This document is intended to be exported as Anki flashcards.
Answers generated/derived by/from AI are coloured in blue! (Gemini/NotebookLM in use)
"Open-book" answers are marked as purple, corrections from model solutions are coloured as green.
Any arguments arising from group discussion are marked as fuchsia.

# 7CCSMPNN
## Practice Questions

If you spot anything wrong, please let me know!
discord: @insertish
izzy · insrt.uk

## Contents

# 1. Tutorial 1

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

- ✗ Classifying numbers into primes and non-primes. — can be more easily and more accurately performed analytically
- ✓ Detecting potential fraud in credit card charges.
- ✗ Determining where a cannon ball is likely to land given information about elevation and direction of the barrel, wind direction, etc. — better to do it analytically, i.e. Newton's equations
- it depends – Identifying the species a newly discovered "bug" belongs to. — some species are defined by simple rules; other data, such as images of different bugs, are a good application

**What types of learning best describes these scenarios?**

- measurements of a large number of coins is taken, algorithm finds these measurements fall in several 'bins'; it finds decision boundaries that separate these binds and uses these to classify new coins — **unsupervised learning (clustering)**
- measurements of each coin presented to classifier which makes a decision; classifier changes its decision boundaries based on whether decision was correct or incorrect — **reinforcement learning**
- measurements of a large number of coins are taken; algorithm uses data with known class labels to infer decision boundaries which is used to classify new coins — **supervised learning (classification)**

**Identify which of these datasets are in the given categories**

| 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) |

1. univariate-continuous — dataset 2
2. multivariate-discrete — dataset 3
3. multivariate-continuous — dataset 4
4. univariate-discrete — dataset 1

**Draw a confusion matrix for the following data**

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 |

Confusion matrix is as follows:

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

**Given the confusion matrix, calculate performance metrics**

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

- error-rate:

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

- the accuracy:

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

- the recall:

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

- the precision:

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

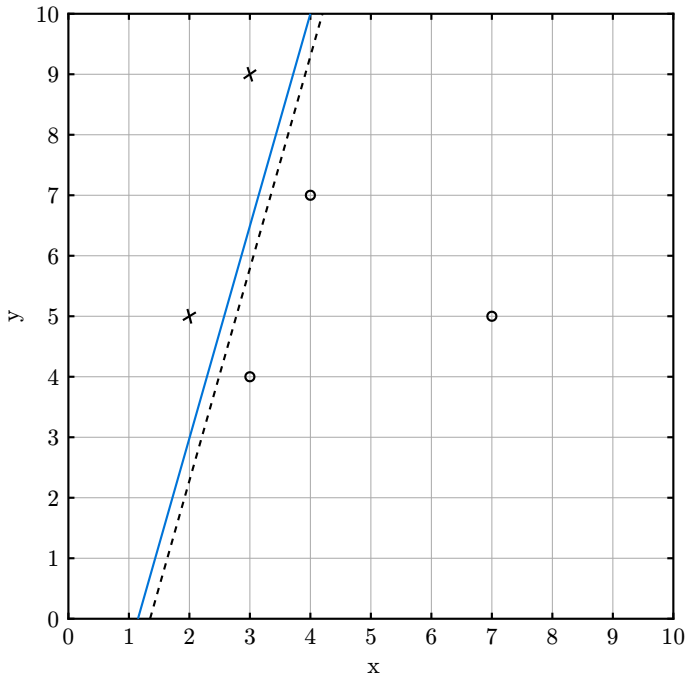
- the $f_1$-score:

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

equivalent to $\frac{2\times \text{ recall } \times \text{ precision}}{\text{recall } + \text{ precision}}$

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**?

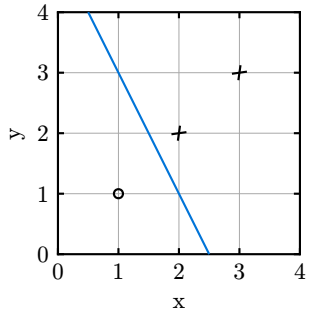
| Class | Features |
|---|---|
| 1 | (3, 4) |
| 1 | (4, 7) |
| 1 | (7, 5) |
| 2 | (2, 5) |
| 2 | (3, 9) |



## 2. Tutorial 2

Consider dichotomizer defined using linear discriminant function $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$ where $\boldsymbol{w} = \begin{pmatrix}2\\1\end{pmatrix}$ and $w_0 = -5$. **Plot decision boundary and determine class of feature vectors:**

$$\boldsymbol{x}_1 = \begin{pmatrix}1\\1\end{pmatrix}, \boldsymbol{x}_2 = \begin{pmatrix}2\\2\end{pmatrix}, \boldsymbol{x}_3 = \begin{pmatrix}3\\3\end{pmatrix}$$




$x_1$ is class 2, $x_2$ and $x_3$ are class 1
**Note!** $\boldsymbol{w}$ is the vector normal to the hyperplane, pointing towards class 1!

