ is the input feature and y is the output class. 8 weak classifies are designed and given as follows:

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

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

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

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



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

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

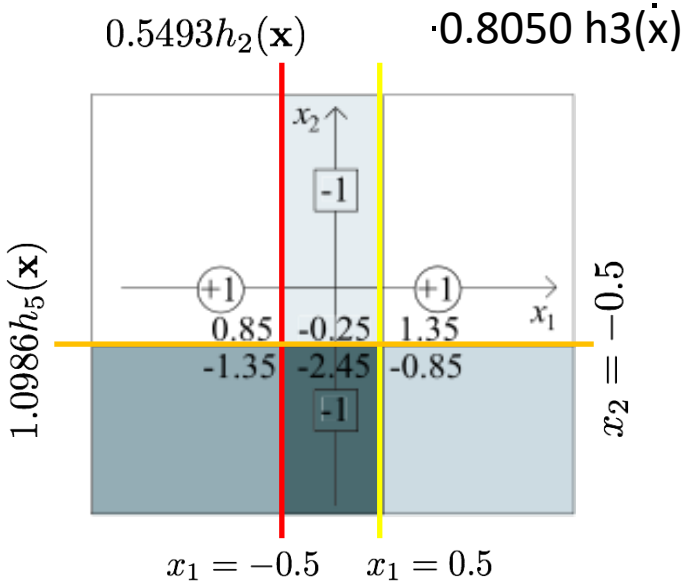
3 • **Determine α_3:** $\alpha_3 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_3}{\varepsilon_3}\right) = \frac{1}{2}\ln\left(\frac{1-0.1000}{0.1000}\right) = 1.0986$.

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

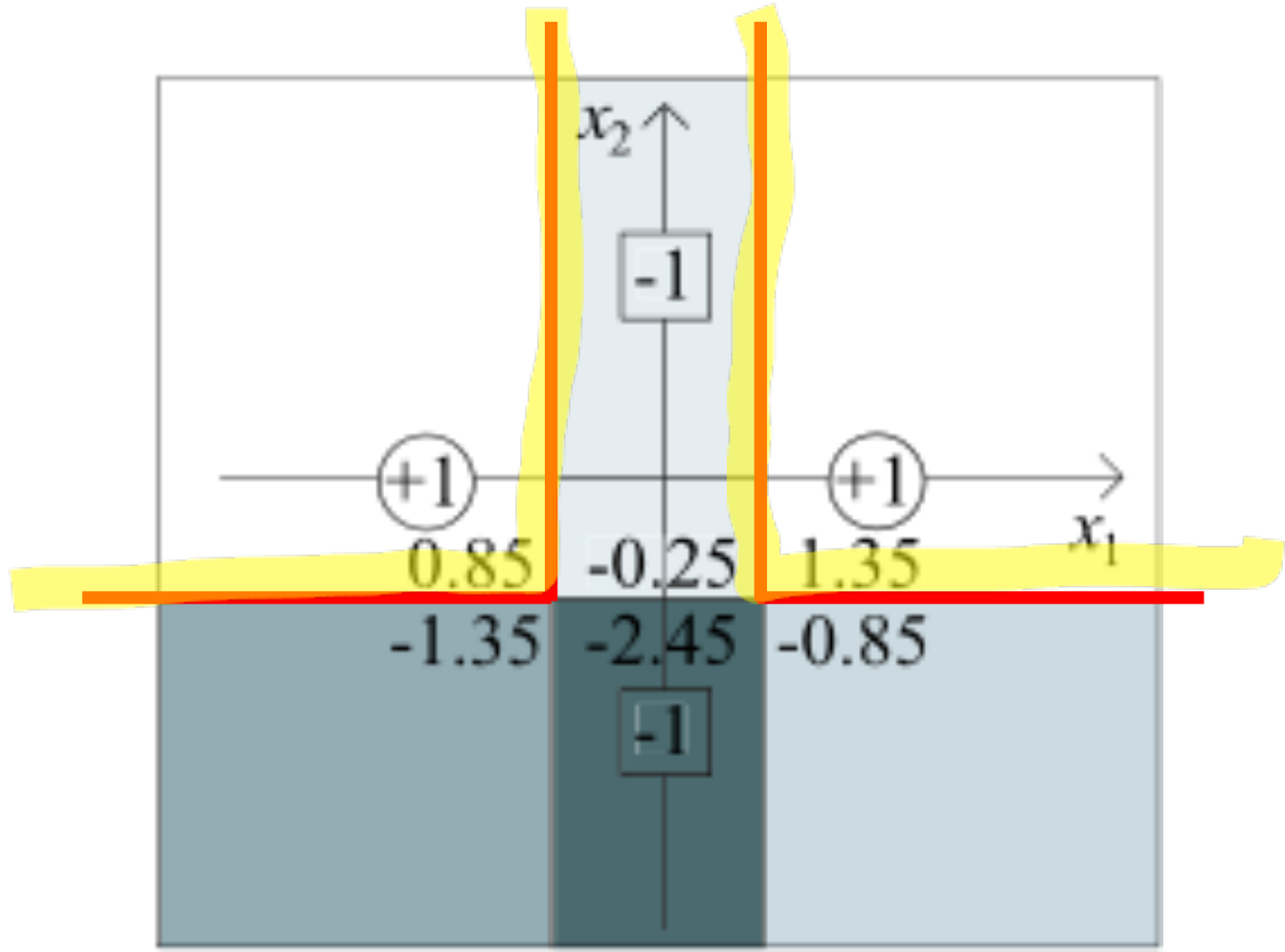
i	Class	$W_k(i)$	$W_k(i)e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}$	$W_{k+1}(i) = W_k(i)e^{-\alpha_k y_i \hat{h}_k(\mathbf{x}_i)}/Z_k$
1	(1,0), +1	0.3000	$0.3000e^{-1.0986} = 0.1000$	$0.1000/0.6333 = 0.1579$
2	(−1,0), +1	0.5000	$0.5000e^{-1.0986} = 0.1667$	$0.1667/0.6333 = 0.2632$
3	(0,1), −1	0.1000	$0.1000e^{1.0986}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

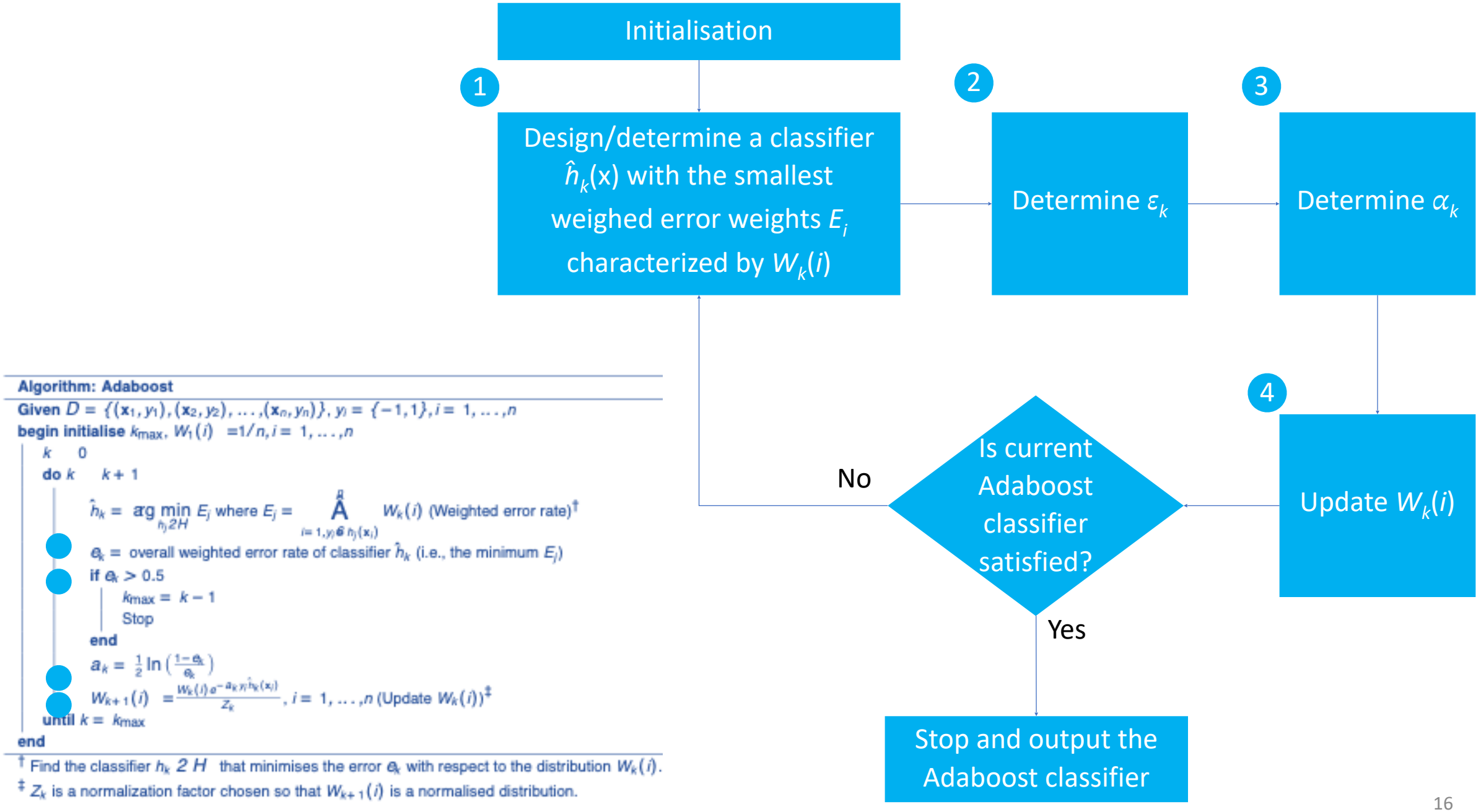
The final hard classifier is: $\text{sgn}(0.5493h_2(\mathbf{x}) + 0.8050h_3(\mathbf{x}) + 1.0986h_5(\mathbf{x}))$



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



How good is this Adaboost classifier?



Algorithm: Adaboost

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

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

$k \quad 0$

do $k \quad k + 1$

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

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

if $e_k > 0.5$

$k_{max} = k - 1$

Stop

end

$a_k = \frac{1}{2} \ln\left(\frac{1 - e_k}{e_k}\right)$

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

until $k = k_{max}$

end

† Find the classifier $h_k \in H$ that minimises the error e_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.

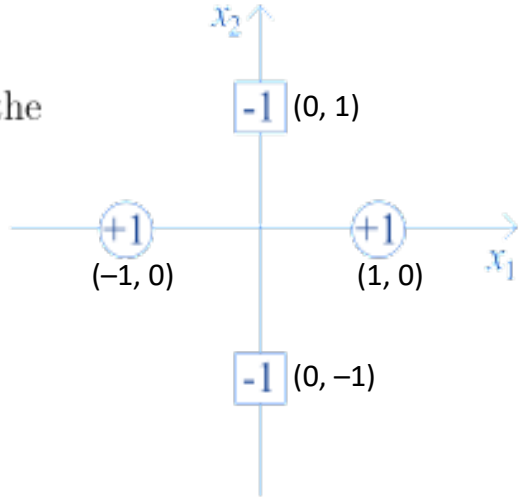
Q2. Using a Bagging algorithm, 8 classifiers shown in Q1 are obtained.

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.



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

Is the bagging classifier working well? Why?

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

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.

