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.

i. Hyper-Parameter

a value used by the learning algorithm in its search for 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

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

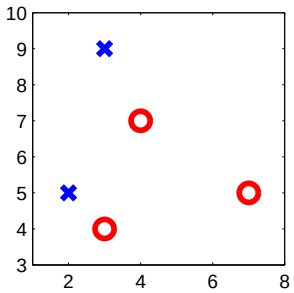
e. the f_1-score

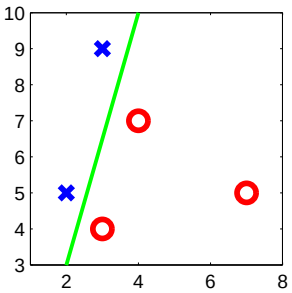
$$= \frac{2\times TP}{2\times TP+FP+FN} = \frac{2\times 3}{2\times 3+1+2} = \frac{6}{9} = \frac{2}{3}$$

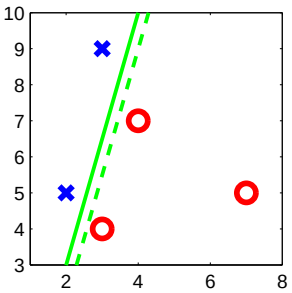
alternatively:f_1-score$= \frac{2\times recall\times precision}{recall+precision} = \frac{2\times\frac{3}{5}\times\frac{3}{4}}{\frac{3}{5}+\frac{3}{4}} = \frac{2\times\frac{9}{20}}{\frac{12}{20}+\frac{15}{20}} = \frac{2\times 9}{27} = \frac{2}{3}$

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)







2 Discriminant Functions

1. Consider a dichotomizer defined using the following 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 the following feature vectors: $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 2 \\ 2 \end{pmatrix}$, and $\begin{pmatrix} 3 \\ 3 \end{pmatrix}$.

At decision boundary $g(\mathbf{x}) = 0$, therefore decision boundary is defined by

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

$$(2 \;\; 1)\mathbf{x} - 5 = 0$$

$\mathbf{w}$ *has two elements, so we know 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.

Let's find where this line intercepts the axes.

$$(2 \;\; 1)\begin{pmatrix} x_1 \\ 0 \end{pmatrix} - 5 = 0$$

$$2x_1 - 5 = 0$$

$$x_1 = 2.5$$

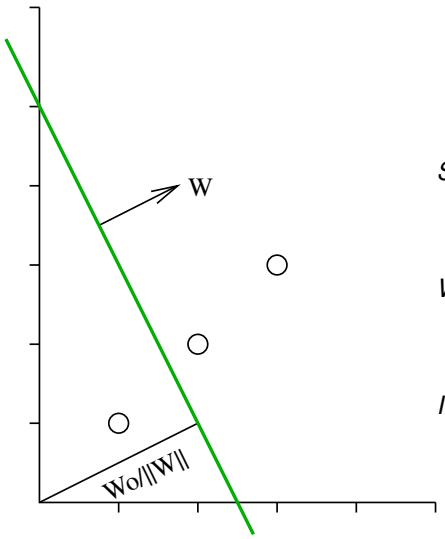
i.e., intercept at (2.5,0)

$$(2 \;\; 1)\begin{pmatrix} 0 \\ x_2 \end{pmatrix} - 5 = 0$$

$$x_2 - 5 = 0$$

$$x_2 = 5$$

i.e., intercept at (0,5)



So, hyperplane has equation:

$$x_2 = mx_1 + c = -2x_1 + 5$$

We could have got this by simply re-arranging

$$(2 \;\; 1)\mathbf{x} - 5 = 0$$

In general,

$$x_2 = mx_1 + c = \frac{-w_1}{w_2}x_1 + \frac{-w_0}{w_2}$$

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

$(2\ \ 1)\begin{pmatrix}1\\1\end{pmatrix} - 5 = 2 + 1 - 5 = -2$ *Therefore* $\begin{pmatrix}1\\1\end{pmatrix}$ *is in class 2.*

$(2\ \ 1)\begin{pmatrix}2\\2\end{pmatrix} - 5 = 4 + 2 - 5 = 1$ *Therefore* $\begin{pmatrix}2\\2\end{pmatrix}$ *is in class 1.*

$(2\ \ 1)\begin{pmatrix}3\\3\end{pmatrix} - 5 = 6 + 3 - 5 = 4$ *Therefore* $\begin{pmatrix}3\\3\end{pmatrix}$ *is in class 1.*

Note, the vector normal to the hyperplane points towards class 1. 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* $\frac{|g(\mathbf{x})|}{\|\mathbf{w}\|}$.

2. In augmented feature space, a dichotomizer is defined using the following linear discriminant function $g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$ **where** $\mathbf{a}^t = (-5, 2, 1)$ **and** $\mathbf{y} = \begin{pmatrix}1\\x\end{pmatrix}$**. Determine the class of the following feature vectors:** $\begin{pmatrix}1\\1\end{pmatrix}$**,** $\begin{pmatrix}2\\2\end{pmatrix}$**, and** $\begin{pmatrix}3\\3\end{pmatrix}$**.**

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

$(-5\ \ 2\ \ 1)\begin{pmatrix}1\\1\\1\end{pmatrix} = -5 + 2 + 1 = -2$ *Therefore* $\begin{pmatrix}1\\1\end{pmatrix}$ *is in class 2.*

$(-5\ \ 2\ \ 1)\begin{pmatrix}1\\2\\2\end{pmatrix} = -5 + 4 + 2 = 1$ *Therefore* $\begin{pmatrix}2\\2\end{pmatrix}$ *is in class 1.*

$(-5\ \ 2\ \ 1)\begin{pmatrix}1\\3\\3\end{pmatrix} = -5 + 6 + 3 = 4$ *Therefore* $\begin{pmatrix}3\\3\end{pmatrix}$ *is in class 1.*

Note, same as previous question, just using augmented vectors.

3. Consider a 3-dimensional feature space and quadratic discriminant function, $g(\mathbf{x})$**, where:**

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

This discriminant function defines two classes, such that $g(\mathbf{x}) > 0$ **if** $\mathbf{x} \in \omega_1$ **and** $g(\mathbf{x}) \leq 0$ **if** $\mathbf{x} \in \omega_2$**. Determine the class of each of the following pattern vectors:** $(1\ \ 1\ \ 1)^t$**,** $(-1\ \ 0\ \ 3)^t$**, and** $(-1\ \ 0\ \ 0)^t$**.**

$g(\mathbf{x} = (1\ \ 1\ \ 1)^t) = 1^2 - 1^2 + 2 + 4 + 3 - 2 + 2 = 9$
hence $(1\ \ 1\ \ 1)^t$ *is in class 1.*

$g(\mathbf{x} = (-1\ \ 0\ \ 3)^t) = (-1)^2 - 3^2 + 0 + 0 - 3 - 0 + 2 = -9$
hence $(-1\ \ 0\ \ 3)^t$ *is in class 2.*

$g(\mathbf{x} = (-1 \;\; 0 \;\; 0)^t) = (-1)^2 - 0 + 0 + 0 - 3 - 0 + 2 = 0$
hence $(-1 \;\; 0 \;\; 0)^t$ *is in class 2.*

4. Consider a dichotomizer defined in a 2-dimensional feature space using a quadratic discriminant function, $g(\mathbf{x})$, where:

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

Classify the following feature vectors: $(0,-1)^t$, and $(1,1)^t$, when:

i) $\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix}$, $\mathbf{b} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, **and** $c = -3$.

ii) $\mathbf{A} = \begin{pmatrix} -2 & 5 \\ 5 & -8 \end{pmatrix}$, $\mathbf{b} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, **and** $c = -3$.

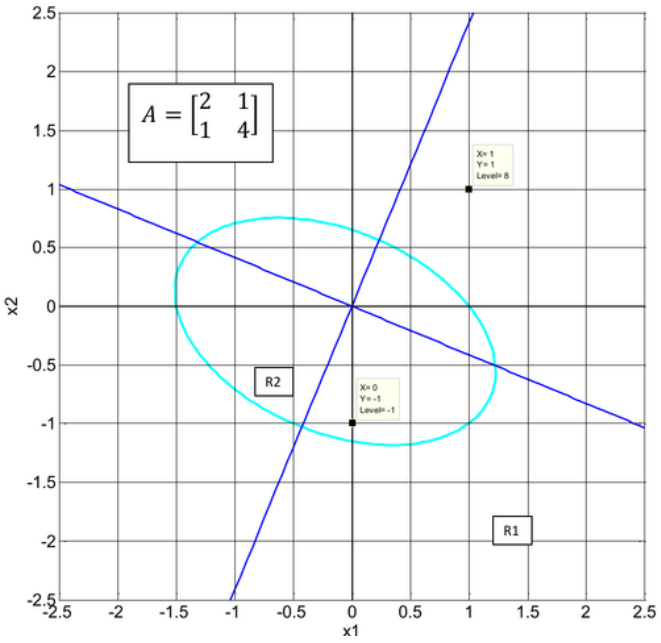
i)

$$\begin{aligned} g(\mathbf{x}) &= (x_1, x_2)\begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (x_1, x_2)\begin{pmatrix} 1 \\ 2 \end{pmatrix} - 3 \\ &= (2x_1 + x_2, x_1 + 4x_2)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + x_1 + 2x_2 - 3 \\ &= 2x_1^2 + x_1x_2 + x_1x_2 + 4x_2^2 + x_1 + 2x_2 - 3 \\ &= 2x_1^2 + 4x_2^2 + 2x_1x_2 + x_1 + 2x_2 - 3 \end{aligned}$$