In augmented feature space, a dichotomizer is defined using $g(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{y}$ where $\boldsymbol{a}^T = (-5, 2, 1)$ and $y = \begin{pmatrix}1\\\boldsymbol{x}\end{pmatrix}$. **Determine class of feature vectors:**

$$\boldsymbol{x}_1 = \begin{pmatrix}1\\1\end{pmatrix}, \boldsymbol{x}_2 = \begin{pmatrix}2\\2\end{pmatrix}, \boldsymbol{x}_3 = \begin{pmatrix}3\\3\end{pmatrix}$$

$$\begin{pmatrix}1\\1\\1\end{pmatrix}(-5, 2, 1) = -5 + 2 + 1 = -2$$

$$\begin{pmatrix}1\\2\\2\end{pmatrix}(-5, 2, 1) = -5 + 4 + 2 = 1$$

$$\begin{pmatrix}1\\3\\3\end{pmatrix}(-5, 2, 1) = -5 + 6 + 3 = 4$$

$x_1$ is class 2, $x_2$ and $x_3$ are class 1

Consider a 3-dimensional feature space and quadratic discriminant function:

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

which defines two classes $\omega_1, \omega_2$ s.t. $g(\boldsymbol{x}) > 0$ if $x \in \omega_1$ and $g(\boldsymbol{x}) \leq 0$ if $x \in \omega_2$.
**Determine the class of the feature vectors:** $\boldsymbol{x}_1 = (1, 1, 1)^T, \boldsymbol{x}_2 = (-1, 0, 3)^T, \boldsymbol{x}_3 = (-1, 0, 0)^T$.

$$\begin{aligned}(1, 1, 1)^T &\Rightarrow 1^2 - 1^2 + 2 \cdot 1 \cdot 1 + 4 \cdot 1 \cdot 1 + 3 \cdot 1 - 2 \cdot 1 + 2 = 9\\(-1, 0, 3)^T &\Rightarrow (-1)^2 - 3^2 + 0 + 0 + 3 \cdot (-1) - 0 + 2 = -9\\(-1, 0, 0)^T &\Rightarrow 1 - 0 + 0 + 0 - 3 - 0 + 2 = 0\end{aligned}$$

$x_1$ is class 1, $x_2, x_3$ are class 2

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

| $\boldsymbol{x}^T$ | $(1,5)$ | $(2,5)$ | $(4,1)$ | $(5,1)$ |
|---|---|---|---|---|
| class | 1 | 1 | 2 | 2 |

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

Using augmented notation and sample normalisation, the dataset is:

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

The update rule is

$$\boldsymbol{a} \leftarrow \boldsymbol{a} + \eta \sum_{y \in \mathcal{X}} \boldsymbol{y}$$

where $\eta = 1$

The dichotomizer is defined as

$$g(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{y}$$

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

| $\boldsymbol{y}^T$ | $g(\boldsymbol{x})$ | misclassified? $- g(\boldsymbol{x})$ |
|---|---|---|
| $(1,1,5)$ | $-4$ | yes |
| $(1,2,5)$ | 2 | no |
| $(-1,-4,-1)$ | $-2$ | yes |
| $(-1,-5,-1)$ | $-8$ | yes |

Make adjustments to $\boldsymbol{a}$:

$$\boldsymbol{a} \leftarrow \begin{pmatrix} -25 \\ 6 \\ 3 \end{pmatrix} + 1 \cdot \left( \begin{pmatrix} 1 \\ 1 \\ 5 \end{pmatrix} + \begin{pmatrix} -1 \\ -4 \\ -1 \end{pmatrix} + \begin{pmatrix} -1 \\ -5 \\ -1 \end{pmatrix} \right) = \begin{pmatrix} -26 \\ -2 \\ 6 \end{pmatrix}$$

Epoch 2 finds misclassified sample 2, hence:

Make adjustments to $\boldsymbol{a}$:

$$\boldsymbol{a} \leftarrow \begin{pmatrix} -26 \\ -2 \\ 6 \end{pmatrix} + 1 \cdot \left( \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix} \right) = \begin{pmatrix} -25 \\ 0 \\ 11 \end{pmatrix}$$

Epoch 3 finds no misclassified samples. Learning has converged.