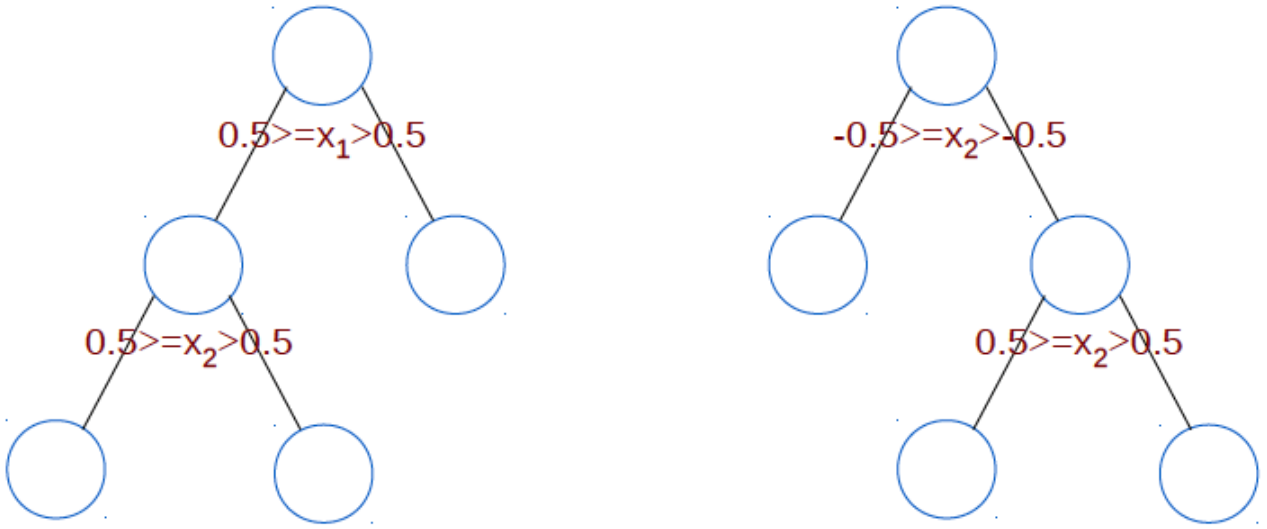
Classifier	(1,0), +1	(−1,0), +1	(0,1), −1	(0,−1), −1	Training Error
$h_1(\mathbf{x})$	+1	−1	+1	+1	0.75
$h_2(\mathbf{x})$	−1	+1	−1	−1	0.25
$h_3(\mathbf{x})$	+1	−1	−1	−1	0.25
$h_4(\mathbf{x})$	−1	+1	+1	+1	0.75
$h_5(\mathbf{x})$	+1	+1	+1	−1	0.25
$h_6(\mathbf{x})$	−1	−1	−1	+1	0.75
$h_7(\mathbf{x})$	−1	−1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

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%)

Q3. Figure 1 shows two binary decision trees.



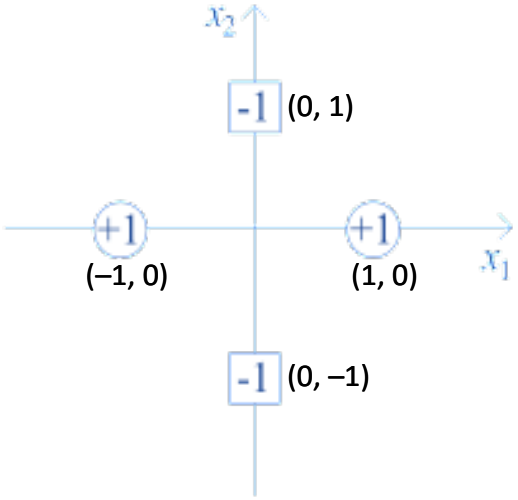


Figure 1: Two decision trees.

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

x_1, x_2

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

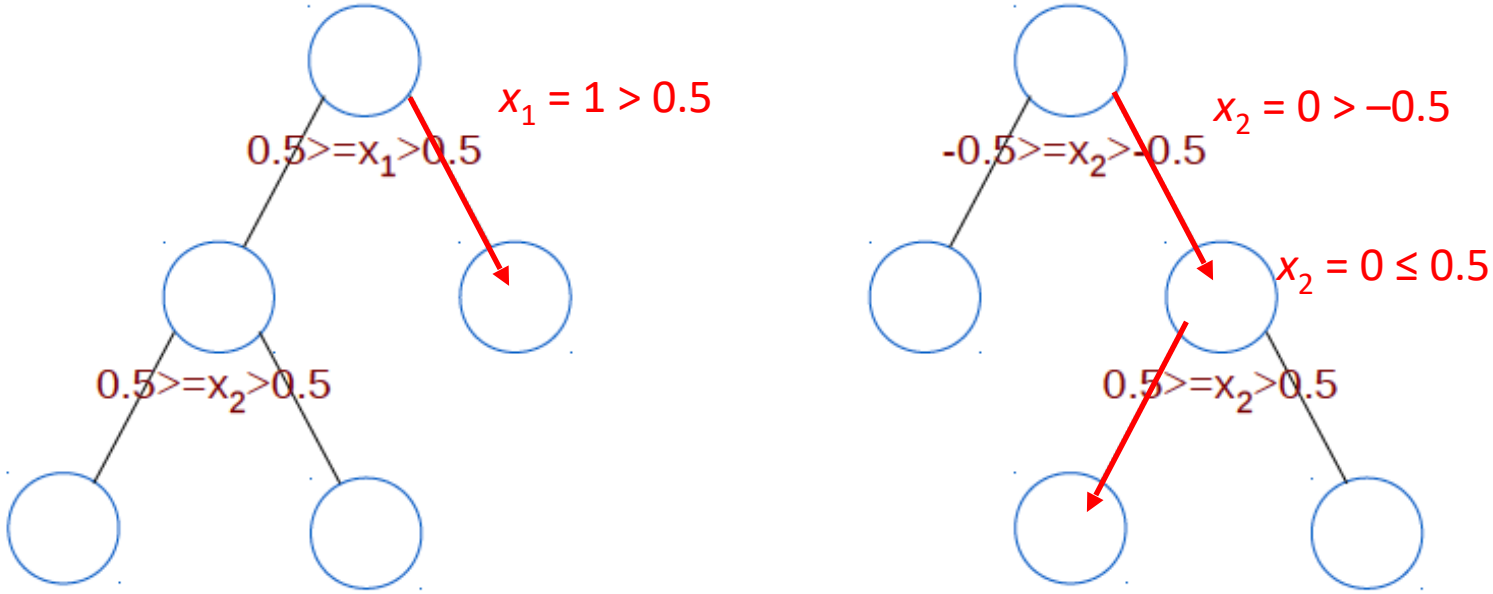


Figure 1: Two decision trees.

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

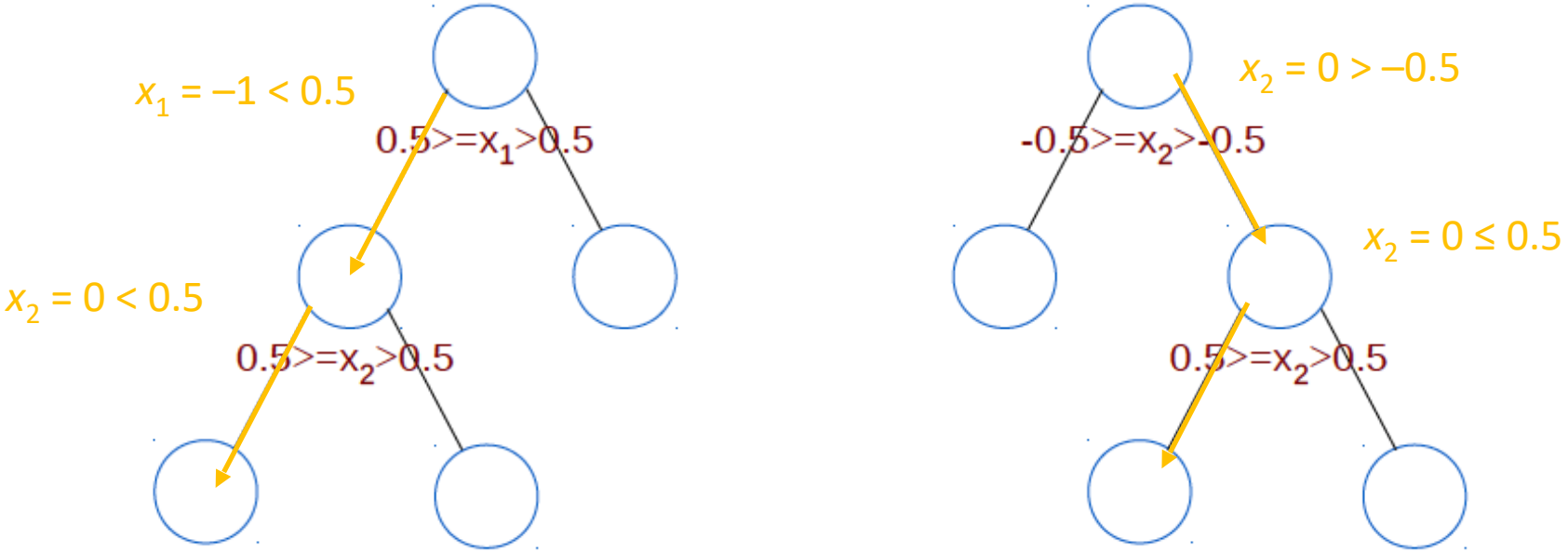


Figure 1: Two decision trees.

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

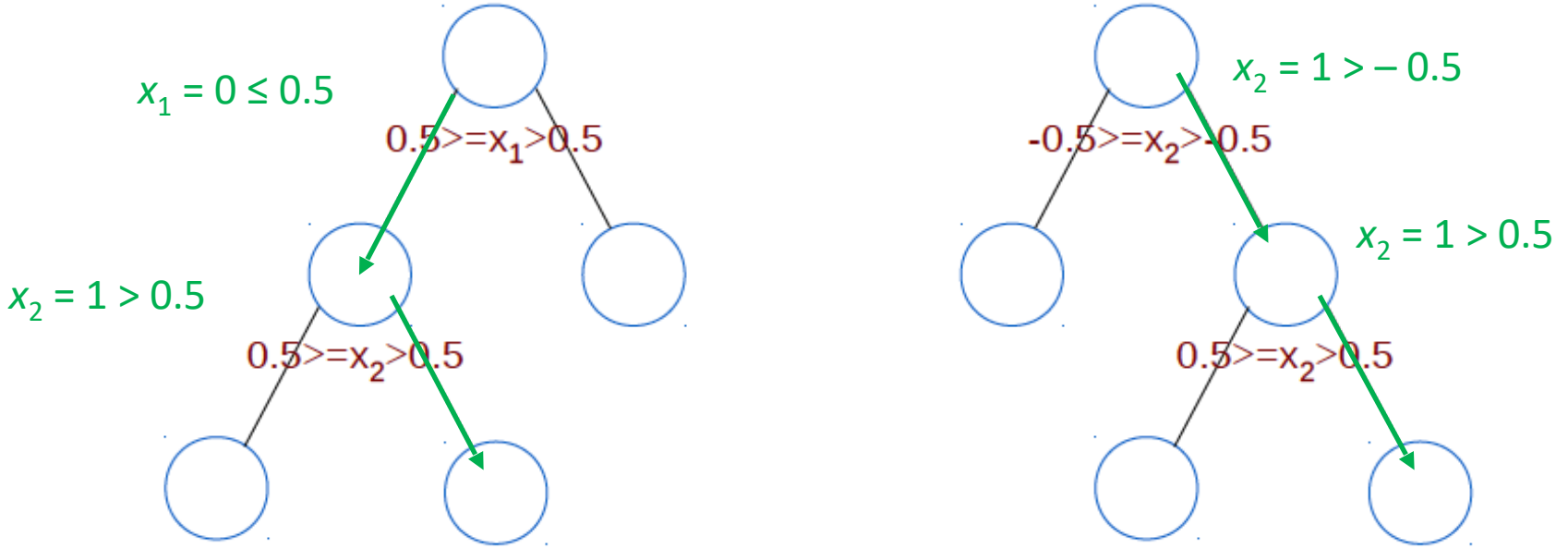


Figure 1: Two decision trees.