When $\mathbf{x} = (0,-1)^t$, $g(\mathbf{x}) = 0 + 4(-1)^2 + 0 + 0 + 2(-1) - 3 = -1$
$g(\mathbf{x}) \leq 0$ *so* $\mathbf{x}$ *is in class 2.*

When $\mathbf{x} = (1,1)^t$, $g(\mathbf{x}) = 2 + 4 + 2 + 1 + 2 - 3 = 8$
$g(\mathbf{x}) > 0$ *so* $\mathbf{x}$ *is in class 1.*

The decision surface looks like this:



ii)

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

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

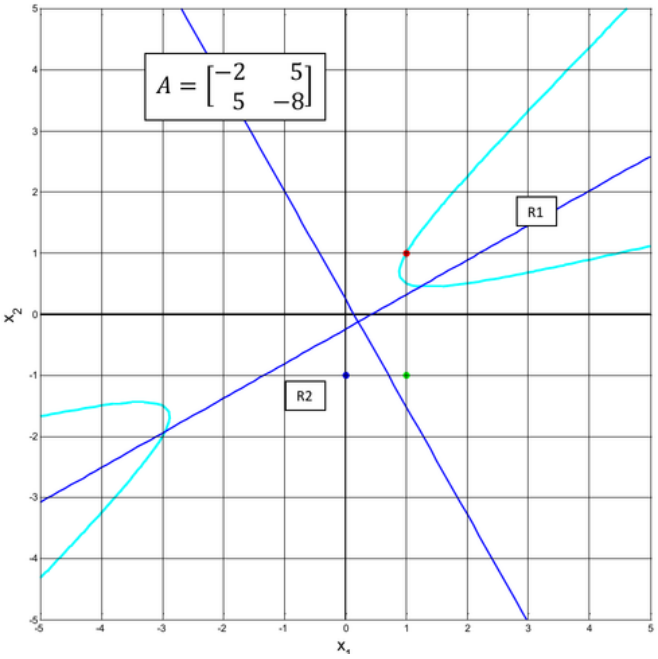
$$= -2x_1^2 + 5x_1x_2 + 5x_1x_2 - 8x_2^2 + x_1 + 2x_2 - 3$$

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

When $\mathbf{x} = (0, -1)^t$, $g(\mathbf{x}) = 0 - 8(-1)^2 + 0 + 0 + 2(-1) - 3 = -13$
$g(\mathbf{x}) \leq 0$ *so* $\mathbf{x}$ *is in class 2.*

When $\mathbf{x} = (1, 1)^t$, $g(\mathbf{x}) = -2 - 8 + 10 + 1 + 2 - 3 = 0$
$g(\mathbf{x}) = 0$ *so* $\mathbf{x}$ *is in class 2.*

The decision surface looks like this:



5. In augmented feature space, a dichotomizer is defined using the following linear discriminant function $g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$ **where** $\mathbf{a}^t = (-3, 1, 2, 2, 2, 4)$ **and** $\mathbf{y}^t = (1, x)$**. Determine the class of the following feature vectors,** $\mathbf{x}$**:** $\begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ **and** $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$**.**

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

$$(-3 \;\; 1 \;\; 2 \;\; 2 \;\; 2 \;\; 4) \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = -3 + 0 - 2 + 0 + 0 + 4 = -1$$

Therefore class 2.

$$(-3 \;\; 1 \;\; 2 \;\; 2 \;\; 2 \;\; 4) \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = -3 + 1 + 2 + 2 + 2 + 4 = 8$$

Therefore class 1.

Note, same as part (i) of previous question, just using generalised linear discriminant function where $\mathbf{y}^t = (1 \;\; x_1 \;\; x_2 \;\; x_1x_2 \;\; x_1^2 \;\; x_2^2)$.

6. A Linear Discriminant Function 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 appropriate parameters for the linear discriminant function, when the data set is as shown.

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

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

Using Augmented notation and sample normalisation, dataset is:

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

For the Batch Perceptron Learning Algorithm, weights are updated such that: $\mathbf{a} \leftarrow \mathbf{a} + \eta \sum_{y \in \chi} \mathbf{y}$. *Here,* $\eta = 1$.

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

$\mathbf{y}$	$g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$	*misclassified (i.e., $g(\mathbf{x}) \leq 0$)?*
$(1,1,5)^t$	$(-25 \times 1) + (6 \times 1) + (3 \times 5) = -4$	yes
$(1,2,5)^t$	$(-25 \times 1) + (6 \times 2) + (3 \times 5) = 2$	no
$(-1,-4,-1)^t$	$(-25 \times -1) + (6 \times -4) + (3 \times -1) = -2$	yes
$(-1,-5,-1)^t$	$(-25 \times -1) + (6 \times -5) + (3 \times -1) = -8$	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: $\mathbf{a} = (-26, -2, 6)^t$

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

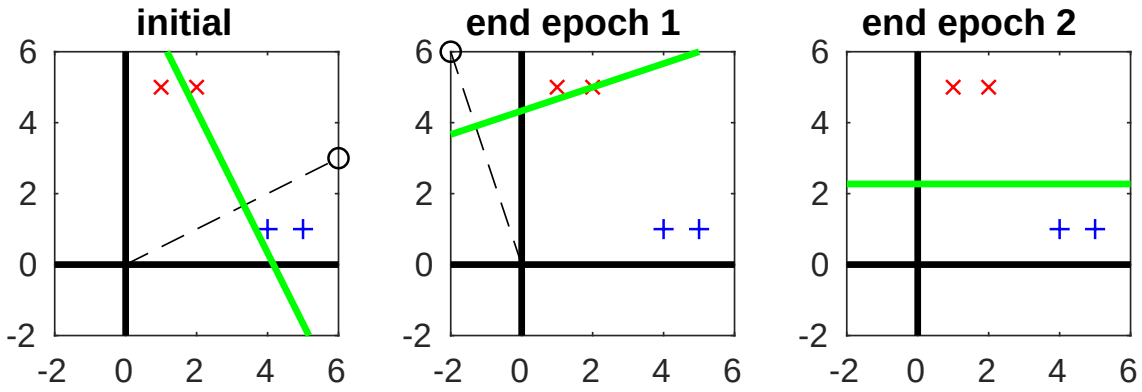
$\mathbf{a} \leftarrow (-26, -2, 6)^t + (1, 2, 5)^t = (-25, 0, 11)^t$

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

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



7. Repeat the previous question using the Sequential Perceptron Learning Algorithm (with augmented notation and sample normalisation).

Using Augmented notation and sample normalisation, dataset is:

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

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 $\mathbf{a} = (-25, 6, 3)^t$

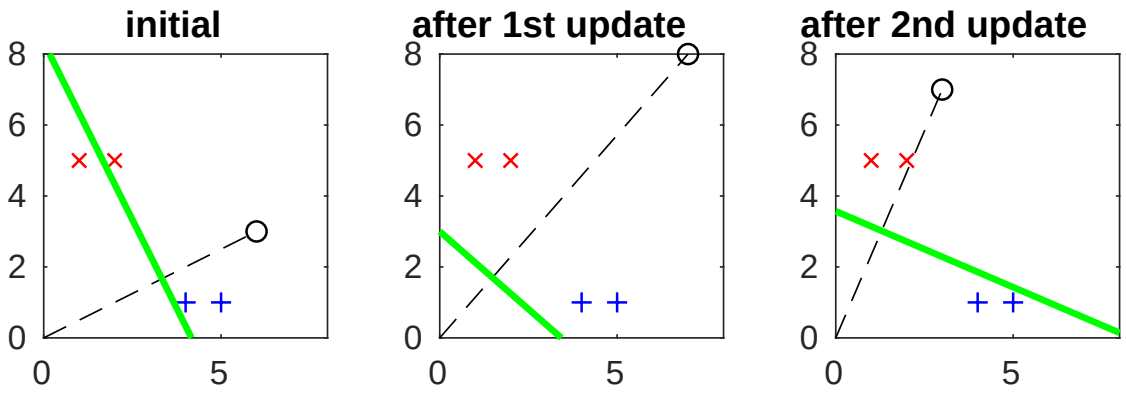
$\mathbf{y}^t$	$g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$	*updated* $\mathbf{a}^t$
$(1,1,5)$	$(-25 \times 1) + (6 \times 1) + (3 \times 5) = -4$	$(-25,6,3) + (1,1,5) = (-24,7,8)$
$(1,2,5)$	$(-24 \times 1) + (7 \times 2) + (8 \times 5) = 30$	$(-24,7,8)$
$(-1,-4,-1)$	$(-24 \times -1) + (7 \times -4) + (8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

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:



8. Write pseudo-code for the sequential Perceptron Learning Algorithm

Augment and apply sample normalisation to the feature vectors.
Initialise $\mathbf{a}$ *and set the learning rate.*
For each sample, $\mathbf{y}_k$*, in the data-set:*
    Calculate $g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$
    If $\mathbf{y}_k$ *is misclassified (i.e., if* $g(\mathbf{x}) \leq 0$*)*
    Update solution such that: $\mathbf{a} \leftarrow \mathbf{a} + \eta\mathbf{y}_k$
Repeat until $\mathbf{a}$ *unchanged by all samples (i.e. all samples correctly classified).*