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

| $\boldsymbol{x}^T$ | $(1,5)$ | $(2,5)$ | $(4,1)$ | $(5,1)$ |
|---|---|---|---|---|
| class | 1 | 1 | 2 | 2 |

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

Using augmented notation and sample normalisation, the dataset is:

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

The update rule is

$$\boldsymbol{a} \leftarrow \boldsymbol{a} + \eta \cdot \boldsymbol{y}_k$$

where $\eta = 1$

The dichotomizer is defined as

$$g(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{y}$$

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

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

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

| $\boldsymbol{y}^T$ | $g(\boldsymbol{x})$ | $\boldsymbol{a}^T$ |
|---|---|---|
| $(1,1,5)$ | 13 | $(-25,3,7)$ |
| $(1,2,5)$ | 16 | $(-25,3,7)$ |
| $(-1,-4,-1)$ | 6 | $(-25,3,7)$ |
| $(-1,-5,-1)$ | 3 | $(-25,3,7)$ |

Learning has converged.

**Write pseudocode for sequential perceptron learning algorithm**

- Initialise $\eta$, learning rate parameter
- Initialise $\boldsymbol{a}$ to random values
- While solution has not converged ($\boldsymbol{a}$ is still changing):
  - For each sample $\boldsymbol{y}_k$:
    - Calculate $g(\boldsymbol{x}_k) = \boldsymbol{a}^T \boldsymbol{y}_k$
    - If sample $\boldsymbol{y}_k$ is misclassified by $g(\boldsymbol{x})$ (i.e. $\leq 0$):

$$\boldsymbol{a} \leftarrow \boldsymbol{a} + \eta \cdot \boldsymbol{y}_k$$

Consider linearly separable dataset:

| $\boldsymbol{x}^T$ | $(0,2)$ | $(1,2)$ | $(2,1)$ | $(-3,1)$ | $(-2,-1)$ | $(-3,-2)$ |
|---|---|---|---|---|---|---|
| class($\omega$) | 1 | 1 | 1 | $-1$ | $-1$ | $-1$ |

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

To apply sequential learning, we must:

- Until $\boldsymbol{a}$ stops changing (converged):
  - For each sample:
    - Calculate $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$
    - If misclassified, update $\boldsymbol{a} \leftarrow \boldsymbol{a} + \eta \cdot \boldsymbol{\omega}_k \cdot \boldsymbol{x}_k$

Epoch 1, $\boldsymbol{a}^T = (1,0,0)$

| class($\omega$) | $\boldsymbol{x}^T$ | $\boldsymbol{y}^T = (1, \boldsymbol{x}^T)$ | $g(\boldsymbol{x}) = w_0 + \boldsymbol{w}^T\boldsymbol{x}$ | $\boldsymbol{a}^T$ |
|---|---|---|---|---|
| 1 | $(0,2)$ | $(1,0,2)$ | 1 | $(1,0,0)$ |
| 1 | $(1,2)$ | $(1,1,2)$ | 1 | $(1,0,0)$ |
| 1 | $(2,1)$ | $(1,2,1)$ | 1 | $(1,0,0)$ |
| $-1$ | $(-3,1)$ | $(1,-3,1)$ | 1 | $(1,0,0) + (-1)(1,-3,1) = (0,3,-1)$ |
| $-1$ | $(-2,-1)$ | $(1,-2,-1)$ | $-5$ | $(0,3,-1)$ |
| $-1$ | $(-3,-2)$ | $(1,-3,-2)$ | $-7$ | $(0,3,-1)$ |

Epoch 2, $\boldsymbol{a}^T = (0,3,-1)$

| class($\omega$) | $\boldsymbol{x}^T$ | $\boldsymbol{y}^T = (1, \boldsymbol{x}^T)$ | $g(\boldsymbol{x}) = w_0 + \boldsymbol{w}^T\boldsymbol{x}$ | $\boldsymbol{a}^T$ |
|---|---|---|---|---|
| 1 | $(0,2)$ | $(1,0,2)$ | $-2$ | $(0,3,-1) + (1)(1,0,2) = (1,3,1)$ |
| 1 | $(1,2)$ | $(1,1,2)$ | 6 | $(1,3,1)$ |
| 1 | $(2,1)$ | $(1,2,1)$ | 8 | $(1,3,1)$ |
| $-1$ | $(-3,1)$ | $(1,-3,1)$ | $-7$ | $(1,3,1)$ |
| $-1$ | $(-2,-1)$ | $(1,-2,-1)$ | $-6$ | $(0,3,1)$ |
| $-1$ | $(-3,-2)$ | $(1,-3,-2)$ | $-10$ | $(1,3,1)$ |

Epoch 3, $\boldsymbol{a}^T = (1,3,1)$

| class($\omega$) | $\boldsymbol{x}^T$ | $\boldsymbol{y}^T = (1, \boldsymbol{x}^T)$ | $g(\boldsymbol{x}) = w_0 + \boldsymbol{w}^T\boldsymbol{x}$ | $\boldsymbol{a}^T$ |
|---|---|---|---|---|
| 1 | $(0,2)$ | $(1,0,2)$ | 3 | $(1,3,1)$ |

Learning has converged!