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

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

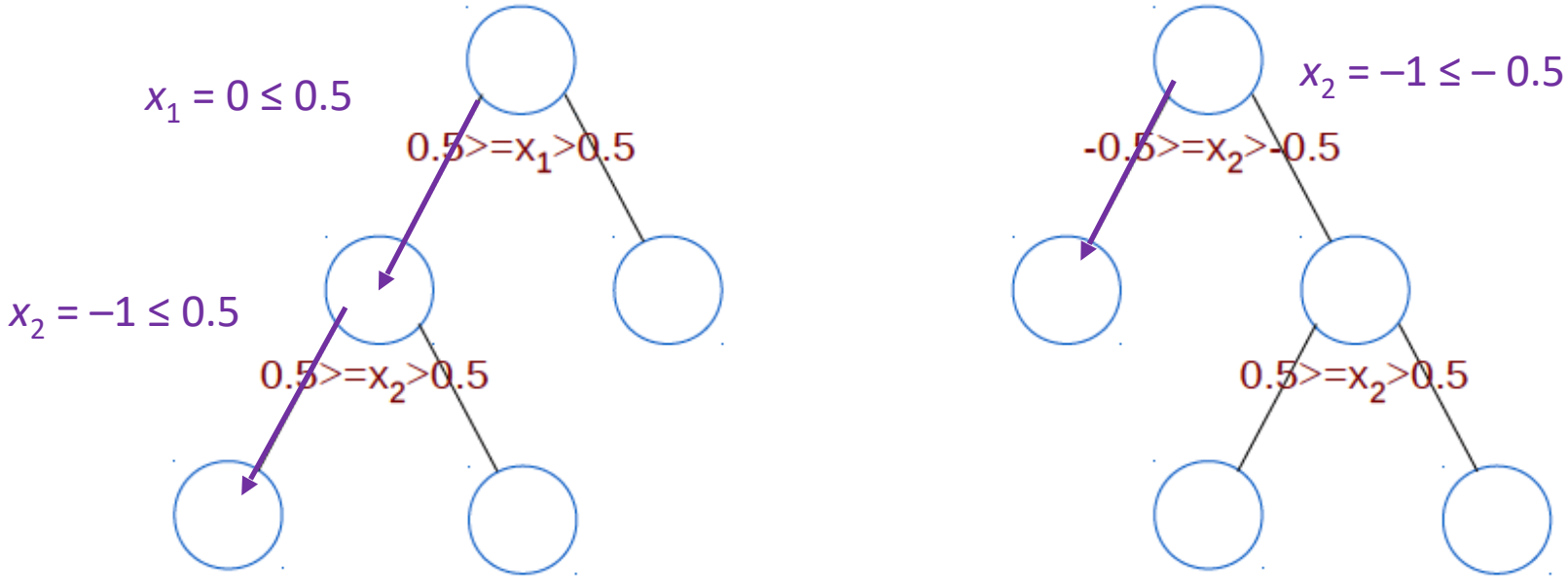


Figure 1: Two decision trees.

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

−1/+1

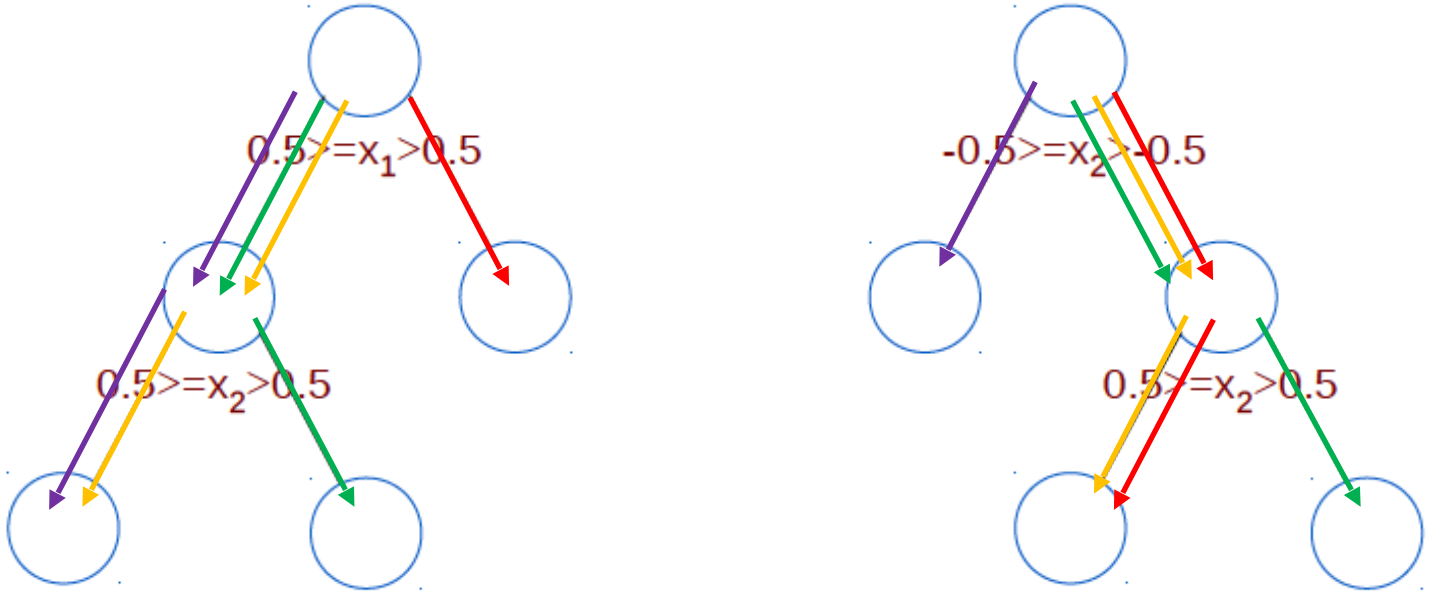


Figure 1: Two decision trees.

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.

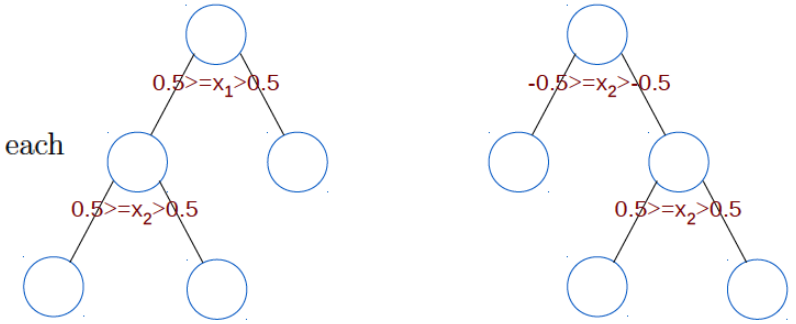


Figure 1: Two decision trees.

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

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

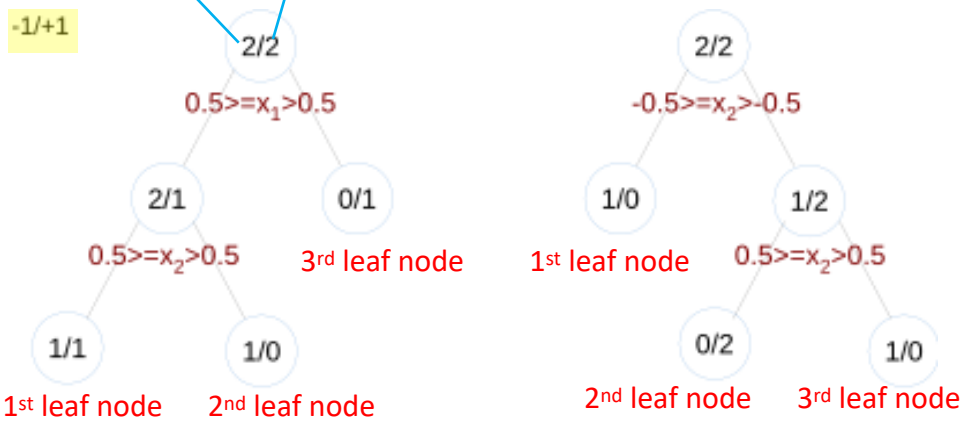
Pass each training sample down each tree, and count how many from each class arrive at each leaf node.

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.

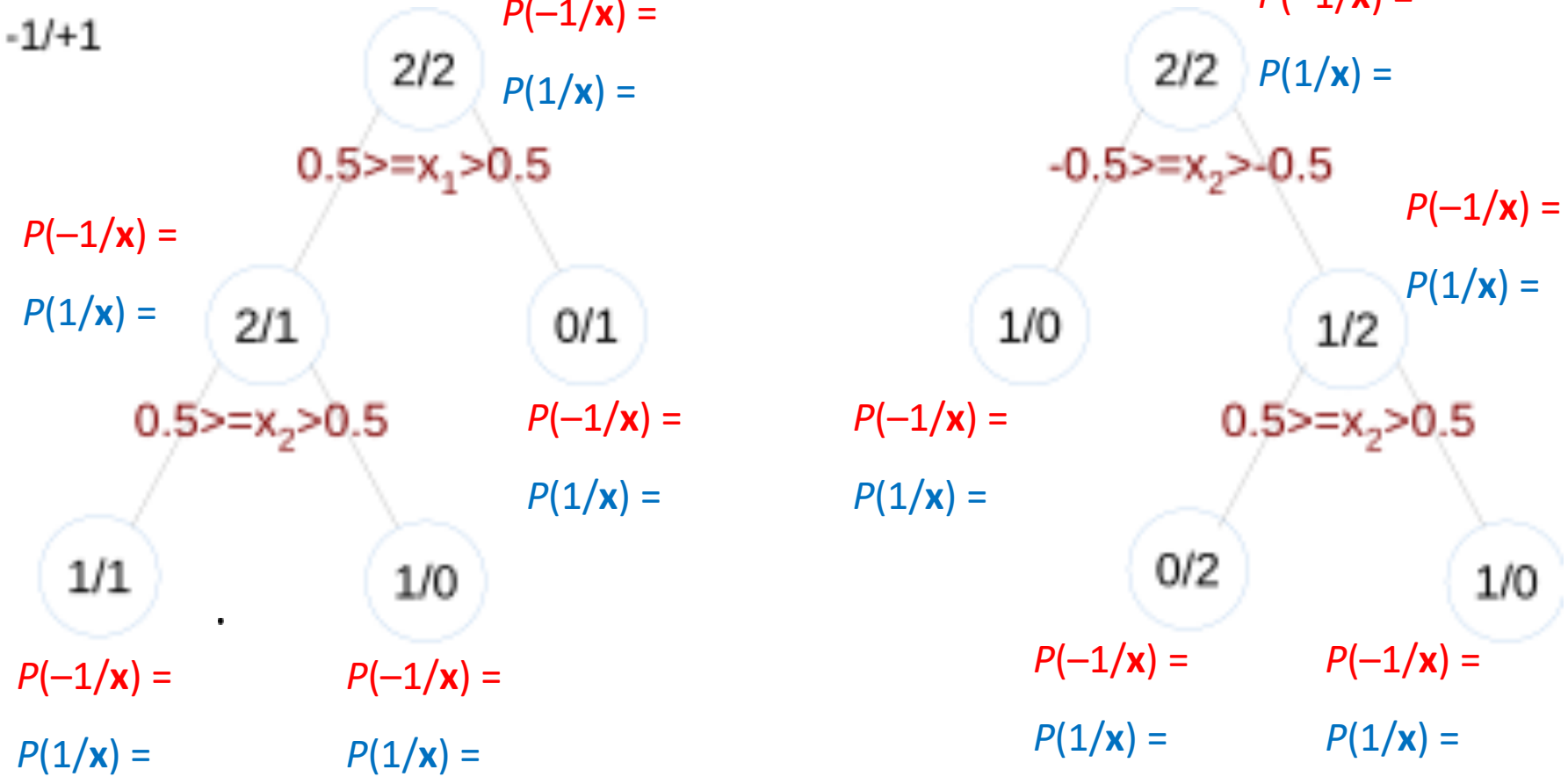
Sample $(1, 0)$ arrives at the second leaf node in tree 2.
Sample $(-1, 0)$ arrives at the second leaf node in tree 2.
Sample $(0, 1)$ arrives at the third leaf node in tree 2.
Sample $(0, -1)$ arrives at the first leaf node in tree 2.