OR
Augment the feature vectors.
Set $\omega_k = 1$ *for samples in class 1, and* $\omega_k = -1$ *for samples in class 2.*
Initialise $\mathbf{a}$ *and set the learning rate.*
For each sample, $\mathbf{y}_k$*, in the data-set:*
    Calculate $g(\mathbf{x}) = \mathbf{a}^T\mathbf{y}$
    If $\mathbf{y}_k$ *is misclassified (i.e., if* $sign(g(\mathbf{x})) \neq \omega_k$*)*
    Update solution such that: $\mathbf{a} \leftarrow \mathbf{a} + \eta\omega_k\mathbf{y}_k$
Repeat until $\mathbf{a}$ *unchanged by all samples (i.e. all samples correctly classified).*

9. Consider the following linearly separable data set.

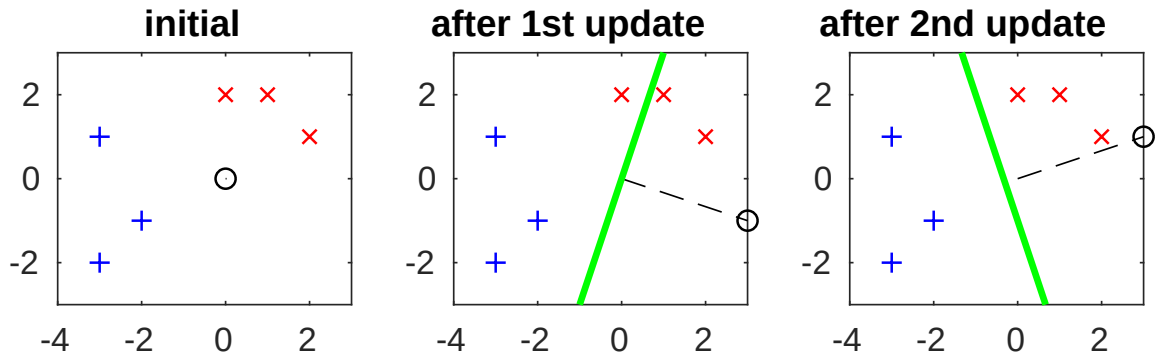
$\mathbf{x}$	$class$
$(0,2)^t$	1
$(1,2)^t$	1
$(2,1)^t$	1
$(-3,1)^t$	-1
$(-2,-1)^t$	-1
$(-3,-2)^t$	-1

Apply the Sequential Perceptron Learning Algorithm to determine the parameters of a linear discriminant function that will correctly classify this data. Assume an initial values of $\mathbf{a} = (w_0, \mathbf{w}^t)^t = (1,0,0)^t$**, and use a learning rate of 1.**

For the Sequential Perceptron Learning Algorithm, weights are updated such that: $\mathbf{a} \leftarrow \mathbf{a} + \eta\omega\mathbf{y}_k$*, where* $\mathbf{y}_k$ *is a misclassified exemplar, and* ω *is the class label associated with* $\mathbf{y}_k$*. Here,* $\eta = 1$*.*

iteration	$\mathbf{a}^t_{old}$	$\mathbf{y}^t$	$g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$	ω	$\mathbf{a}^t_{new} = \mathbf{a}^t_{old} + \omega\mathbf{y}^t$ *if misclassified*
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:*



10. Repeat previous question using the sample normalisation method of implementing the Sequential Perceptron Learning Algorithm.

Using Augmented notation and sample normalisation, dataset is:

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

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,* $\eta = 1$.

iteration	$\mathbf{a}_{old}^t$	$\mathbf{y}^t$	$g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$	$\mathbf{a}_{new}^t = \mathbf{a}_{old}^t + \omega\mathbf{y}^t$ *if misclassified*
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 $\mathbf{a} = (1,3,1)^t$.

11. A data-set consists of exemplars from three classes.
Class 1: $\begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}2\\0\end{pmatrix}$. **Class 2:** $\begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}-1\\1\end{pmatrix}$. **Class 3:** $\begin{pmatrix}-1\\-1\end{pmatrix}$. **Use the Sequential Multiclass Perceptron Learning algorithm to find the parameters for three linear discriminant functions that will correctly classify this data. Assume initial values for all parameters are zero, and use a learning rate of 1. If more than one discriminant function produces the maximum output, choose the function with the highest index (i.e., the one that represents the largest class label).**

Sequential Multiclass Perceptron Learning algorithm:

- *Initialise* $\mathbf{a}_j$ *for each class.*
- *For each exemplar* $(\mathbf{y}_k, \omega_k)$ *in turn*
  - *Find predicted class* $j = \arg\max_{j'}(\mathbf{a}_{j'}^T\mathbf{y}_k)$
  - *If predicted class is not true class (i.e.,* $j \neq \omega_k$*), update weights:*
    $\mathbf{a}_{\omega_k} = \mathbf{a}_{\omega_k} + \eta\mathbf{y}_k$
    $\mathbf{a}_j = \mathbf{a}_j - \eta\mathbf{y}_k$
- *repeat until weights stop changing*

Using Augmented notation dataset is:

$\mathbf{y}^t$	ω
$(1,1,1)$	1
$(1,2,0)$	1
$(1,0,2)$	2
$(1,-1,1)$	2
$(1,-1,-1)$	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 $\mathbf{a}_1 = (1, 2, 0)^t, \mathbf{a}_2 = (0, -1, 1)^t, \mathbf{a}_3 = (-1, -1, -1)^t$.

12. Consider the following linearly separable data set.

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

Use the pseudoinverse (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 $\mathbf{b} = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]^t$**.**

Using Augmented notation and sample normalisation, dataset is:

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

$\mathbf{Ya} = \mathbf{b}$ *where* $\mathbf{Y} = \begin{pmatrix} 1 & 0 & 2 \\ 1 & 1 & 2 \\ 1 & 2 & 1 \\ -1 & 3 & -1 \\ -1 & 2 & 1 \\ -1 & 3 & 2 \end{pmatrix}$

Find the pseudo-inverse of $\mathbf{Y}$ *(*$\mathbf{Y}^\dagger = (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t$*) using MATLAB command pinv:*

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

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

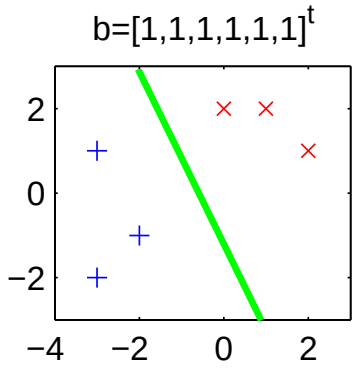
$$\mathbf{a} = \begin{pmatrix} 0.2273 \\ 0.3864 \\ 0.1894 \end{pmatrix}$$

Check:

$\mathbf{y}^t$	$g(\mathbf{x}) = \mathbf{a}^t\mathbf{y} = (0.2273, 0.3864, 0.1894)\mathbf{y}$
$(1, 0, 2)$	0.6061
$(1, 1, 2)$	0.9924
$(1, 2, 1)$	1.1894
$(-1, 3, -1)$	0.7424
$(-1, 2, 1)$	0.7348
$(-1, 3, 2)$	1.3106

All positive, so discriminant function provides correct classification.

In feature space, the decision surface looks like this:

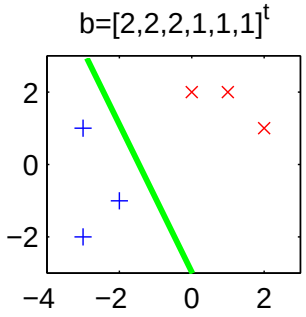


13. Repeat the previous question using (a) $\mathbf{b} = [2, 2, 2, 1, 1, 1]^t$**, and (b)** $\mathbf{b} = [1, 1, 1, 2, 2, 2]^t$**.**

a) $\mathbf{a} = \mathbf{Y}^\dagger \mathbf{b} = \begin{pmatrix} 0.0682 & 0.1648 & 0.3807 & 0.1023 & -0.2330 & -0.2557 \\ -0.0341 & 0.0426 & 0.1847 & 0.1989 & -0.0085 & 0.0028 \\ 0.1402 & 0.0748 & -0.1203 & -0.2064 & 0.1184 & 0.1828 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix}$

$$\mathbf{a} = \begin{pmatrix} 0.8409 \\ 0.5795 \\ 0.2841 \end{pmatrix}$$

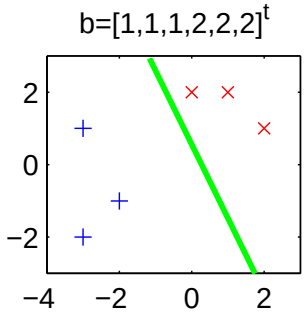
In feature space, the decision surface looks like this:



b) $\mathbf{a} = \mathbf{Y}^{\dagger}\mathbf{b} = \begin{pmatrix} 0.0682 & 0.1648 & 0.3807 & 0.1023 & -0.2330 & -0.2557 \\ -0.0341 & 0.0426 & 0.1847 & 0.1989 & -0.0085 & 0.0028 \\ 0.1402 & 0.0748 & -0.1203 & -0.2064 & 0.1184 & 0.1828 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \end{pmatrix}$

$$\mathbf{a} = \begin{pmatrix} -0.1591 \\ 0.5795 \\ 0.2841 \end{pmatrix}$$

In feature space, the decision surface looks like this:



Each element of $\mathbf{b}$ *corresponds to a data point. Increasing the value of* $\mathbf{b}$ *increases the margin (i.e. distance) of the hyperplane from the corresponding data point.*

14. For the same dataset used in the preceding question, apply 12 iterations of the Sequential Widrow-Hoff Learning Algorithm. Assume an initial values of $\mathbf{a} = (w_0, \mathbf{w}^t)^t = (1, 0, 0)^t$ **, use a margin vector** $\mathbf{b} = [111111]^t$**, and a learning rate of 0.1.**

Using Augmented notation and sample normalisation, dataset is:

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

For the Sequential Widrow-Hoff Learning Algorithm, 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	$\mathbf{a}^t$	$\mathbf{y}_k^t$	$\mathbf{a}^t\mathbf{y}_k$	$\mathbf{a}_{new}^t = \mathbf{a}^t + 0.1(b_k - \mathbf{a}^t\mathbf{y}_k)\mathbf{y}_k^t$
1	$[1, 0, 0]$	$[1, 0, 2]$	1	$[1, 0, 0] + 0.1(1\text{-}1)[1, 0, 2] = [1, 0, 0]$
2	$[1, 0, 0]$	$[1, 1, 2]$	1	$[1, 0, 0] + 0.1(1\text{-}1)[1, 1, 2] = [1, 0, 0]$
3	$[1, 0, 0]$	$[1, 2, 1]$	1	$[1, 0, 0] + 0.1(1\text{-}1)[1, 2, 1] = [1, 0, 0]$
4	$[1, 0, 0]$	[-1, 3, -1]	-1	$[1, 0, 0] + 0.1(1\text{-}(-1))[\text{-}1, 3, \text{-}1] = [0.8, 0.6, \text{-}0.2]$
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]

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

Class	feature vector
1	**(0.15,0.35)**
2	**(0.15,0.28)**
2	**(0.12,0.2)**
3	**(0.1,0.32)**
3	**(0.06,0.25)**

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

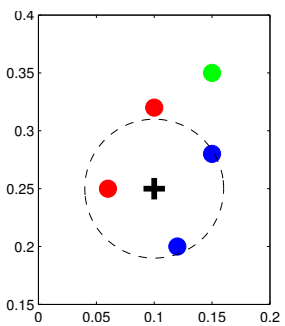
Calculate distance of each sample to x:

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	*(0.12,0.2)*	$\sqrt{(0.1-0.12)^2 + (0.25-0.2)^2} = 0.0539$
3	*(0.1,0.32)*	$\sqrt{(0.1-0.1)^2 + (0.25-0.32)^2} = 0.0700$
3	*(0.06,0.25)*	$\sqrt{(0.1-0.06)^2 + (0.25-0.25)^2} = 0.0400$

a) The nearest neighbour has class label 3. Therefore class new sample as 3.

b) The three nearest neighbours have class labels 3, 2, and 2. Therefore class new sample as 2.

Note, the feature space looks like this:



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

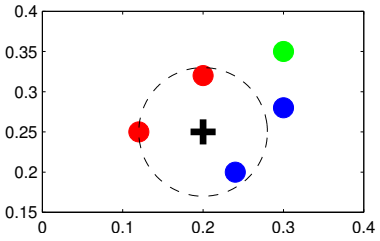
Class	*feature vector*	*Euclidean distance to (0.2,0.25)*
1	*(0.3,0.35)*	$\sqrt{(0.2-0.3)^2+(0.25-0.35)^2} = 0.1414$
2	*(0.3,0.28)*	$\sqrt{(0.2-0.3)^2+(0.25-0.28)^2} = 0.1044$
2	*(0.24,0.2)*	$\sqrt{(0.2-0.24)^2+(0.25-0.2)^2} = 0.0640$
3	*(0.2,0.32)*	$\sqrt{(0.2-0.2)^2+(0.25-0.32)^2} = 0.0700$
3	*(0.12,0.25)*	$\sqrt{(0.2-0.12)^2+(0.25-0.25)^2} = 0.0800$

For k=1, class of new sample is 2

For k=3, class of new sample is 3

Note, this is the opposite of the result we got previously. kNN is very sensitive to the scale of the feature dimensions!

The feature space now looks like this:

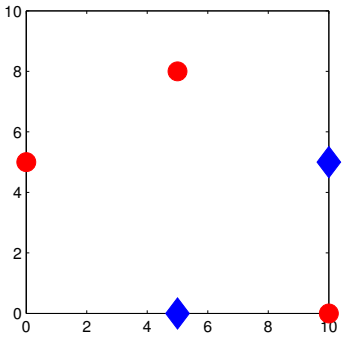


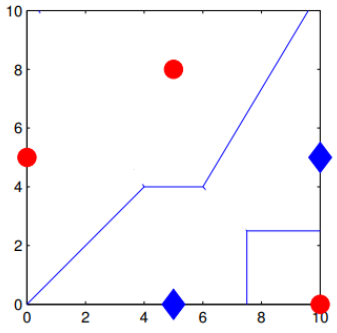
16. a) Plot the decision boundaries that result from applying a nearest neighbour classifier (i.e. a kNN classifier with k=1), to the data shown in the table. Assume Euclidean distance is used to define distance.

Class	x_1	x_2
1	**0**	**5**
	5	**8**
	10	**0**
2	**5**	**0**
	10	**5**

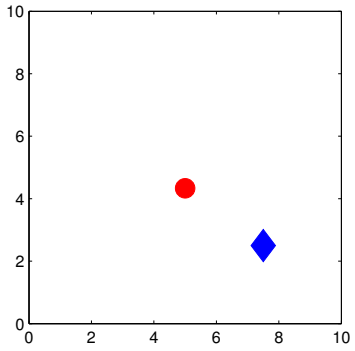
b) For the same data, plot the decision boundaries that result from applying a nearest mean classifier (i.e. one in which a new feature vector, x, is classified by assigning it to the same category as nearest sample mean).

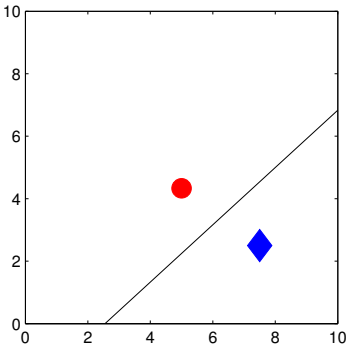
a)





b)





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	$H(-1.5\times1+2\times0) = H(-1.5) = 0$	1	$[1,0]$	$[-0.5,2]$
$(1,1)$	0	$H(-0.5\times1+2\times1) = H(1.5) = 1$	-1	$[-1,-1]$	$[-1.5,1]$
$(1,0)$	1	$H(-1.5\times1+1\times0) = H(-1.5) = 0$	1	$[1,0]$	$[-0.5,1]$
$(1,1)$	0	$H(-0.5\times1+1\times1) = H(0.5) = 1$	-1	$[-1,-1]$	$[-1.5,0]$
$(1,0)$	1	$H(-1.5\times1+0\times0) = H(-1.5) = 0$	1	$[1,0]$	$[-0.5,0]$
$(1,1)$	0	$H(-0.5\times1+0\times1) = H(-0.5) = 0$	0	$[0,0]$	$[-0.5,0]$
$(1,0)$	1	$H(-0.5\times1+0\times0) = H(-0.5) = 0$	1	$[1,0]$	$[0.5,0]$
$(1,1)$	0	$H(0.5\times1+0\times1) = H(0.5) = 1$	-1	$[-1,-1]$	$[-0.5,-1]$
$(1,0)$	1	$H(-0.5\times1-1\times0) = H(-0.5) = 0$	1	$[1,0]$	$[0.5,-1]$
$(1,1)$	0	$H(0.5\times1+-1\times1) = H(-0.5) = 0$	0	$[0,0]$	$[0.5,-1]$
$(1,0)$	1	$H(0.5\times1-1\times0) = H(0.5) = 1$	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	$H(-1.5\times1+2\times0) = H(-1.5) = 0$	1	$[1,0]$	
$(1,1)$	0	$H(-1.5\times1+2\times1) = H(0.5) = 1$	-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	$H(-1.5\times1+1\times0) = H(-1.5) = 0$	1	$[1,0]$	
$(1,1)$	0	$H(-1.5\times1+1\times1) = H(-0.5) = 0$	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	$H(-0.5\times1+1\times0) = H(-0.5) = 0$	1	$[1,0]$	
$(1,1)$	0	$H(-0.5\times1+1\times1) = H(0.5) = 1$	-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	$H(-0.5\times1+0\times0) = H(-0.5) = 0$	1	$[1,0]$	
$(1,1)$	0	$H(-0.5\times1+0\times1) = H(-0.5) = 0$	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* $\mathbf{x} = [1, x_1, x_2]^t$*.*

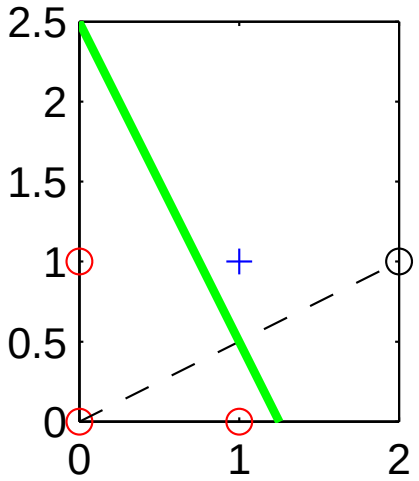
For the Delta rule, weights are updated such that: $\mathbf{w} \leftarrow \mathbf{w} + \eta(t-y)\mathbf{x}^t$