Consider linearly separable dataset:

| $\boldsymbol{x}^T$ | $(0,2)$ | $(1,2)$ | $(2,1)$ | $(-3,1)$ | $(-2,-1)$ | $(-3,-2)$ |
|---|---|---|---|---|---|---|
| class($\omega$) | 1 | 1 | 1 | $-1$ | $-1$ | $-1$ |

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

To apply sequential learning, we must:

- Until $\boldsymbol{a}$ stops changing (converged):
  - For each sample:
    - Calculate $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$
    - If misclassified, update $\boldsymbol{a} \leftarrow \boldsymbol{a} + \eta \cdot \boldsymbol{\omega}_k \cdot \boldsymbol{x}_k$

To normalise the samples: generate $\boldsymbol{y}_k = \omega_k \cdot (1, \boldsymbol{x}^T)$ for each sample

Learning summary:
Results will converge on $\boldsymbol{a}^T = (1,3,1)$ as seen before with the same values.
Our weight updates are simply $\boldsymbol{a}^T \leftarrow \boldsymbol{a}^T + \eta \cdot \boldsymbol{y}_k$ when misclassification occurs.

A dataset contains 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 sequential multi-class perceptron learning algorithm to find parameters for three linear discriminant functions that will correctly classify the data.** Assume all initial parameters are zero and learning rate is one. Note: when more than discriminant function produces maximum output, choose function that represents the largest class label.

The procedure works as follows:

- Until parameters stop changing:
  - Predict class $c = \arg\max_k(\boldsymbol{a}_k^T \boldsymbol{y}_k)$
  - If true class $k$ is not predicted class $c$:
    - Adjust $\boldsymbol{a}_k^T \leftarrow \boldsymbol{a}_k^T + \eta \cdot \boldsymbol{y}_k$ — add to real
    - Adjust $\boldsymbol{a}_c^T \leftarrow \boldsymbol{a}_c^T - \eta \cdot \boldsymbol{y}_k$ — remove from pred.

| Epoch | $\omega$ | $\boldsymbol{y}_k = (1, \boldsymbol{x}_k)$ | $g(\boldsymbol{x}) = \arg\max_k(\boldsymbol{a}_k^T \boldsymbol{y}_k)$ | $\boldsymbol{a}^T \leftarrow$ | | |
|---|---|---|---|---|---|---|
| | | | | $\boldsymbol{a}_1^T$ | $\boldsymbol{a}_2^T$ | $\boldsymbol{a}_3^T$ |
| | | | initial | $(0,0,0)$ | $(0,0,0)$ | $(0,0,0)$ |
| 1 | $(1,1,1)$ | 1 | $[0,0,0] \to 1$ | $(1,1,1)$ | $(0,0,0)$ | $(-1,-1,-1)$ |
| | $(1,2,0)$ | 1 | $[3,0,-3] \to 1$ | $(1,1,1)$ | $(0,0,0)$ | $(-1,-1,-1)$ |
| | $(1,0,2)$ | 2 | $[3,0,-3] \to 1$ | $(0,1,-1)$ | $(1,0,2)$ | $(-1,-1,-1)$ |
| | $(1,-1,1)$ | 2 | $[-2,3,-1] \to 2$ | $(0,1,-1)$ | $(1,0,2)$ | $(-1,-1,-1)$ |
| | $(1,-1,-1)$ | 3 | $[0,-1,1] \to 3$ | $(0,1,-1)$ | $(1,0,2)$ | $(-1,-1,-1)$ |
| 2 | $(1,1,1)$ | 1 | $[0,3,-3] \to 2$ | $(1,2,0)$ | $(0,-1,1)$ | $(-1,-1,-1)$ |
| | $(1,2,0)$ | 1 | $[5,-2,-3] \to 1$ | $(1,2,0)$ | $(0,-1,1)$ | $(-1,-1,-1)$ |
| | $(1,0,2)$ | 2 | $[1,2,-3] \to 2$ | $(1,2,0)$ | $(0,-1,1)$ | $(-1,-1,-1)$ |
| | $(1,-1,1)$ | 2 | $[-1,2,-1] \to 2$ | $(1,2,0)$ | $(0,-1,1)$ | $(-1,-1,-1)$ |
| | $(1,-1,-1)$ | 3 | $[-1,0,1] \to 3$ | $(1,2,0)$ | $(0,-1,1)$ | $(-1,-1,-1)$ |
| 3 | $(1,1,1)$ | 1 | $[3,0,-3] \to 1$ | $(1,2,0)$ | $(0,-1,1)$ | $(-1,-1,-1)$ |

Learning has converged.