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 $P(class = -1|\mathrm{x})$ for each leaf node.



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.

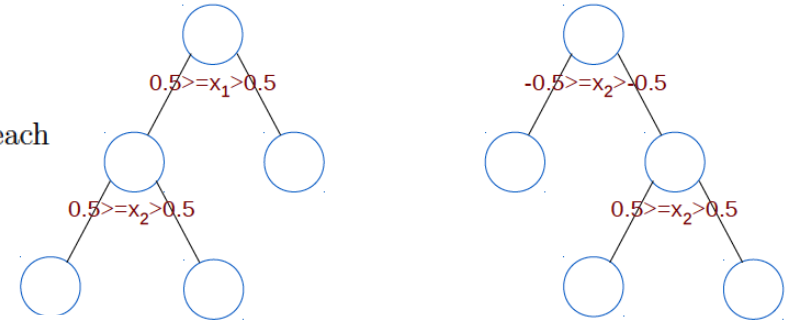
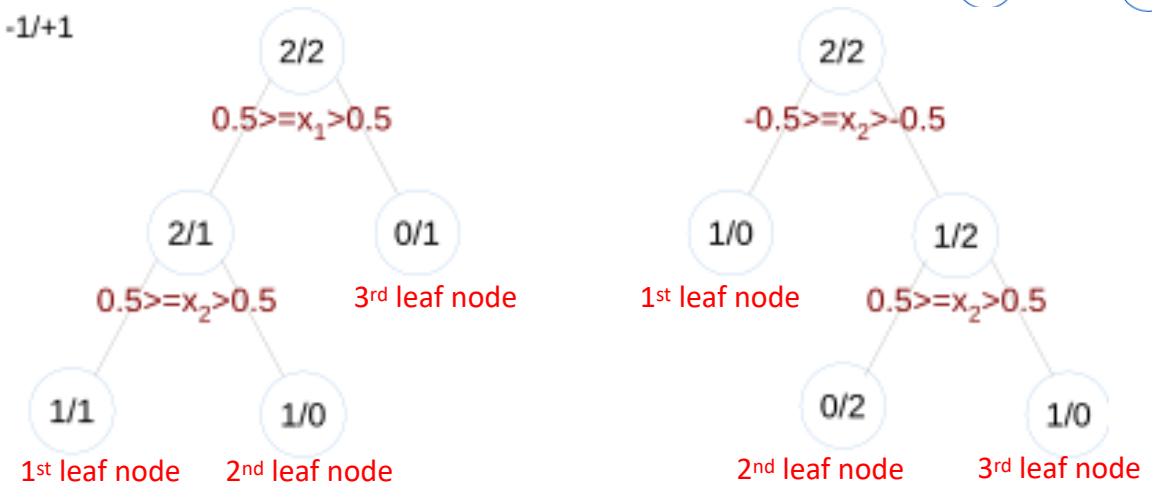


Figure 1: Two decision trees.



1/(1+1) = 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.

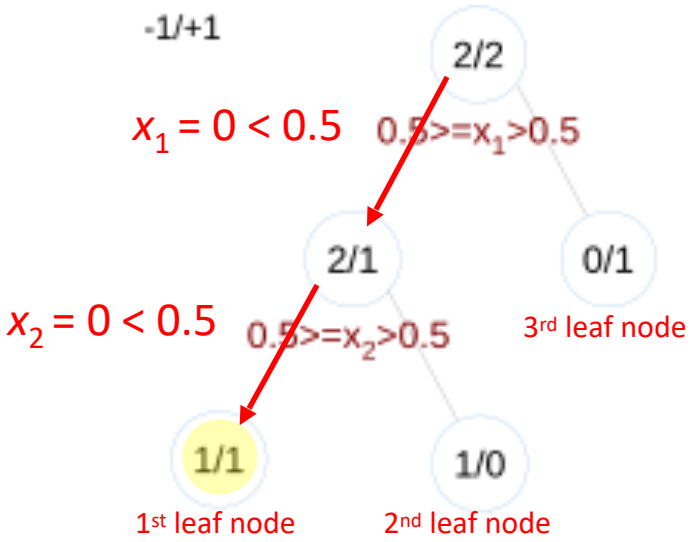
0/(1+1) = 0

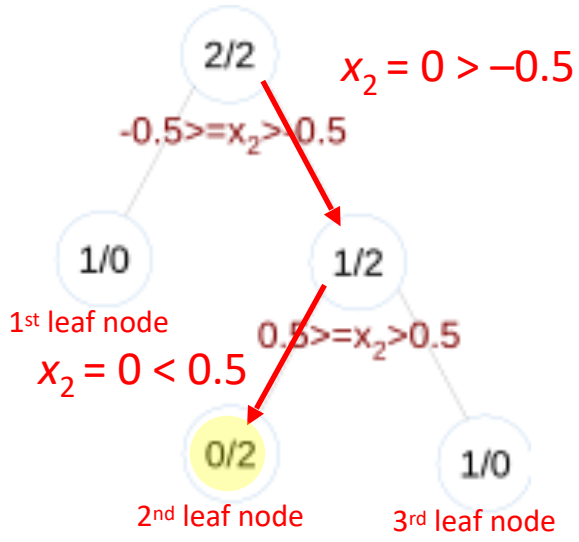
1/(1+0) = 1

Q3. Figure 1 shows two binary decision trees.

x_1, x_2

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



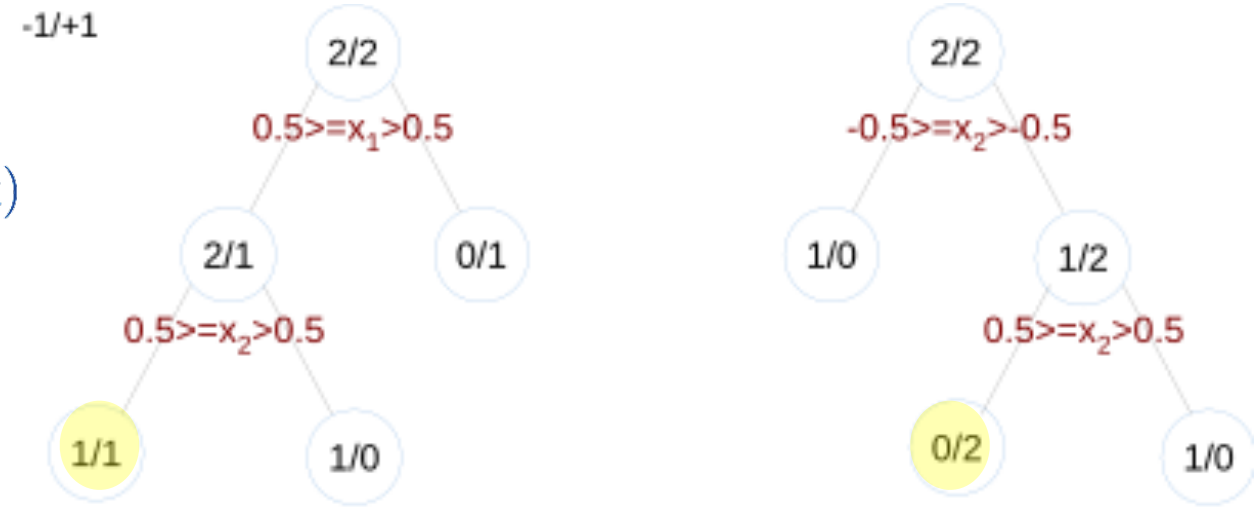


Which class **x** = (0, 0) belongs to?

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?

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



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)

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

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

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

So new sample is in class $+1$.

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

Tutorial 10: Solutions

Q1. Apply the K-means algorithm to the following dataset, $\mathbf{S}$, to find 2 natural groupings (K = 2) using the Euclidean distance criterion. Start with initial values of cluster centres of $\mathbf{m}_1$ and $\mathbf{m}_2$ accordingly.

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}, \mathbf{m}_1 = \begin{bmatrix} -1 \\ 3 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

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

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	$\sqrt{40}$	?
$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$	$\sqrt{5}$	5	1
$\begin{bmatrix} 0 \\ 5 \end{bmatrix}$	$\sqrt{5}$	$\sqrt{41}$	?
$\begin{bmatrix} 4 \\ -1 \end{bmatrix}$	$\sqrt{41}$	$\sqrt{5}$	2
$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	5	$\sqrt{5}$	?
$\begin{bmatrix} 5 \\ 1 \end{bmatrix}$	$\sqrt{40}$	0	2

New cluster centres:

$\mathbf{m}_1 =$?

$\mathbf{m}_2 =$

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, $\mathbf{S}$, to find 2 natural groupings (K = 2) using the Euclidean distance criterion. Start with initial values of cluster centres of $\mathbf{m}_1$ and $\mathbf{m}_2$ accordingly.

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}, \mathbf{m}_1 = \begin{bmatrix} -1 \\ 3 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

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

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	?
$\begin{bmatrix} 0 \\ 5 \end{bmatrix}$	1	$\sqrt{41}$	1
$\begin{bmatrix} 4 \\ -1 \end{bmatrix}$	$\sqrt{41}$	1	?
$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	5	1	2
$\begin{bmatrix} 5 \\ 1 \end{bmatrix}$	$\sqrt{34}$	$\sqrt{2}$	?

New cluster centres:

$\mathbf{m}_1 =$?