Epoch 1, 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	$H(0.5\times 1+1\times 0+1\times 0)=1$	-1	$[-1,0,0]$	$[-0.5,1,1]$
$(1,0,1)$	0	$H(-0.5\times 1+1\times 0+1\times 1)=H(0.5)=1$	-1	$[-1,0,-1]$	$[-1.5,1,0]$
$(1,1,0)$	0	$H(-1.5\times 1+1\times 1+0\times 0)=H(-0.5)=0$	0	$[0,0,0]$	$[-1.5,1,0]$
$(1,1,1)$	1	$H(-1.5\times 1+1\times 1+0\times 1)=H(-0.5)=0$	1	$[1,1,1]$	$[-0.5,2,1]$
$(1,0,0)$	0	$H(-0.5\times 1+2\times 0+1\times 0)=H(-0.5)=0$	0	$[0,0,0]$	$[-0.5,2,1]$
$(1,0,1)$	0	$H(-0.5\times 1+2\times 0+1\times 1)=H(0.5)=1$	-1	$[-1,0,-1]$	$[-1.5,2,0]$
$(1,1,0)$	0	$H(-1.5\times 1+2\times 1+0\times 0)=H(0.5)=1$	-1	$[-1,-1,0]$	$[-2.5,1,0]$
$(1,1,1)$	1	$H(-2.5\times 1+1\times 1+0\times 1)=H(-1.5)=0$	1	$[1,1,1]$	$[-1.5,2,1]$
$(1,0,0)$	0	$H(-1.5\times 1+2\times 0+1\times 0)=H(-1.5)=0$	0	$[0,0,0]$	$[-1.5,2,1]$
$(1,0,1)$	0	$H(-1.5\times 1+2\times 0+1\times 1)=H(-0.5)=0$	0	$[0,0,0]$	$[-1.5,2,1]$
$(1,1,0)$	0	$H(-1.5\times 1+2\times 1+1\times 0)=H(0.5)=1$	-1	$[-1,-1,0]$	$[-2.5,1,1]$
$(1,1,1)$	1	$H(-2.5\times 1+1\times 1+1\times 1)=H(-0.5)=0$	1	$[1,1,1]$	$[-1.5,2,2]$
$(1,0,0)$	0	$H(-1.5\times 1+2\times 0+2\times 0)=H(-1.5)=0$	0	$[0,0,0]$	$[-1.5,2,2]$
$(1,0,1)$	0	$H(-1.5\times 1+2\times 0+2\times 1)=H(0.5)=1$	-1	$[-1,0,-1]$	$[-2.5,2,1]$
$(1,1,0)$	0	$H(-2.5\times 1+2\times 1+1\times 0)=H(-0.5)=0$	0	$[0,0,0]$	$[-2.5,2,1]$
$(1,1,1)$	1	$H(-2.5\times 1+2\times 1+1\times 1)=H(0.5)=1$	0	$[0,0,0]$	$[-2.5,2,1]$
$(1,0,0)$	0	$H(-2.5\times 1+2\times 0+1\times 0)=H(-2.5)=0$	0	$[0,0,0]$	$[-2.5,2,1]$
$(1,0,1)$	0	$H(-2.5\times 1+2\times 0+1\times 1)=H(-1.5)=0$	0	$[0,0,0]$	$[-2.5,2,1]$
$(1,1,0)$	0	$H(-2.5\times 1+2\times 1+1\times 0)=H(-0.5)=0$	0	$[0,0,0]$	$[-2.5,2,1]$
$(1,1,1)$	1	$H(-2.5\times 1+2\times 1+1\times 1)=H(0.5)=1$	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	$\mathbf{e}^T$	$(\mathbf{We})^T$	$\mathbf{y}^T$	$(\mathbf{W}^T\mathbf{y})^T$
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.**

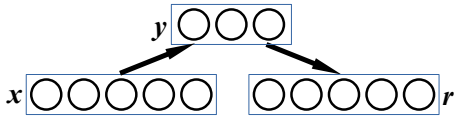
$$\tilde{\mathbf{W}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$

iteration	$\mathbf{e}^T$	$(\tilde{\mathbf{W}}\mathbf{e})^T$	$\mathbf{y}^T$	$(\mathbf{W}^T\mathbf{y})^T$
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	$(1.0825, 1.0825, 0)$	$(1.0825, 0.72165)$	$(0.83505, 0.10997)$	$(0.94502, 0.94502, 0.10997)$

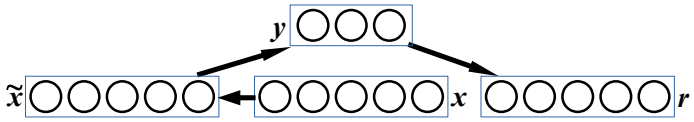
So output is $\begin{pmatrix} 0.83505 \\ 0.10997 \end{pmatrix}$

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 avoids overfitting.*

4 Deep Discriminative Neural Networks

1. Give brief definitions of the following terms:

a. Feature map

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.

b. Sub-sampling

The process of reducing the width and height of a feature map by pooling (also the process of reducing the resolution of an image).

c. Adversarial example

A sample that has been manipulated so that it is classified differently to the original sample.

d. Data augmentation

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.

e. Transfer learning

A method of training a classifier that utilises the results of pre-training the classifier using another dataset.

f. Dropout

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.

g. Regularization

Any method intended to reduce over-fitting and improve generalisation.

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.

When errors, backpropagated through a neural network, become smaller and smaller resulting in smaller and smaller weight updates. It is due to the error being multiplied at each layer of the network by derivatives or weights that are small. It can result in little or no learning occurring in the earlier layers of the network, and hence, causing these layer to fail to contribute to solving the task.

Methods that reduce the likelihood of gradients vanishing are:

- *using activation functions, like ReLU, which have a smaller range for which the derivative is* < 1.
- *initialising the weights in ways that have been found, empirically, to reduce the effects of vanishing gradients.*
- *using variations of standard backpropagation that use momentum and/or an adaptive learning rate.*
- *using batch normalisation to keep the mean and standard deviation of each neuron close to 0 and 1 respectively.*

- *using skip connections so the gradient can by-pass layers in the network where the gradient has vanished.*

3. Mathematically define the following activation functions:

a. ReLU

$\varphi(net_j) = net_j$ *if* $net_j \geq 0$ *or* 0 *if* $net_j < 0$

b. 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.*

c. 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.*

4. The following array show the output produced by a mask in a convolutional layer of a CNN.

$$net_j = \begin{bmatrix} 1 & 0.5 & 0.2 \\ -1 & -0.5 & -0.2 \\ 0.1 & -0.1 & 0 \end{bmatrix}$$

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

a. ReLU,

b. LReLU when a=0.1,

c. tanh,

d. Heaviside function where each neuron has a threshold of 0.1 (define H(0) as 0.5).

a) ReLU:

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

b) LReLU when a=0.1:

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

c) tanh:

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

d) Heaviside function where each neuron has a threshold of 0.1:

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

5. The following arrays show the output produced by a convolutional layer to all 4 samples in a batch. $X_1 = \begin{bmatrix} 1 & 0.5 & 0.2 \\ -1 & -0.5 & -0.2 \\ 0.1 & -0.1 & 0 \end{bmatrix}$, $X_2 = \begin{bmatrix} 1 & -1 & 0.1 \\ 0.5 & -0.5 & -0.1 \\ 0.2 & -0.2 & 0 \end{bmatrix}$, $X_3 = \begin{bmatrix} 0.5 & -0.5 & -0.1 \\ 0 & -0.4 & 0 \\ 0.5 & 0.5 & 0.2 \end{bmatrix}$, $X_4 = \begin{bmatrix} 0.2 & 1 & -0.2 \\ -1 & -0.6 & -0.1 \\ 0.1 & 0 & 0.1 \end{bmatrix}$.
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.**

Batch normalisation modifies the output of an individual neuron, x*, to become:*

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

Where $E(x)$ *is the mean, and* $Var(x)$ *is the variance, of* x *within the batch.*

For the top-left neuron:
$E(x) = (1 + 1 + 0.5 + 0.2)/4 = 0.675$,
$Var(x) = \frac{(1-0.675)^2+(1-0.675)^2+(0.5-0.675)^2+(0.2-0.675)^2}{4} = 0.1169$
$BN(x_1) = 0 + 1 \times \frac{1-0.675}{\sqrt{0.1169+0.1}} = 0.6979$
$BN(x_2) = 0 + 1 \times \frac{1-0.675}{\sqrt{0.1169+0.1}} = 0.6979$
$BN(x_3) = 0 + 1 \times \frac{0.5-0.675}{\sqrt{0.1169+0.1}} = -0.3758$
$BN(x_4) = 0 + 1 \times \frac{0.2-0.675}{\sqrt{0.1169+0.1}} = -1.0200$

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

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

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

Note, 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.