INCOMPLETE ex. 14–16

## 3. Tutorial 3

A neuron has a transfer function which is a linear weighted sum of its input and an activation function that is the Heaviside function. If the weights are $\boldsymbol{w} = [0.1, -0.5, 0.4]$, and the threshold zero, **what is the output of this neuron when the input is: $\boldsymbol{x_1} = \boldsymbol{[0.1, -0.5, 0.4]^t}$ and $\boldsymbol{x_2} = \boldsymbol{[0.1, 0.5, 0.4]^t}$?**

$$y_j = \underbrace{H\left(\overbrace{\sum_i w_{ji}x_i - \theta_j}^{\text{transfer function}}\right)}_{\text{activation function}} \qquad \text{where } H(u) = \text{heaviside step function} = \begin{cases} 1, \text{if } u \geq 0 \\ 0, \text{otherwise} \end{cases}$$

$$\begin{aligned} y_1 &= H(0.01 + 0.25 + 0.16 - 0.00) = H(0.42) \quad &= 1 \\ y_2 &= H(0.01 - 0.25 + 0.16 - 0.00) = H(-0.18) &= 0 \end{aligned}$$

A Linear Threshold Unit has one input, $x_1$, that can take binary values. **Apply sequential Delta learning rule so that the output of this neuron, $y$, is equal to $\overline{x_1}$ (i.e. $\neg x_1$), such that:**

| $x_1$ | $y$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

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

Augmented notation: $y = H(\boldsymbol{wx})$ where $\boldsymbol{w} = [-\theta, w_1]$ and $\boldsymbol{x} = [1, x_1]^T$
Sequential (online) update is: $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta(t - y)\boldsymbol{x}^T$

| Epoch | $\boldsymbol{x}^T = [1, x_1]$ | $t$ | $y = H(\boldsymbol{wx})$ | $\eta(t-y)\boldsymbol{x}^t$ | $\boldsymbol{w}$ |
|---|---|---|---|---|---|
| | | | | initial | $[-1.5, 2]$ |
| 1 | $[1, 0]$ | 1 | $H(-1.5) = 0$ | $[1, 0]$ | $[-0.5, 2]$ |
| | $[1, 1]$ | 0 | $H(1.5) = 1$ | $[-1, -1]$ | $[-1.5, 1]$ |
| 2 | $[1, 0]$ | 1 | $H(-1.5) = 0$ | $[1, 0]$ | $[-0.5, 1]$ |
| | $[1, 1]$ | 0 | $H(0.5) = 1$ | $[-1, -1]$ | $[-1.5, 0]$ |
| 3 | $[1, 0]$ | 1 | $H(-1.5) = 0$ | $[1, 0]$ | $[-0.5, 0]$ |
| | $[1, 1]$ | 0 | $H(-0.5) = 0$ | $[0, 0]$ | $[-0.5, 0]$ |
| 4 | $[1, 0]$ | 1 | $H(-0.5) = 0$ | $[1, 0]$ | $[0.5, 0]$ |
| | $[1, 1]$ | 0 | $H(0.5) = 1$ | $[-1, -1]$ | $[-0.5, -1]$ |
| 5 | $[1, 0]$ | 1 | $H(-0.5) = 0$ | $[1, 0]$ | $[0.5, -1]$ |
| | $[1, 1]$ | 0 | $H(-0.5) = 0$ | $[0, 0]$ | $[0.5, -1]$ |
| 6 | $[1, 0]$ | 1 | $H(0.5) = 1$ | $[0, 0]$ | $[0.5, -1]$ |
| | $[1, 1]$ | 0 | $H(-0.5) = 0$ | $[0, 0]$ | $[0.5, -1]$ |

Learning has converged.

A Linear Threshold Unit has one input, $x_1$, that can take binary values. **Apply batch Delta learning rule so that the output of this neuron, $y$, is equal to $\overline{x_1}$ (i.e. $\neg x_1$), such that:**

| $x_1$ | $y$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

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

Augmented notation: $y = H(\boldsymbol{w}\boldsymbol{x})$ where $\boldsymbol{w} = [-\theta, w_1]$ and $\boldsymbol{x} = [1, x_1]^T$
Sequential (online) update is: $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \sum_p (t_p - y_p)\boldsymbol{x}_p^T$