$\mathbf{m}_2 =$

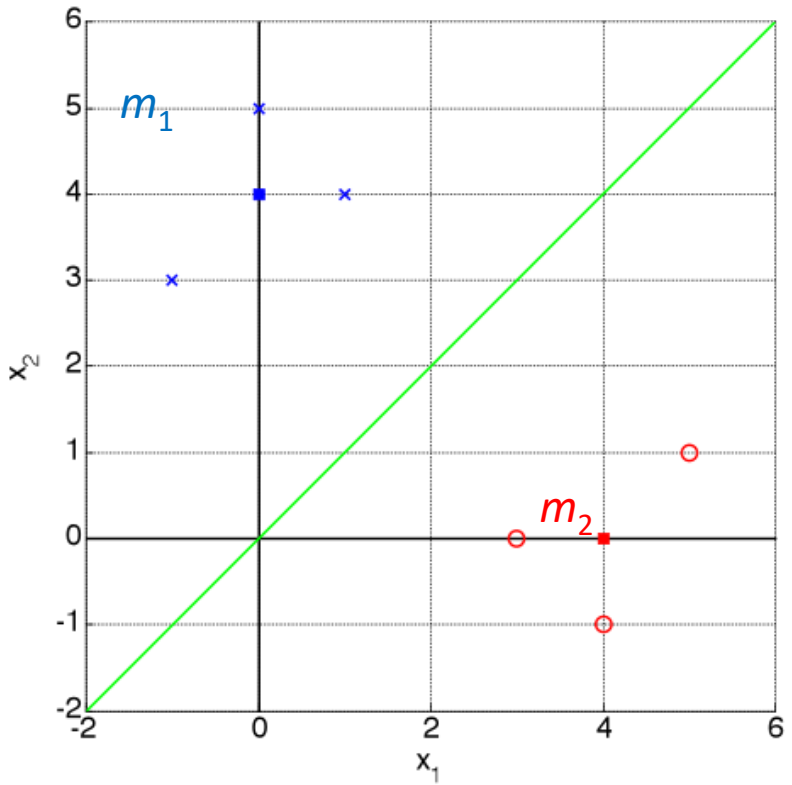
Q1. Apply the K-means algorithm to the following dataset, $\mathbf{S}$, to find 2 natural groupings (K = 2) using the Euclidean distance criterion. Start with initial values of cluster centres of $\mathbf{m}_1$ and $\mathbf{m}_2$ accordingly.

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}, \mathbf{m}_1 = \begin{bmatrix} -1 \\ 3 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

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

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 $\mathbf{X} = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, 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 $\mathbf{x} = \begin{bmatrix} 3 \\ -2 \\ 5 \end{bmatrix}$ (after removing the mean) using both sets of clusters centres.

Q2. Consider the dataset $\mathbf{X} = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, 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: $\mathbf{x}_{KL} = \hat{\mathbf{V}}^T (\mathbf{x} - \mathbf{m})$

$$\mathbf{m} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix},$$

$$\mathbf{X}_m = \begin{bmatrix} 3 & -1 & 1 & -3 \\ 1 & -3 & 3 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} 5 & 3 & 0 \\ 3 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} -0.7071 & -0.7071 & 0 \\ -0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

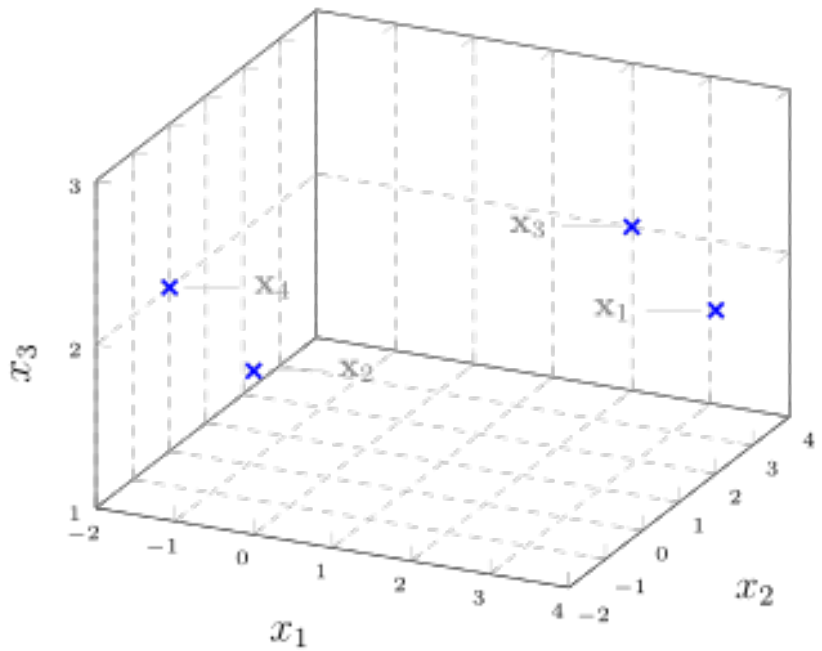
$$\hat{\mathbf{V}}^T = \begin{bmatrix} -0.7071 & -0.7071 & 0 \\ -0.7071 & 0.7071 & 0 \end{bmatrix}$$

$$\mathbf{X}_{KL} = \begin{bmatrix} -2.8284 & 2.8284 & -2.8284 & 2.8284 \\ -1.4142 & -1.4142 & 1.4142 & 1.4142 \end{bmatrix}$$ (using two components)

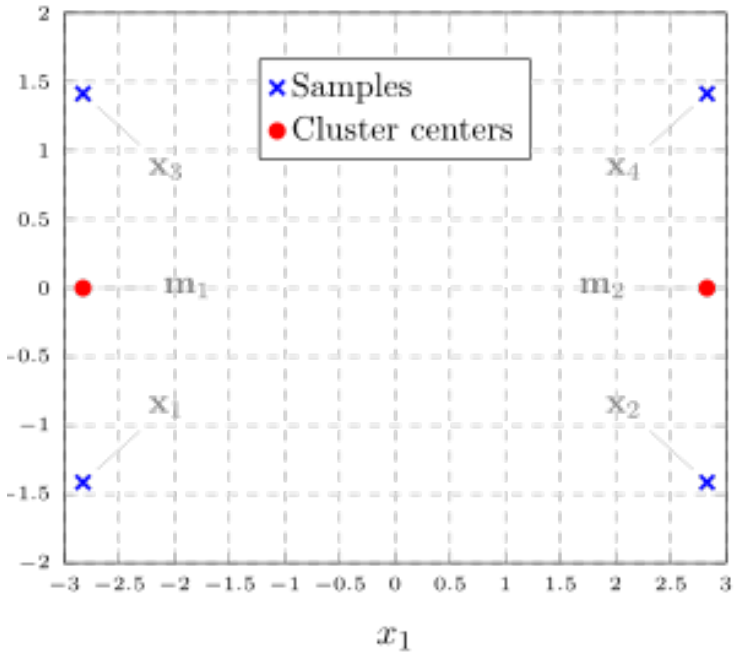
Q2. Consider the dataset $\mathbf{X} = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, use principal component analysis (PCA) to:

$\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3 \ \mathbf{x}_4$

a. project the data onto 2D plane,



(a) 3D plot of samples.



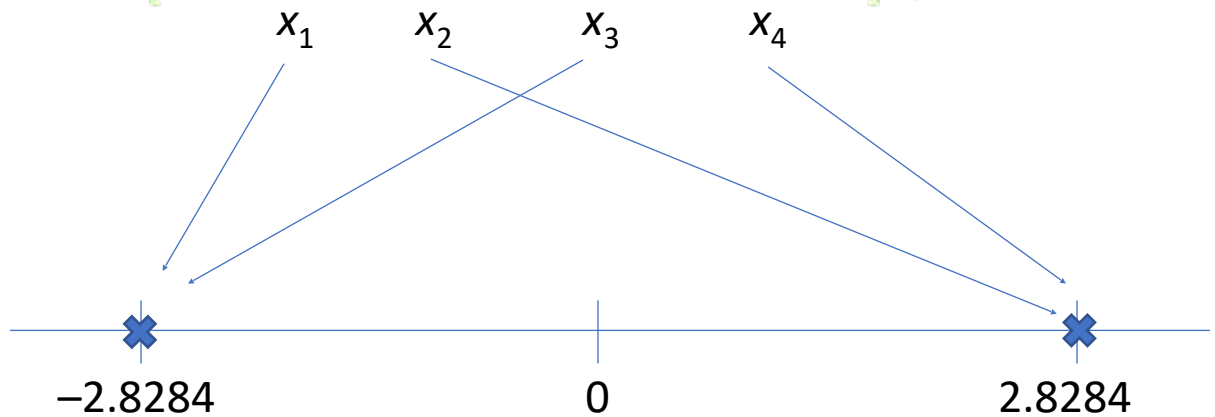
(b) 2D plot of transformed samples.

$$\mathbf{X}_{KL} = \begin{bmatrix} -2.8284 & 2.8284 & -2.8284 & 2.8284 \\ -1.4142 & -1.4142 & 1.4142 & 1.4142 \end{bmatrix} \text{ (using two components)}$$

Q2. Consider the dataset $\mathbf{X} = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, 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 $\mathbf{X} = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, use principal component analysis (PCA) to:

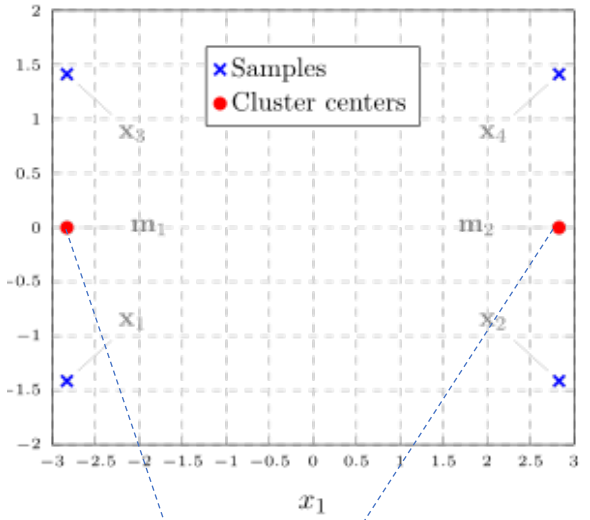
c. The dataset consists of samples from two classes. By observation, find the cluster centres based on the transformed 2D and 1D samples. Classify the new data $\mathbf{x} = \begin{bmatrix} 3 \\ -2 \\ 5 \end{bmatrix}$ (after removing the mean) using both sets of clusters centres.

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:

$$\mathbf{x}_{LT} = \hat{\mathbf{V}}^T\mathbf{x} = \begin{bmatrix} -0.7071 & -0.7071 & 0 \\ -0.7071 & 0.7071 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \\ 5 \end{bmatrix} = \begin{bmatrix} -0.7071 \\ -3.5355 \end{bmatrix}.$$



(b) 2D plot of transformed samples.

4.1231

5

(–0.7071, –3.5355)

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 ?

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

Q2. Consider the dataset $\mathbf{X} = \begin{bmatrix} 4 & 0 & 2 & -2 \\ 2 & -2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$, use principal component analysis (PCA) to:

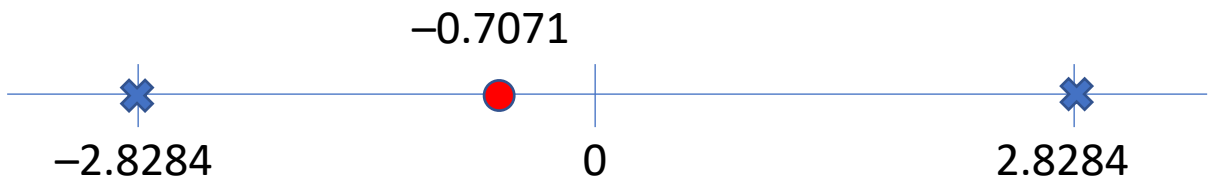
c. The dataset consists of samples from two classes. By observation, find the cluster centres based on the transformed 2D and 1D samples. Classify the new data $\mathbf{x} = \begin{bmatrix} 3 \\ -2 \\ 5 \end{bmatrix}$ (after removing the mean) using both sets of clusters centres.