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

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

If a mask, H, has two channels defined as:

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

Calculate the output produced by mask H when using:

a. padding=0, stride=1

b. padding=1, stride=1

c. padding=1, stride=2

d. padding=0, stride=1, dilation=2

a) padding=0, stride=1:

$$X_1 \star H_1 + X_2 \star H_2 =$$

$$\begin{bmatrix} (0.2 \times 1) + (1 \times \text{-}0.1) + (\text{-}1 \times 1) + (0 \times \text{-}0.1) & (1 \times 1) + (0 \times \text{-}0.1) + (0 \times 1) + (\text{-}0.1 \times \text{-}0.1) \\ (\text{-}1 \times 1) + (0 \times \text{-}0.1) + (0.1 \times 1) + (0 \times \text{-}0.1) & (0 \times 1) + (\text{-}0.1 \times \text{-}0.1) + (0 \times 1) + (0.1 \times \text{-}0.1) \end{bmatrix}$$
$$+ \begin{bmatrix} (1 \times 0.5) + (0.5 \times 0.5) + (\text{-}1 \times \text{-}0.5) + (\text{-}0.5 \times \text{-}0.5) & (0.5 \times 0.5) + (0.2 \times 0.5) + (\text{-}0.5 \times \text{-}0.5) + (\text{-}0.2 \times \text{-}0.5) \\ (\text{-}1 \times 0.5) + (\text{-}0.5 \times 0.5) + (0.1 \times \text{-}0.5) + (\text{-}0.1 \times \text{-}0.5) & (\text{-}0.5 \times 0.5) + (\text{-}0.2 \times 0.5) + (\text{-}0.1 \times \text{-}0.5) + (0 \times \text{-}0.5) \end{bmatrix}$$
$$= \begin{bmatrix} -0.9 & 1.01 \\ -0.9 & 0 \end{bmatrix} + \begin{bmatrix} 1.5 & 0.7 \\ -0.75 & -0.3 \end{bmatrix} = \begin{bmatrix} 0.6 & 1.71 \\ -1.65 & -0.3 \end{bmatrix}$$

b) padding=1, stride=1:

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

$$X_1' \star H_1 + X_2' \star H_2 =$$

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

c) padding=1, stride=2:

using answer to previous part:

$$\begin{bmatrix} -0.52 & 0.65 \\ -0.46 & -0.3 \end{bmatrix}$$

d) padding=0, stride=1, dilation=2:

$$H_1' = \begin{bmatrix} 1 & 0 & -0.1 \\ 0 & 0 & 0 \\ 1 & 0 & -0.1 \end{bmatrix}, \; H_2' = \begin{bmatrix} 0.5 & 0 & 0.5 \\ 0 & 0 & 0 \\ -0.5 & 0 & -0.5 \end{bmatrix}$$

$$X_1 \star H_1' + X_2 \star H_2'$$
$$= \begin{bmatrix} 0.29 \end{bmatrix} + \begin{bmatrix} 0.55 \end{bmatrix} = \begin{bmatrix} 0.84 \end{bmatrix}$$

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

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

Calculate the output produced by 1x1 convolution when the 3 channels of the 1x1 mask are: $[1, -1, 0.5]$**.**

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

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

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 when using:

a. **average pooling with a pooling region of 2x2 and stride=2**
b. **max pooling with a pooling region of 2x2 and stride=2**
c. **max pooling with a pooling region of 3x3 and stride=1**

a) average pooling with a pooling region of 2x2 and stride=2:

$$\begin{bmatrix} (0.2 + 1 - 1 + 0)/4 & (0 + 0.4 - 0.1 - 0.1)/4 \\ (0.1 + 0 + 0.4 - 0.7)/4 & (-1 - 0.5 - 0.5 + 1)/4 \end{bmatrix} = \begin{bmatrix} 0.05 & 0.05 \\ -0.05 & -0.25 \end{bmatrix}$$

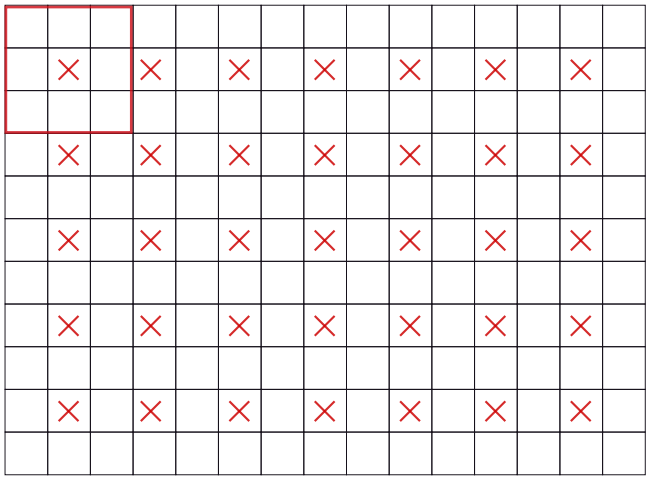
b) max pooling with a pooling region of 2x2 and stride=2:

$$\begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}$$

c) max pooling with a pooling region of 3x3 and stride=1:

$$\begin{bmatrix} 1 & 1 \\ 0.4 & 1 \end{bmatrix}$$

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 will the output produced by a single mask with 6 channels and 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.



output size will be: $5 \times 7 \times 1$

Note, in general $\text{outputDim} = 1 + \frac{(\text{inputDim} - \text{maskDim} + 2 \times \text{padding})}{\text{stride}}$.
So for this example:

$\text{outputHeight} = 1 + \frac{(11-3+2\times 0)}{2} = 1 + 4 = 5.$

$\text{outputWidth} = 1 + \frac{(15-3+2\times 0)}{2} = 1 + 6 = 7.$

10. A CNN processes an image of size 200x200x3 using the following sequence of layers:

- **convolution with 40 masks of size 5x5x3 with stride=1, padding=0**
- **pooling with 2x2 pooling regions stride=2**
- **convolution with 80 masks of size 4x4 with stride=2, padding=1**
- **1x1 convolution with 20 masks**

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

Using: $\text{outputDim} = 1 + \frac{(\text{inputDim} - \text{maskDim} + 2 \times \text{padding})}{\text{stride}}$.

- *convolution with 40 masks of size 5x5x3 with stride=1, padding=0:*
  width=height=1+(200-5)/1=196
  number of channels=40 (as there are 40 masks)
  so size of output is 196x196x40
- *pooling with 2x2 pooling regions stride=2:*
  width=height=1+(196-2)/2=98
  number of channels=40 (as pooling does not change this)
  so size of output is 98x98x40
- *convolution with 80 masks of size 4x4 with stride=2, padding=1:*
  width=height=1+(98-4+2)/2=49
  number of channels=80 (as there are 80 masks)
  so size of output is 49x49x80
- *1x1 convolution with 20 masks:*
  width=height=49 (as 1x1 convolution does not change this)
  number of channels=20 (as there are 20 masks)
  so size of output is 49x49x20
- *after flatterning, length of feature vector is: 48020*

11. The following images show exemplars from two datasets:

MNIST

Fashion MNIST

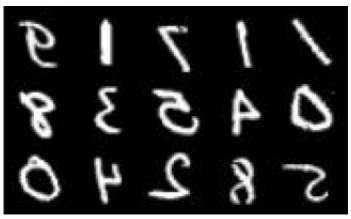
Each dataset is to be expanded using data augmentation. Which of the following transformations are appropriate:

a. rescaling

MNIST: yes, FashionMNIST: yes

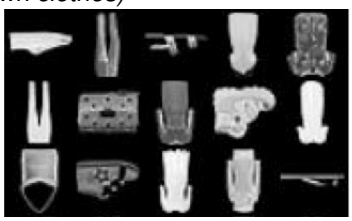
b. horizontal flip

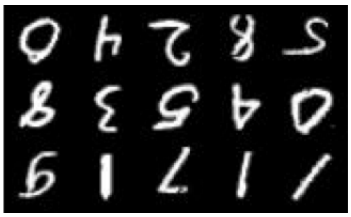
MNIST: no, FashionMNIST: yes



c. rotation

MNIST: no (small rotations would be OK), FashionMNIST: yes (if we want classifier to recognise up-side-down clothes)





d. cropping

MNIST: yes, FashionMNIST: yes
Assuming object still recognisable after crop

5 Feature Extraction

1. Briefly define The following terms:

a. Feature Engineering

modifying measured values to make them suitable for classification.

b. Feature Selection

choosing a subset of measured/possible features to use for classification.

c. Feature Extraction

projecting the chosen feature vectors into a new feature space.

d. Dimensionality Reduction

selecting/extracting feature vectors of lower dimensionality (length) than the original feature vectors.

e. 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.

2. List 5 methods that can be used to perform feature extraction.

Any five from:

- *Principal Component Analysis (PCA)*
- *Whitening*
- *Linear Discriminant Analysis (LDA)*
- *Independent Component Analysis (ICA)*
- *Random Projections*
- *Sparse Coding*

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

1. *Subtract the mean from all data vectors.*
2. *Calculate the covariance matrix of the zero-mean data.*
3. *Find the eigenvalues and eigenvectors of the covariance matrix.*
4. *Order eigenvalues from large to small, and discard small eigenvalues and their respective vectors. Form a matrix ($\hat{V}$) of the remaining eigenvectors.*
5. *Project the zero-mean data onto the PCA subspace by multiplying by the transpose of $\hat{V}$.*

4. Use the Karhunen-Loève Transform to project the following 3-dimensional data onto the first two principal components (the MATLAB command eig can be used to find eigenvectors and eignevalues).