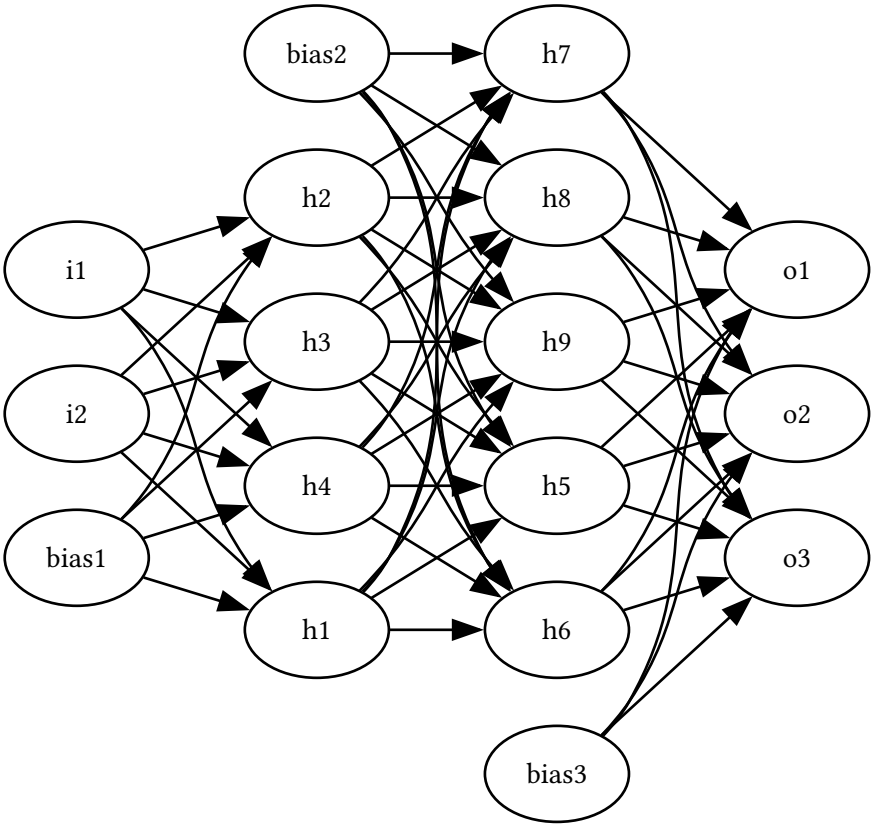
| Epoch | $\boldsymbol{x}^T = [1, x_1]$ | $t$ | $H(\boldsymbol{w}\boldsymbol{x})$ | $(t-y)\boldsymbol{x}^t$ | $\eta \sum_p [...]$ | $\boldsymbol{w}$ |
|---|---|---|---|---|---|---|
| | | | | | initial | $[-1.5, 2]$ |
| 1 | $[1,0]$ | 1 | $H(-1.5) = 0$ | $[1,0]$ | $[0,-1]$ | $[-1.5, 1]$ |
| | $[1,1]$ | 0 | $H(0.5) = 1$ | $[-1,-1]$ | | |
| 2 | $[1,0]$ | 1 | $H(-1.5) = 0$ | $[1,0]$ | $[1,0]$ | $[-0.5, 1]$ |
| | $[1,1]$ | 0 | $H(-0.5) = 0$ | $[0,0]$ | | |
| 3 | $[1,0]$ | 1 | $H(-0.5) = 0$ | $[1,0]$ | $[0,-1]$ | $[-0.5, 0]$ |
| | $[1,1]$ | 0 | $H(0.5) = 1$ | $[-1,-1]$ | | |
| 4 | $[1,0]$ | 1 | $H(-0.5) = 0$ | $[1,0]$ | $[1,0]$ | $[0.5, 0]$ |
| | $[1,1]$ | 0 | $H(-0.5) = 0$ | $[0,0]$ | | |
| 5 | $[1,0]$ | 1 | $H(0.5) = 1$ | $[0,0]$ | $[-1,-1]$ | $[-0.5, -1]$ |
| | $[1,1]$ | 0 | $H(0.5) = 1$ | $[-1,-1]$ | | |
| 6 | $[1,0]$ | 1 | $H(-0.5) = 0$ | $[1,0]$ | $[1,0]$ | $[0.5, -1]$ |
| | $[1,1]$ | 0 | $H(-1.5) = 0$ | $[0,0]$ | | |
| 7 | $[1,0]$ | 1 | $H(0.5) = 1$ | $[0,0]$ | $[0,0]$ | $[0.5, -1]$ |
| | $[1,1]$ | 0 | $H(-0.5) = 0$ | $[0,0]$ | | |

Learning has converged.