Form the 1D plot, we choose:
$\mathbf{m}_1 = \begin{bmatrix} -2.8284 \end{bmatrix}$ and $\mathbf{m}_2 = \begin{bmatrix} 2.8284 \end{bmatrix}$.

Transformed data:

$$\mathbf{x}_{LT} = \hat{\mathbf{V}}^T\mathbf{x} = \begin{bmatrix} -0.7071 & -0.7071 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \\ 5 \end{bmatrix} = \begin{bmatrix} -0.7071 \end{bmatrix}.$$

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 ?



Q3. Competitive learning algorithm (without normalisation) is employed to find 3 cluster centres for the dataset $\mathbf{S}$ in Q1. The initial cluster centres are chosen as $\mathbf{x}_1/2$, $\mathbf{x}_3/2$ and $\mathbf{x}_5/2$, and $\eta = 0.1$. The algorithm is run for 5 iterations and the samples are chosen in the order of $\mathbf{x}_3$, $\mathbf{x}_1$, $\mathbf{x}_1$, $\mathbf{x}_5$, $\mathbf{x}_6$ where the left most data is chosen first.

a. Find the cluster centres for each iteration.

b. Plot the samples $\mathbf{S}$ and cluster centres in a figure. Classify the samples.

c. Classify the new data $\mathbf{x} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$.

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$$

Q3. Competitive learning algorithm (without normalisation) is employed to find 3 cluster centres for the dataset $\mathbf{S}$ in Q1. The initial cluster centres are chosen as $\mathbf{x}_1/2$, $\mathbf{x}_3/2$ and $\mathbf{x}_5/2$, and $\eta = 0.1$. The algorithm is run for 5 iterations and the samples are chosen in the order of $\mathbf{x}_3$, $\mathbf{x}_1$, $\mathbf{x}_1$, $\mathbf{x}_5$, $\mathbf{x}_6$ where the left most data is chosen first.

a. Find the cluster centres for each iteration.

$$\mathbf{m}_1 = \begin{bmatrix} -0.5 \\ 1.5 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 0 \\ 2.5 \end{bmatrix}, \mathbf{m}_3 = \begin{bmatrix} 1.5 \\ 0 \end{bmatrix}.$$

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$$

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

Updated $\mathbf{m}_2$ is used

Distance between sample and (updated) cluster centre

Identify the cluster centre with the shortest distance from sample

Updated the identified cluster centre

Iteration	Sample $\mathbf{x}$	$\|\mathbf{x} - \mathbf{m}_1\|$	$\|\mathbf{x} - \mathbf{m}_2\|$	$\|\mathbf{x} - \mathbf{m}_3\|$	$j \leftarrow \arg\min_{j'} \|\mathbf{x} - \mathbf{m}_{j'}\|$	$\mathbf{m}_3 \leftarrow \mathbf{m}_3 + \eta(\mathbf{x} - \mathbf{m}_3)$
1	$\begin{bmatrix} 0 \\ 5 \end{bmatrix}$					m? ←
2	$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$					
3	$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$					
4	$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$					
5	$\begin{bmatrix} 5 \\ 1 \end{bmatrix}$					

Updated $\mathbf{m}_3$ is used

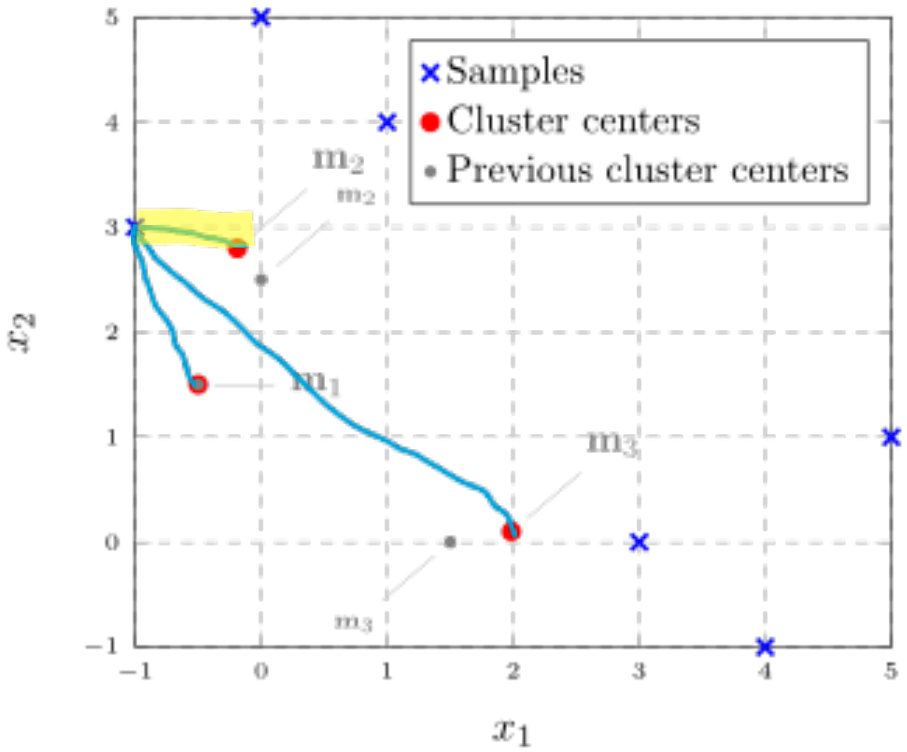
After 5 iterations, we have $\mathbf{m}_1 = \begin{bmatrix} -0.5 \\ 1.5 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} -0.19 \\ 2.7975 \end{bmatrix}, \mathbf{m}_3 = \begin{bmatrix} 1.985 \\ 0.1 \end{bmatrix}.$

Q3. Competitive learning algorithm (without normalisation) is employed to find 3 cluster centres for the dataset $\mathbf{S}$ in Q1. The initial cluster centres are chosen as $\mathbf{x}_1/2$, $\mathbf{x}_3/2$ and $\mathbf{x}_5/2$, and $\eta = 0.1$. The algorithm is run for 5 iterations and the samples are chosen in the order of $\mathbf{x}_3$, $\mathbf{x}_1$, $\mathbf{x}_1$, $\mathbf{x}_5$, $\mathbf{x}_6$ where the left most data is chosen first.

b. Plot the samples $\mathbf{S}$ and cluster centres in a figure. Classify the samples.

$$\mathbf{m}_1 = \begin{bmatrix} -0.5 \\ 1.5 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} -0.19 \\ 2.7975 \end{bmatrix}, \mathbf{m}_3 = \begin{bmatrix} 1.985 \\ 0.1 \end{bmatrix}.$$

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$$

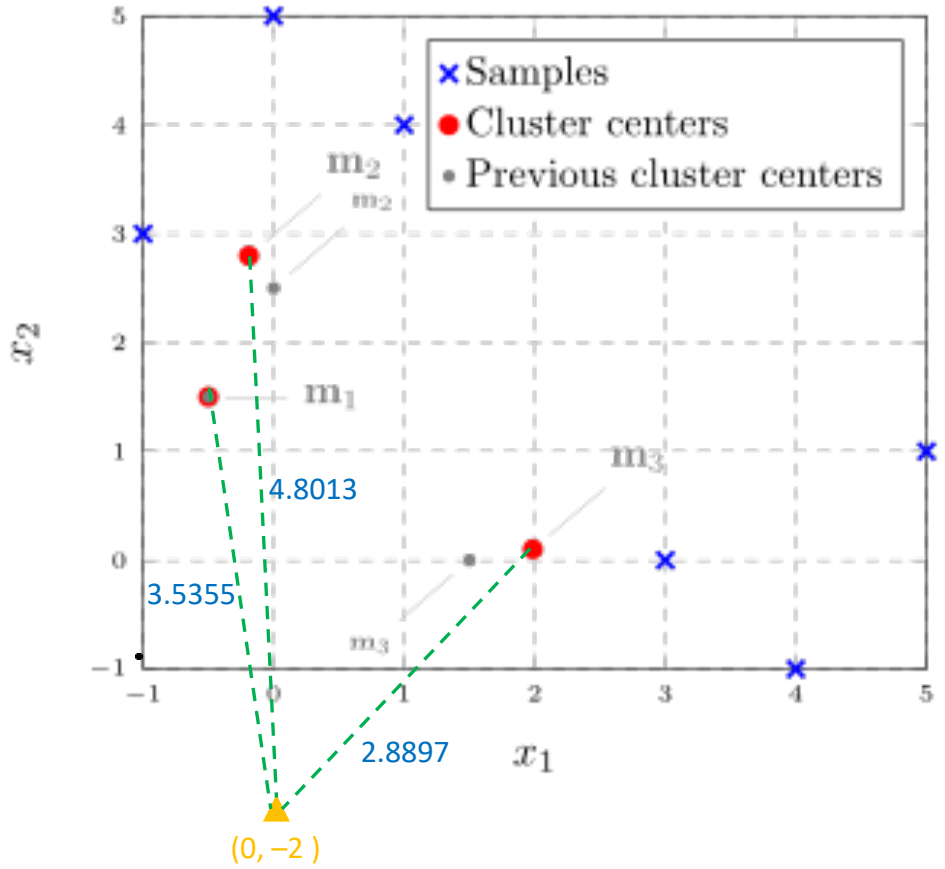


Sample $\mathbf{x}$	Class
$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	?
$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$	?
$\begin{bmatrix} 0 \\ 5 \end{bmatrix}$	?
$\begin{bmatrix} 4 \\ -1 \end{bmatrix}$	?
$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	?
$\begin{bmatrix} 5 \\ 1 \end{bmatrix}$	?

Q3. Competitive learning algorithm (without normalisation) is employed to find 3 cluster centres for the dataset $\mathbf{S}$ in Q1. The initial cluster centres are chosen as $\mathbf{x}_1/2$, $\mathbf{x}_3/2$ and $\mathbf{x}_5/2$, and $\eta = 0.1$. The algorithm is run for 5 iterations and the samples are chosen in the order of $\mathbf{x}_3$, $\mathbf{x}_1$, $\mathbf{x}_1$, $\mathbf{x}_5$, $\mathbf{x}_6$ where the left most data is chosen first.

c. Classify the new data $\mathbf{x} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$.

= which class?



Q4. Basic leader follower algorithm (without normalisation), with $\theta = 3$ and $\eta = 0.5$, is employed to find the cluster centres of the dataset $\mathbf{S}$ in Q1. The algorithm is run for 5 iterations and the samples are chosen in the order of $\mathbf{x}_3$, $\mathbf{x}_1$, $\mathbf{x}_1$, $\mathbf{x}_5$, $\mathbf{x}_6$ where the left most data is chosen first.

a. Find the cluster centres for each iteration.

b. Classify the samples in $\mathbf{S}$.

c. Classify the new data $\mathbf{x} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$.

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$$

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 $\mathbf{S}$ in Q1. The algorithm is run for 5 iterations and the samples are chosen in the order of $\mathbf{x}_3$, $\mathbf{x}_1$, $\mathbf{x}_1$, $\mathbf{x}_5$, $\mathbf{x}_6$ 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

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	$\begin{bmatrix} 0 \\ 5 \end{bmatrix}$	?	?	?	?	?
2	$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	?	?	?	?	?
3	$\begin{bmatrix} -1 \\ 3 \end{bmatrix}$	?	?	?	?	?
4	$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	?	?	?	?	?
5	$\begin{bmatrix} 5 \\ 1 \end{bmatrix}$	?	?	?	?	?