$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix} 3 \\ 5 \\ 1 \end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}.$

The mean of the data is $\mu = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}$*, hence, the zero-mean data is:*

$\mathbf{x}'_1 = \begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix}, \mathbf{x}'_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \mathbf{x}'_3 = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \mathbf{x}'_4 = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}.$

The covariance matrix is

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$$

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

$$\mathbf{C} = \frac{1}{4}\left[\begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix}\begin{pmatrix} -1 & -1 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}\begin{pmatrix} 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}\begin{pmatrix} 1 & 2 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 & -1 & 0 \end{pmatrix}\right]$$

$$\mathbf{C} = \frac{1}{4}\left[\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 2 & 0 \\ 2 & 4 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}\right]$$

$$\mathbf{C} = \frac{1}{4}\begin{pmatrix} 2 & 3 & 0 \\ 3 & 6 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Calculating eigenvectors and eignevalues of $\mathbf{C}$ *(using MATLAB command eig):*

$$\mathbf{V} = \begin{pmatrix} 0 & -0.8817 & 0.4719 \\ 0 & 0.4719 & 0.8817 \\ 1 & 0 & 0 \end{pmatrix} \qquad \mathbf{E} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0.0986 & 0 \\ 0 & 0 & 1.9014 \end{pmatrix}$$

We need to choose the two eigenvectors corresponding to the two largest eigenvectors:

$$\hat{\mathbf{V}} = \begin{pmatrix} 0.4719 & -0.8817 \\ 0.8817 & 0.4719 \\ 0 & 0 \end{pmatrix}$$

Projection of the data onto the subspace spanned by the 1st two principal components is given by: $\mathbf{y}_i = \hat{\mathbf{V}}^T(\mathbf{x}_i - \mu)$

Hence, $\mathbf{y}_i = \begin{pmatrix} 0.4719 & 0.8817 & 0 \\ -0.8817 & 0.4719 & 0 \end{pmatrix}\mathbf{x}'_i$

Therefore,

$$\mathbf{y}_1 = \begin{pmatrix} 0.4719 & 0.8817 & 0 \\ -0.8817 & 0.4719 & 0 \end{pmatrix}\begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1.3536 \\ 0.4098 \end{pmatrix}$$

$$\mathbf{y}_2 = \begin{pmatrix} 0.4719 & 0.8817 & 0 \\ -0.8817 & 0.4719 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\mathbf{y}_3 = \begin{pmatrix} 0.4719 & 0.8817 & 0 \\ -0.8817 & 0.4719 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 2.2353 \\ 0.0621 \end{pmatrix}$$

$$\mathbf{y}_4 = \begin{pmatrix} 0.4719 & 0.8817 & 0 \\ -0.8817 & 0.4719 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.8817 \\ -0.4719 \end{pmatrix}$$

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).

5. What is the proportion of the variance explained by the 1st two principal components in the preceding question?

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 1st 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 1st PC without losing too much information.

6. Use the Karhunen-Loève Transform to project the following 2-dimensional dataset onto the first principal component (the MATLAB command eig can be used to find eigenvectors and eignevalues).
$\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \begin{pmatrix} 5 \\ 4 \end{pmatrix}, \begin{pmatrix} 5 \\ 6 \end{pmatrix}, \begin{pmatrix} 8 \\ 7 \end{pmatrix}, \begin{pmatrix} 9 \\ 7 \end{pmatrix}$.

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

$$\mathbf{C} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$$

$$\mathbf{C} = \frac{1}{6} \left[\begin{pmatrix} -5 \\ -4 \end{pmatrix} (-5 \quad -4) + \begin{pmatrix} -2 \\ 0 \end{pmatrix} (-2 \quad 0) + \begin{pmatrix} 0 \\ -1 \end{pmatrix} (0 \quad -1) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} (0 \quad 1) + \begin{pmatrix} 3 \\ 2 \end{pmatrix} (3 \quad 2) + \begin{pmatrix} 4 \\ 2 \end{pmatrix} (4 \quad 2) \right]$$

$$\mathbf{C} = \frac{1}{6}\left[\begin{pmatrix} 25 & 20 \\ 20 & 16 \end{pmatrix} + \begin{pmatrix} 4 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 9 & 6 \\ 6 & 4 \end{pmatrix} + \begin{pmatrix} 16 & 8 \\ 8 & 4 \end{pmatrix}\right]$$

$$\mathbf{C} = \frac{1}{6}\begin{pmatrix} 54 & 34 \\ 34 & 26 \end{pmatrix} = \begin{pmatrix} 9 & 5.67 \\ 5.67 & 4.33 \end{pmatrix}$$

Calculating eigenvectors and eignevalues of $\mathbf{C}$ *(using MATLAB command eig):*

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

We need to choose the eigenvector corresponding to the largest eigenvector:

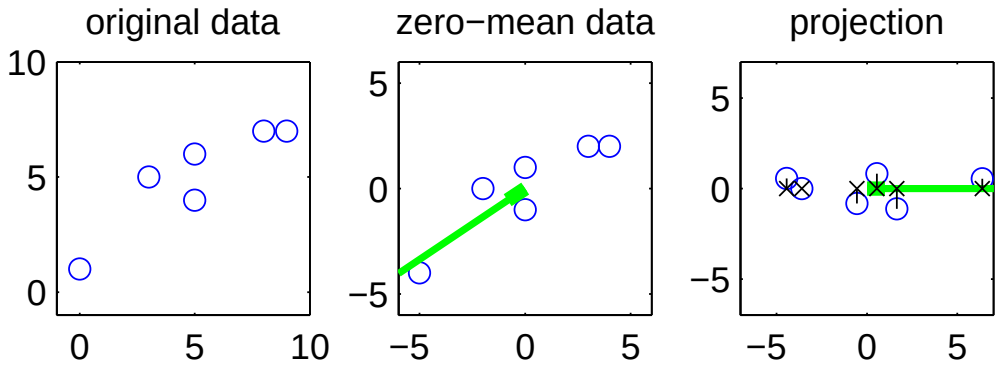
$$\hat{\mathbf{V}} = \begin{pmatrix} -0.8309 \\ -0.5564 \end{pmatrix}$$

Projection of the data onto the subspace spanned by the 1st principal component is given by: $\mathbf{y}_i = \hat{\mathbf{V}}^T(\mathbf{x}_i - \mu)$

Hence, $\mathbf{y}_i = \begin{pmatrix} -0.8309 & -0.5564 \end{pmatrix}\left[\mathbf{x}_i - \begin{pmatrix} 5 \\ 5 \end{pmatrix}\right]$

Therefore, the new dataset is: $6.3801, 1.6618, 0.5564, -0.5564, -3.6055, -4.4364.$

The projection looks like this:



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

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

Epoch 1, initial $\mathbf{w} = [-1, 0]$

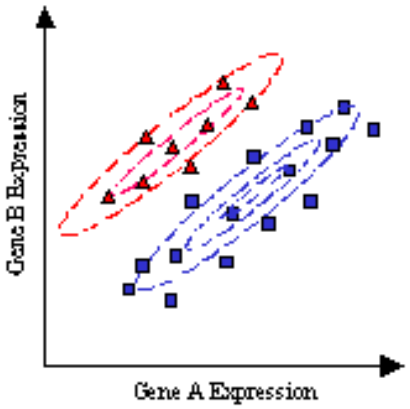
$\mathbf{x}^t$	$y = \mathbf{wx}$	$\mathbf{x}^t - y\mathbf{w}$	$\eta y(\mathbf{x}^t - y\mathbf{w})$	$\mathbf{w}$
$(-5, -4)$	5	$(0, -4)$	$(0, -0.2)$	
$(-2, 0)$	2	$(0, 0)$	$(0, 0)$	
$(0, -1)$	0	$(0, -1)$	$(0, 0)$	
$(0, 1)$	0	$(0, 1)$	$(0, 0)$	
$(3, 2)$	-3	$(0, 2)$	$(0, -0.06)$	
$(4, 2)$	-4	$(0, 2)$	$(0, -0.08)$	
		total weight change	$(0, -0.34)$	$(-1, -0.34)$

Epoch 2, initial $\mathbf{w} = [-1, -0.34]$

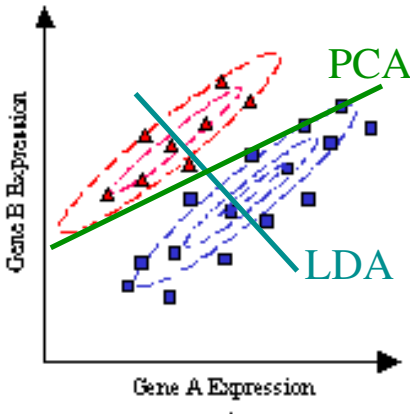
$\mathbf{x}^t$	$y = \mathbf{wx}$	$\mathbf{x}^t - y\mathbf{w}$	$\eta y(\mathbf{x}^t - y\mathbf{w})$	$\mathbf{w}$
$(-5, -4)$	6.36	$(1.36, -1.84)$	$(0.087, -0.117)$	
$(-2, 0)$	2	$(0, 0.68)$	$(0, 0.0136)$	
$(0, -1)$	0.34	$(0.34, -0.88)$	$(0.001, -0.003)$	
$(0, 1)$	-0.34	$(-0.34, 0.88)$	$(0.001, -0.003)$	
$(3, 2)$	-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	$(0.146, -0.156)$	$(-0.854, -0.496)$

Note:
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: $\mathbf{w} = (-0.8302, -0.5556)$
cf., result for previous question when calculating first PC via the Karhunen-Loève Transform.