INCOMPLETE ex. 5–10

# 4. Tutorial 4

**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?**




No. of connection weights = $(2+1)\cdot 4 + (4+1)\cdot 5 + (5+1)\cdot 3 = 55$ connection weights

**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:

- $d$ - no. of inputs
- $L$ - no. of hidden layers
- $n_h$ - no. of units in $h$-th hidden layer
- $h = 1, 2, \ldots, L$
- $z$ - no. of outputs

$$(d+1)\cdot n_1 + \sum_{h=1}^{L-1}(n_h+1)\cdot n_{h+1} + (n_L+1)\cdot z$$

INCOMPLETE ex. 3–8

# 5. Tutorial 5

The following array shows output produced by mask in convolutional layer of CNN:

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

**Calculate values when applying:**

- ReLU:

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

- LReLU when $a = 0.1$:

$$\text{net}_j = \begin{pmatrix} 1 & 0.5 & 0.2 \\ -0.1 & -0.05 & -0.02 \\ 0.1 & -0.01 & 0 \end{pmatrix}$$

- tanh:

$$\text{net}_j = \begin{pmatrix} 0.7616 & 0.4621 & 0.1974 \\ -0.7616 & -0.4621 & -0.1974 \\ 0.0997 & -0.0997 & 0 \end{pmatrix}$$

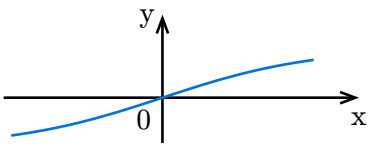
- heaviside with threshold $0.1$, let $H(0) = 0.5$:

$$\text{net}_j = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0.5 & 0 & 0 \end{pmatrix}$$

**tanh** activation function

$$\delta(\text{net}_j) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

**Graph**




$$\text{Var}(x) = \frac{1}{N} \sum_n (1 - x_n)^2$$

Following arrays are input to convolutional layer of a CNN:

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

**Calculate (flashcard note: describe how to get) the output of mask $H$ where:**

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

**For the following configurations:**

- padding = 0, stride = 1

$$X_1 \cdot H_1 + X_2 \cdot H_2 = \begin{pmatrix} -0.9 + 1.5 & 1.01 + 0.7 \\ -0.9 + 0.75 & 0.0 - 0.3 \end{pmatrix} = \begin{pmatrix} 0.6 & 1.71 \\ -1.65 & -0.3 \end{pmatrix}$$

$$\text{e.g. } (-0.9) = 1 \cdot 0.2 + 1 \cdot (-0.1) - 1 \cdot 1 + 0 \cdot (-0.9)$$

- padding = 1, stride = 1

$$X_1 \Rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2 & 1 & 0 & 0 \\ 0 & -1 & 0 & -0.1 & 0 \\ 0 & 0.1 & 0 & 0.1 & 0 \end{pmatrix}, \text{ similarly for } X_2$$

$$\text{as before } X_1 \cdot H_1 + X_2 \cdot H_2 \text{ with padding in } X_n$$

- padding = 1, stride = 2 ⟶ use same methodology but move mask across by two units each time
- padding = 0, stride = 1, dilation = 2

$$\begin{aligned} &[0.2 \cdot 1 + 0 \cdot (-0.1) + 0.1 \cdot 1 + 0.1 \cdot (-0.1)] \\ +&[1 \cdot 0.5 + 0.2 \cdot 0.5 + 0.1 \cdot (-0.5) + 0 \cdot (-0.5)] \\ &= [0.29] + [0.55] \\ &= [0.84] \end{aligned}$$

Following arrays show feature maps that provide input into convolutional layer of CNN:

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

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

$$\begin{aligned} Y &= 1 \cdot X_1 - 1 \cdot X_2 + 0.5 \cdot X_3 \\ &= \begin{bmatrix} (0.2 \cdot 1) + (1 \cdot -1) + (0.5 \cdot 0.5) & (1 \cdot 1) + (0.5 \cdot -1) + (-0.5 \cdot 0.5) + \ldots \\ \ldots \\ \ldots \end{bmatrix} \\ &= \begin{bmatrix} -0.55 & 0.25 & -0.25 \\ 0 & 0.3 & 0.1 \\ 0.25 & 0.35 & 0.2 \end{bmatrix} \end{aligned}$$

Given input to pooling layer of 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 when using:

- average pooling with region of 2x2 and stride = 2

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

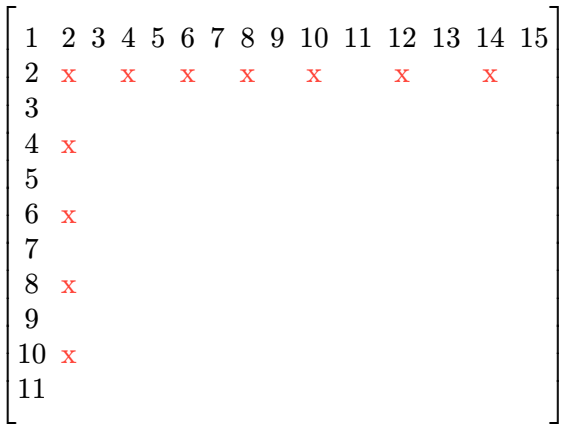
- max pooling with region 2x2 and stride = 2

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

- max pooling with region 3x3 and stride = 1

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

Input to convolutional layer of CNN consists of 6 feature maps each of which has a height of 11 and width of 15. **What is the size of output produced by a *single* mask of size 3x3 with 6 channels when using stride of 2 and padding of 0?**




Horizontal, 7, vertical, 5. Therefore, output is 5x7x1.