Note: In each iteration, the updated cluster centres are used.

After 5 iterations, we have $\mathbf{m}_1 = \begin{bmatrix} -0.75 \\ 3.5 \end{bmatrix}$, $\mathbf{m}_2 = \begin{bmatrix} 4 \\ 0.5 \end{bmatrix}$.

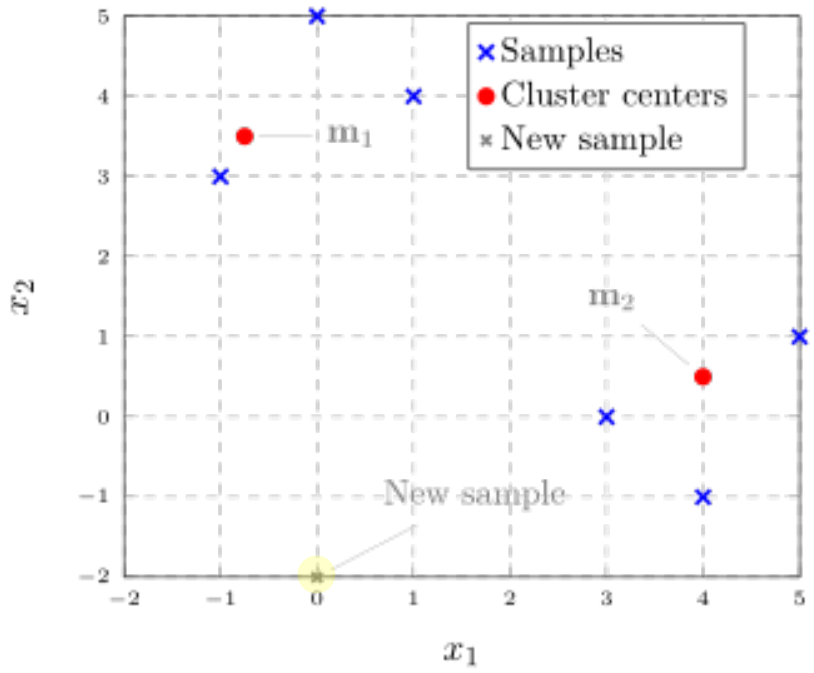
Q4. Basic leader follower algorithm (without normalisation), with $\theta = 3$ and $\eta = 0.5$, is employed to find the cluster centres of the dataset $\mathbf{S}$ in Q1. The algorithm is run for 5 iterations and the samples are chosen in the order of $\mathbf{x}_3$, $\mathbf{x}_1$, $\mathbf{x}_1$, $\mathbf{x}_5$, $\mathbf{x}_6$ where the left most data is chosen first.

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$$

b. Classify the samples in $\mathbf{S}$.

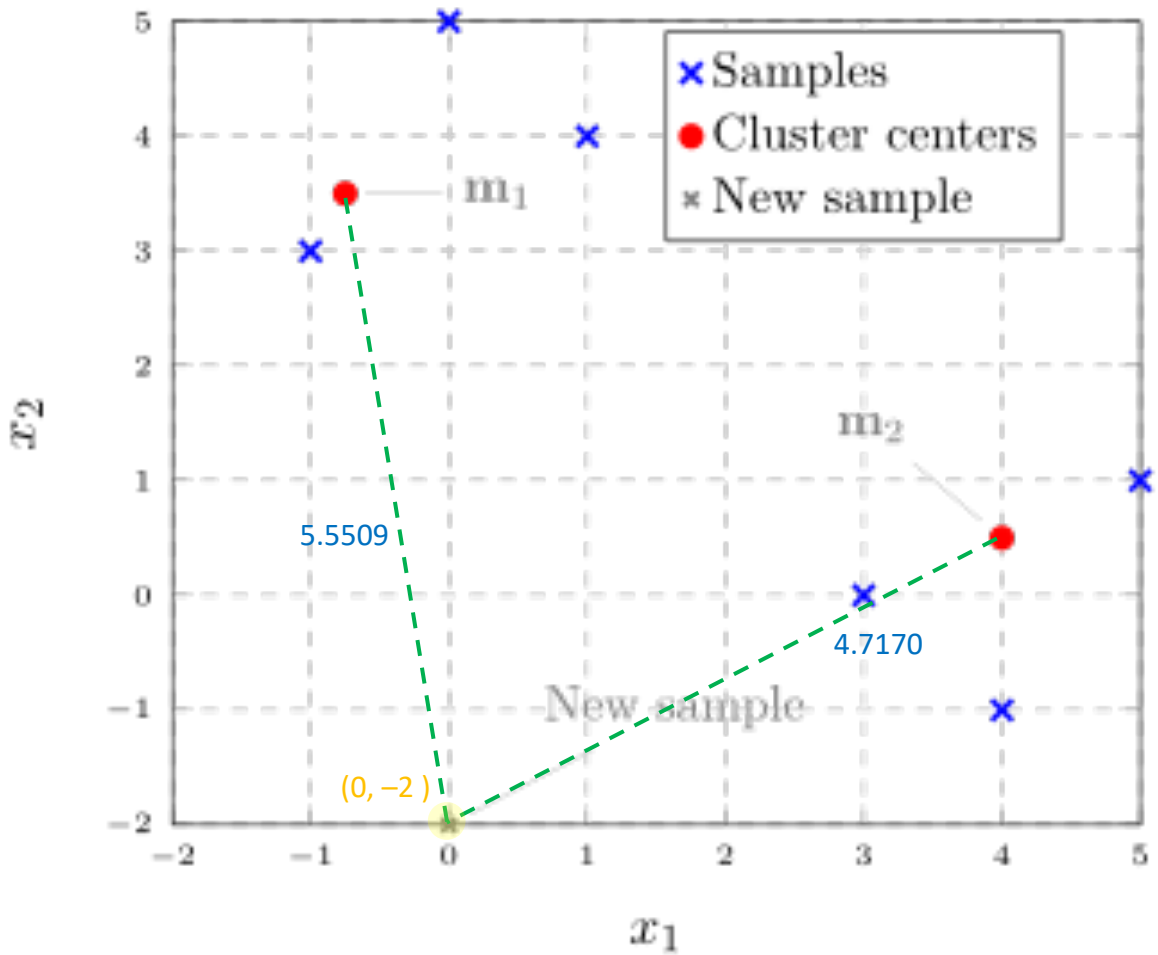
After 5 iterations, we have $\mathbf{m}_1 = \begin{bmatrix} -0.75 \\ 3.5 \end{bmatrix}$, $\mathbf{m}_2 = \begin{bmatrix} 4 \\ 0.5 \end{bmatrix}$.

Sample $\mathbf{x}$	$\|\mathbf{x} - \mathbf{m}_1\|$	$\|\mathbf{x} - \mathbf{m}_2\|$	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 $\mathbf{S}$ in Q1. The algorithm is run for 5 iterations and the samples are chosen in the order of $\mathbf{x}_3$, $\mathbf{x}_1$, $\mathbf{x}_1$, $\mathbf{x}_5$, $\mathbf{x}_6$ where the left most data is chosen first.

c. Classify the new data $\mathbf{x} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$.



New sample: $\mathbf{x} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$

Euclidean distance:

$\|\mathbf{x} - \mathbf{m}_1\| = 5.5509,$

$\|\mathbf{x} - \mathbf{m}_2\| = 4.7170.$

As $\mathbf{x}$ is closer to $\mathbf{m}_2$, it is classified as class 2.

Q5. Apply the Fuzzy K-means algorithm, with $K = 2$, to the dataset: $\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$.

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 $(K = c)$, 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

Q5. Apply the Fuzzy K-means algorithm, with $K = 2$, to the dataset: $\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$.
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 ($K = c$), 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

$$\mu'_{11} = \frac{\mu_{11}}{\mu_{11}+\mu_{21}} = \frac{1}{1+0} = 1$$
$$\mu'_{21} = \frac{\mu_{21}}{\mu_{11}+\mu_{21}} = \frac{0}{1+0} = 0$$
$$\mu'_{11} + \mu'_{21} = 1$$

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:

$$\mu = \begin{bmatrix} \mu_{11} & \mu_{12} & \mu_{13} & \mu_{14} & \mu_{15} & \mu_{16} \\ \mu_{21} & \mu_{22} & \mu_{23} & \mu_{24} & \mu_{25} & \mu_{26} \end{bmatrix} = \begin{bmatrix} 1 & ? & & & & \\ 0 & ? & & & & \end{bmatrix}.$$

Q5. Apply the Fuzzy K-means algorithm, with K = 2, to the dataset: $\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$.
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 $(K = c)$, 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

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: $\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$.

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 $(K = c)$, 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

Normalise memberships:

$$\mu = \begin{bmatrix} \mu_{11} & \mu_{12} & \mu_{13} & \mu_{14} & \mu_{15} & \mu_{16} \\ \mu_{21} & \mu_{22} & \mu_{23} & \mu_{24} & \mu_{25} & \mu_{26} \end{bmatrix} = \begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}.$$

$$\mathbf{m}_i = \frac{\sum_{j=1}^{n} \mu_{ij}^b \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^b} = \frac{\sum_{j=1}^{6} \mu_{ij}^2 \mathbf{x}_j}{\sum_{j=1}^{6} \mu_{ij}^2} = \frac{\mu_{i1}^2 \mathbf{x}_1 + \mu_{i2}^2 \mathbf{x}_2 + \mu_{i3}^2 \mathbf{x}_3 + \mu_{i4}^2 \mathbf{x}_4 + \mu_{i5}^2 \mathbf{x}_5 + \mu_{i6}^2 \mathbf{x}_6}{\mu_{i1}^2 + \mu_{i2}^2 + \mu_{i3}^2 + \mu_{i4}^2 + \mu_{i5}^2 + \mu_{i6}^2}$$

$$\mathbf{m}_1 = \frac{? \begin{bmatrix} -1 \\ 3 \end{bmatrix} + 0.5^2 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + \quad ? \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.5^2 \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 0.5^2 \begin{bmatrix} 3 \\ 0 \end{bmatrix} + ? \begin{bmatrix} 5 \\ 1 \end{bmatrix}}{1^2 + 0.5^2 + 0.5^2 + 0.5^2 + 0.5^2 + 0^2} = \begin{bmatrix} 0.5 \\ 2.5 \end{bmatrix}$$

?

$$\mathbf{m}_2 = \frac{0^2 \begin{bmatrix} -1 \\ 3 \end{bmatrix} + 0.5^2 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 0.5^2 \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.5^2 \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 0.5^2 \begin{bmatrix} 3 \\ 0 \end{bmatrix} + 1^2 \begin{bmatrix} 5 \\ 1 \end{bmatrix}}{0^2 + 0.5^2 + 0.5^2 + 0.5^2 + 0.5^2 + 1^2} = \begin{bmatrix} 3.5 \\ 1.5 \end{bmatrix}$$

Q5. Apply the Fuzzy K-means algorithm, with K = 2, to the dataset: $\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & 4 & 3 & 5 \\ 3 & 4 & 5 & -1 & 0 & 1 \end{bmatrix}$. 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: $\mu_{ij} = \dfrac{\left(1/\|\mathbf{x}_j - \mathbf{m}_i\|\right)^{\frac{2}{b-1}}}{\sum_{r=1}^{c} \left(1/\|\mathbf{x}_j - \mathbf{m}_r\|\right)^{\frac{2}{b-1}}}$