8. The graph below shows a two-dimensional dataset in which examplars come from two classes. Exemplars from one class are plotted using triangular markers, and exemplars from the other class are plotted using square markers.



- **Draw the approximate direction of the first principal component of this data.**
- **Draw the approximate direction of the axis onto which the data would be projected using LDA.**



9. Briefly describe the optimisation performed by Fisher's Linear Discriminant Analysis to find a projection of the original data onto a subspace.

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.

10. For the data in the Table below use Fisher's method to determine which of the following projection weights is more effective at performing Linear Discriminant Analysis (LDA).

- $\mathbf{w}^T = [-1, 5]$
- $\mathbf{w}^T = [2, -3]$

Class	Feature vector $\mathbf{x}^T$
1	$[1, 2]$
1	$[2, 1]$
1	$[3, 3]$
2	$[6, 5]$
2	$[7, 8]$

Fisher's Linear Discriminant Analysis maximise the cost function $J(\mathbf{w}) = \frac{sb}{sw}$, *where:*

$$sb = |\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)|^2$$

$$sw = \sum_{\mathbf{x}\in\omega_1} (\mathbf{w}^T(\mathbf{x} - \mathbf{m}_1))^2 + \sum_{\mathbf{x}\in\omega_2} (\mathbf{w}^T(\mathbf{x} - \mathbf{m}_2))^2$$

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{x\in\omega_i} x$$

Sample mean for class 1 is $\mathbf{m}_1^T = \frac{1}{3}([1, 2] + [2, 1] + [3, 3]) = [2, 2]$.

Sample mean for class 2 is $\mathbf{m}_2^T = \frac{1}{2}([6, 5] + [7, 8]) = [6.5, 6.5]$.

$\mathbf{w}^T = [-1, 5]$*:*
Between class scatter (sb) = $|\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)|^2$
$= |[-1, 5] \times ([2, 2] - [6.5, 6.5])^T|^2 = |[-1, 5] \times [-4.5, -4.5]^T|^2 = |-18|^2 = 324$

Within class scatter (sw) = $\sum_{\mathbf{x}\in\omega_1} (\mathbf{w}^T(\mathbf{x} - \mathbf{m}_1))^2 + \sum_{\mathbf{x}\in\omega_2} (\mathbf{w}^T(\mathbf{x} - \mathbf{m}_2))^2$
$= ([-1, 5] \times ([1, 2] - [2, 2])^T)^2 + ([-1, 5] \times ([2, 1] - [2, 2])^T)^2 + ([-1, 5] \times ([3, 3] - [2, 2])^T)^2 + ([-1, 5] \times ([6, 5] - [6.5, 6.5])^T)^2 + ([-1, 5] \times ([7, 8] - [6.5, 6.5])^T)^2 = 140$

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

For $\mathbf{w}^T = [2, -3]$*:*
Between class scatter (sb) = $|\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)|^2$
$= |[2, -3] \times ([2, 2] - [6.5, 6.5])^T|^2 = |[2, -3] \times [-4.5, -4.5]^T|^2 = |4.5|^2 = 20.25$

Within class scatter (sw) = $\sum_{\mathbf{x}\in\omega_1} (\mathbf{w}^T(\mathbf{x} - \mathbf{m}_1))^2 + \sum_{\mathbf{x}\in\omega_2} (\mathbf{w}^T(\mathbf{x} - \mathbf{m}_2))^2$
$= ([2, -3] \times ([1, 2] - [2, 2])^T)^2 + ([2, -3] \times ([2, 1] - [2, 2])^T)^2 + ([2, -3] \times ([3, 3] - [2, 2])^T)^2 + ([2, -3] \times ([6, 5] - [6.5, 6.5])^T)^2 + ([2, -3] \times ([7, 8] - [6.5, 6.5])^T)^2 = 38.5$

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

As $J(\mathbf{w})$ *given by* $\mathbf{w}^T = [-1, 5]$ *is higher, it is a more effective projection weight.*

Note, projection of the data into the new feature space defined by the two projection weights is:

Class	*Feature vector* $\mathbf{x}^T$	$y = \mathbf{w}^T\mathbf{x}$	
		$\mathbf{w}^T = [-1, 5]$	$\mathbf{w}^T = [2, -3]$
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	$[7, 8]$	$[-1, 5] \times [7, 8]^T = 33$	$[2, -3] \times [7, 8]^T = -10$

It can be seen that $\mathbf{w}^T = [-1, 5]$ *makes the data linearly separable, while* $\mathbf{w}^T = [2, -3]$ *does not.*

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:

$$\mathbf{V} = \begin{pmatrix} -0.62 & 0.44 & -0.91 \\ -0.81 & -0.09 & 0.02 \\ 0.74 & -0.91 & -0.60 \\ -0.82 & -0.92 & 0.71 \\ -0.26 & 0.68 & 0.15 \\ 0.80 & -0.94 & -0.83 \end{pmatrix}$$

The weights to the output neuron are: $\mathbf{w} = (0, 0, 0, -1, 0, 0, 2)$**. 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:** $\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$**.**

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:*

$$\mathbf{Y} = H[\mathbf{VX}] = H\left[\begin{pmatrix} -0.62 & 0.44 & -0.91 \\ -0.81 & -0.09 & 0.02 \\ 0.74 & -0.91 & -0.60 \\ -0.82 & -0.92 & 0.71 \\ -0.26 & 0.68 & 0.15 \\ 0.80 & -0.94 & -0.83 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}\right]$$

$$\mathbf{Y} = H\begin{pmatrix} -0.62 & -1.53 & -0.18 & -1.09 \\ -0.81 & -0.79 & -0.90 & -0.88 \\ 0.74 & 0.14 & -0.17 & -0.77 \\ -0.82 & -0.11 & -1.74 & -1.03 \\ -0.26 & -0.11 & 0.42 & 0.57 \\ 0.80 & -0.03 & -0.14 & -0.97 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

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:*

$$\mathbf{Z} = \mathbf{wY} = \begin{pmatrix} 0 & 0 & 0 & -1 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} = (1 \quad -1 \quad 0 \quad 0)$$

12. Given a dictionary, $\mathbf{V}^t$, what is the best sparse code for the signal $\mathbf{x}$ out of the following two alternatives:
i) $\mathbf{y_1}^t = (1,0,0,0,1,0,0,0)$
ii) $\mathbf{y_2}^t = (0,0,1,0,0,0,-1,0)$
Where $\mathbf{V}^t = \begin{pmatrix} 0.4 & 0.55 & 0.5 & -0.1 & -0.5 & 0.9 & 0.5 & 0.45 \\ -0.6 & -0.45 & -0.5 & 0.9 & -0.5 & 0.1 & 0.5 & 0.55 \end{pmatrix}$**, and** $\mathbf{x} = \begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix}$**. Assume that sparsity is measured as the count of elements that are non-zero.**

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

For (i) error = $\|\mathbf{x} - \mathbf{V}^t\mathbf{y_1}\|_2$

$$= \left\| \begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} 0.4 & 0.55 & 0.5 & -0.1 & -0.5 & 0.9 & 0.5 & 0.45 \\ -0.6 & -0.45 & -0.5 & 0.9 & -0.5 & 0.1 & 0.5 & 0.55 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right\|_2$$

$$= \left\| \begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} 0.4 - 0.5 \\ -0.6 - 0.5 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} -0.1 \\ -1.1 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} 0.05 \\ 0.15 \end{pmatrix} \right\|_2$$

$$= \sqrt{0.05^2 + 0.15^2} = 0.158$$

For (ii) error = $\|\mathbf{x} - \mathbf{V}^t\mathbf{y_2}\|_2$

$$= \left\| \begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} 0.4 & 0.55 & 0.5 & -0.1 & -0.5 & 0.9 & 0.5 & 0.45 \\ -0.6 & -0.45 & -0.5 & 0.9 & -0.5 & 0.1 & 0.5 & 0.55 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{pmatrix} \right\|_2$$

$$= \left\| \begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} 0.5 - 0.5 \\ -0.5 - 0.5 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} 0 \\ -1 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} -0.05 \\ 0.05 \end{pmatrix} \right\|_2$$

$$= \sqrt{0.05^2 + 0.05^2} = 0.071$$

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

13. Repeat the previous questions when the two alternatives are:
i) $\mathbf{y_1}^t = (1,0,0,0,1,0,0,0)$
ii) $\mathbf{y_2}^t = (0,0,0,-1,0,0,0,0)$

(i) is the same as in the previous question, therefore for (i) the error is 0.158.

For (ii) error = $\|\mathbf{x} - \mathbf{V}^t\mathbf{y_2}\|_2$

$$\left\| \begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} 0.4 & 0.55 & 0.5 & -0.1 & -0.5 & 0.9 & 0.5 & 0.45 \\ -0.6 & -0.45 & -0.5 & 0.9 & -0.5 & 0.1 & 0.5 & 0.55 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right\|_2$$

$$= \left\| \begin{pmatrix} -0.05 \\ -0.95 \end{pmatrix} - \begin{pmatrix} 0.1 \\ -0.9 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} -0.15 \\ -0.05 \end{pmatrix} \right\|_2 = \sqrt{0.15^2 + 0.05^2} = 0.158$$

Hence, error is the same in both cases. We should therefore prefer the sparser solution, which is solution (ii).