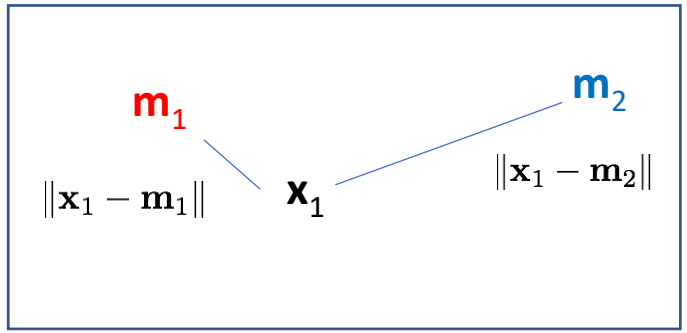
until no change in $\mathbf{m}_i$

return $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$

end

Calculate membership values:

$$\mu_{ij} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^{\frac{2}{b-1}}}{\sum_{r=1}^{c}(1/\|\mathbf{x}_j - \mathbf{m}_r\|)^{\frac{2}{b-1}}} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^2}{\sum_{r=1}^{2}(1/\|\mathbf{x}_j - \mathbf{m}_r\|)^2} = \_\_\_\_\_$$



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

$$\mu_{11} = \frac{(1/\|\mathbf{x}_1 - \mathbf{m}_1\|)^2}{(1/\|\mathbf{x}_1 - \mathbf{m}_1\|)^2 + (1/\|\mathbf{x}_1 - \mathbf{m}_2\|)^2} = \frac{0.4}{0.4 + 0.0444} = 0.9$$

$$\mu_{21} = \frac{(1/\|\mathbf{x}_1 - \mathbf{m}_2\|)^2}{(1/\|\mathbf{x}_1 - \mathbf{m}_2\|)^2 + (1/\|\mathbf{x}_1 - \mathbf{m}_2\|)^2} = \frac{0.0444}{0.4 + 0.0444} = 0.1$$

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	$\begin{bmatrix}-1\\3\end{bmatrix}$	1.5811	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:

$$\mathbf{m}_i = \frac{\sum_{j=1}^{n} \mu_{ij}^b \mathbf{x}_j}{\sum_{j=1}^{n} \mu_{ij}^b} = \frac{\sum_{j=1}^{6} \mu_{ij}^2 \mathbf{x}_j}{\sum_{j=1}^{6} \mu_{ij}^b} = \frac{\mu_{i1}^b \mathbf{x}_1 + \mu_{i2}^b \mathbf{x}_2 + \mu_{i3}^b \mathbf{x}_3 + \mu_{i4}^b \mathbf{x}_4 + \mu_{i5}^b \mathbf{x}_5 + \mu_{i6}^b \mathbf{x}_6}{\mu_{i1}^b + \mu_{i2}^b + \mu_{i3}^b + \mu_{i4}^b + \mu_{i5}^b + \mu_{i6}^b}$$

$$\mathbf{m}_1 = \frac{0.9^2 \begin{bmatrix} -1 \\ 3 \end{bmatrix} + 0.8333^2 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 0.7903^2 \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.2097^2 \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 0.1667^2 \begin{bmatrix} 3 \\ 0 \end{bmatrix} + 0.1^2 \begin{bmatrix} 5 \\ 1 \end{bmatrix}}{0.9^2 + 0.8333^2 + 0.7903^2 + 0.2097^2 + 0.1667^2 + 0.1^2}$$

$$= \begin{bmatrix} 0.0876 \\ 3.7529 \end{bmatrix}$$

$$\mathbf{m}_2 = \frac{0.1^2 \begin{bmatrix} -1 \\ 3 \end{bmatrix} + 0.1667^2 \begin{bmatrix} 1 \\ 4 \end{bmatrix} + 0.2097^2 \begin{bmatrix} 0 \\ 5 \end{bmatrix} + 0.7903^2 \begin{bmatrix} 4 \\ -1 \end{bmatrix} + 0.8333^2 \begin{bmatrix} 3 \\ 0 \end{bmatrix} + 0.9^2 \begin{bmatrix} 5 \\ 1 \end{bmatrix}}{0.1^2 + 0.1667^2 + 0.2097^2 + 0.7903^2 + 0.8333^2 + 0.9^2}$$

$$= \begin{bmatrix} 3.9124 \\ 0.2471 \end{bmatrix}$$

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:

$$\mu_{ij} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^{\frac{2}{b-1}}}{\sum_{r=1}^{c} (1/\|\mathbf{x}_j - \mathbf{m}_r\|)^{\frac{2}{b-1}}} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^2}{\sum_{r=1}^{2} (1/\|\mathbf{x}_j - \mathbf{m}_r\|)^2} = \frac{(1/\|\mathbf{x}_j - \mathbf{m}_i\|)^2}{(1/\|\mathbf{x}_j - \mathbf{m}_1\|)^2 + (1/\|\mathbf{x}_j - \mathbf{m}_2\|)^2}$$

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

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: $\mathbf{m}_1 = \begin{bmatrix} 0.0187 \\ 4.0009 \end{bmatrix}$, $\mathbf{m}_2 = \begin{bmatrix} 3.9813 \\ -0.0009 \end{bmatrix}$.

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

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:

$$\mathbf{m}_1 = \begin{bmatrix} 0.0187 \\ 4.0009 \end{bmatrix}, \mathbf{m}_2 = \begin{bmatrix} 3.9813 \\ -0.0009 \end{bmatrix}.$$

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, $\hat{c} \leftarrow n$, $\mathcal{D}_i \leftarrow \{\mathbf{x}_i\}$, $i = 1, 2, \ldots, n$
  do $\hat{c} = \hat{c} - 1$
    Find the nearest clusters, say, $\mathcal{D}_i$ and $\mathcal{D}_j$
    Merge $\mathcal{D}_i$ and $\mathcal{D}_j$
  until $c = \hat{c}$
  return c clusters
end

c: target number of clusters; $\hat{c}$: current number of clusters.

Initialisation: make each sample a cluster.

Iteration 1

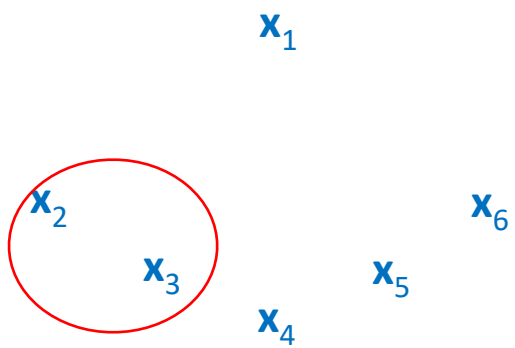
Find most similar clusters:

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).

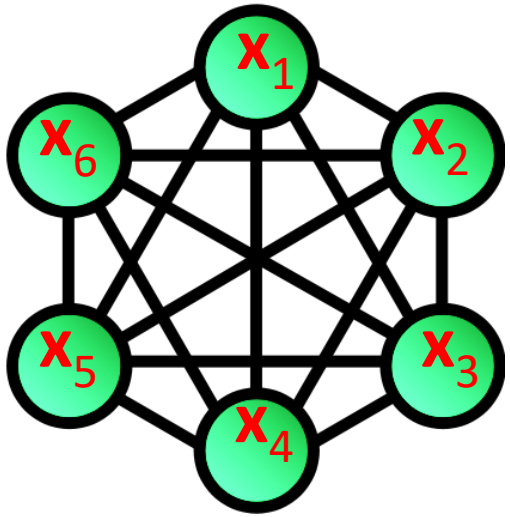
Merge these clusters: clusters 3 and 2 are a new cluster called "2+3"



Pseudo Code: Algorithm for Agglomerative hierarchical clustering

begin initialise c, $\hat{c} \leftarrow n$, $\mathscr{D}_i \leftarrow \{\mathbf{x}_i\}$, $i = 1, 2, \ldots, n$
  do $\hat{c} = \hat{c} - 1$
    Find the nearest clusters, say, $\mathscr{D}_i$ and $\mathscr{D}_j$
    Merge $\mathscr{D}_i$ and $\mathscr{D}_j$
  until $c = \hat{c}$
  return c clusters
end

c: target number of clusters; $\hat{c}$: current number of clusters.



Pseudo Code: Algorithm for Agglomerative hierarchical clustering

begin initialise c, $\hat{c} \leftarrow n$, $\mathscr{D}_i \leftarrow \{\mathbf{x}_i\}$, $i = 1, 2, \ldots, n$
  do $\hat{c} = \hat{c} - 1$
    Find the nearest clusters, say, $\mathscr{D}_i$ and $\mathscr{D}_j$
    Merge $\mathscr{D}_i$ and $\mathscr{D}_j$
  until $c = \hat{c}$
  return c clusters
end

c: target number of clusters; $\hat{c}$: current number of clusters.

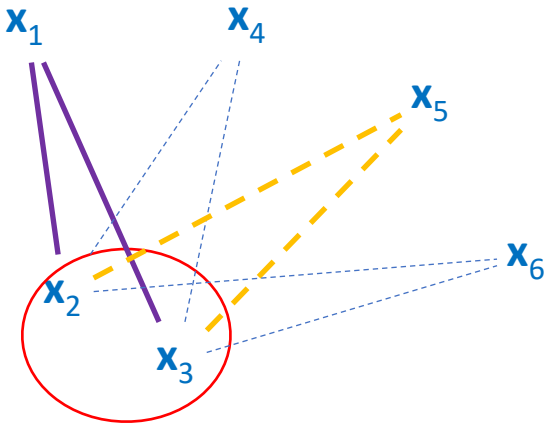
- $d_{min}(\mathscr{D}_i, \mathscr{D}_j) = \min_{\mathbf{x} \in \mathscr{D}_i, \mathbf{x}' = \mathscr{D}_j} \|\mathbf{x} - \mathbf{x}'\|$
- Single-link clustering - distance between clusters is the minimum distance between constituent samples

Iteration 2

Find most similar clusters:

Update proximity matrix

$\mathbf{x}^T$	cluster	1	2+3	4	5	6
$[-1, 3]$	1	-	-	-	-	-
$[1, 2]$, $[0, 1]$	2+3	2.2361	-	-	-	-
$[4, 0]$	4	5.8310	3.6056	-	-	-
$[5, 4]$	5	6.0828	4.4721	4.1231	-	-
$[3, 2]$	6	4.1231	2.0000	2.2361	2.8284	-

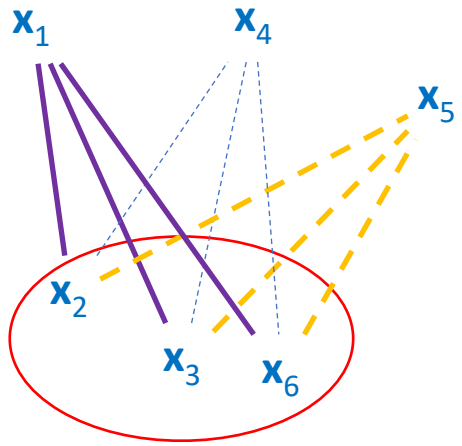


Which clusters are to be merged?

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.



Iteration 3

Find most similar clusters:

Update proximity matrix

$\mathbf{x}^T$	cluster	1	2+3+6	4	5
$[-1, 3]$	1	-	-	-	-
$[1, 2]$, $[0, 1]$, $[3, 2]$	2+3+6	2.2361	-	-	-
$[4, 0]$	4	5.8310	2.2361	-	-
$(5, 4)$	5	6.0828	2.8284	4.1231	-

Which clusters are to be merged?