For any convolutional layer, the output dimension is given by:

$$\text{output}_{\text{dim}} = 1 + \frac{\text{input}_{\text{dim}} - \text{mask}_{\text{dim}} + 2 \cdot \text{padding}}{\text{stride}}$$

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

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

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

1. Create 40 channels of size $200 - 5 + 1 = 196$x196; 196x196x40
2. Retain 40 channels, shrink size to $196 = 96$x96; 98x98x40
3. Create 80 channels of size $1 + \frac{98-4+2\cdot1}{2} = 48$x48; 48x48x80
4. Retain 20 channels of equivalent size; 48x48x20

The following images show exemplars from two datasets.

Each is expanded using data augmentations, **which are appropriate?**

- ✓ rescaling
- **depends** horizontal flip (✗ for MNIST; ✓ for FashionMNIST)
- ✓ rotation (only small rot. for MNIST)
- ✓ cropping

# 6. Tutorial 6

INCOMPLETE ex. 1–3

# 7. Tutorial 7

> **Note**
>
> Skipped 1-4.

The eigenvectors and eigenvalues of some $\boldsymbol{C}$ (covariance matrix) are:

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

**What is the proportion of variance explained by the first 2 and 1 principal components?**
The proportion is given by the sum of eigenvalues for selected components divided by sum of all of the eigenvalues.

$$\frac{1.9015 + 0.0986}{1.9015 + 0.0986 + 0} = 1$$
$$\frac{1.9015}{1.9015 + 0.0986} = 0.95$$

Given the data $\binom{0}{1}, \binom{3}{5}, \binom{5}{4}, \binom{5}{6}, \binom{8}{7}, \binom{9}{7}$ which has mean $\mu = \binom{5}{5}$, **project on to first principle component using the KLT method.**

The covariance matrix is given by:

$$\begin{aligned} \boldsymbol{C} &= \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \mu)(\boldsymbol{x}_i - \mu)^T \\ &= \frac{1}{6}\left(\binom{-5}{-4}(-5-4) + \binom{-2}{0}(-2\ \ 0) + \ldots\right) \\ &= \frac{1}{6}\left(\begin{pmatrix} 25 & 20 \\ 20 & 16 \end{pmatrix} + \begin{pmatrix} 4 & 0 \\ 0 & 0 \end{pmatrix} + \ldots\right) \\ &= \begin{pmatrix} 9 & 5.67 \\ 5.67 & 4.33 \end{pmatrix} \end{aligned}$$

The eigenvalues/eigenvectors of $\boldsymbol{C}$ are given by:

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

Hence, choose eigenvector corresponding to largest eigenvalue, $\hat{\boldsymbol{V}} = \binom{-0.8309}{-0.5564}$.

The projection on to the subspace spanned by first principal component is given by:
Multiply the transpose eigenvector by the zero-mean data to find projected data!

$$\begin{aligned} \boldsymbol{y}_i &= \hat{\boldsymbol{V}}^T(\boldsymbol{x}_i - \mu) \\ &= (-0.8309\ \ -0.5564)\left(\boldsymbol{x}_i - \binom{5}{5}\right) \\ &= [6.3801\ \ 1.6618\ \ 0.5564\ \ \ldots] \end{aligned}$$

Given the data $\binom{0}{1}, \binom{3}{5}, \binom{5}{4}, \binom{5}{6}, \binom{8}{7}, \binom{9}{7}$, apply Oja's learning rule using a learning rate of $0.01$ and an initial weight vector of $(-1, 0)$.

Batch update rule is:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \sum_i y(\boldsymbol{x}_i^T - y\boldsymbol{w})$$

**Briefly describe the optimisation method performed by Fisher's LDA 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.

For the data given:

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

**Use Fisher's LDA method to determine which of the following projection weights is more effective at performing LDA: (i) $\boldsymbol{w^T = (-1, 5)}$, (ii) $\boldsymbol{w^T = (2, -3)}$**

Fisher's method maximises $J(\boldsymbol{w}) = \frac{\text{sb}}{\text{sw}} = \frac{\text{between class scatter}}{\text{within class scatter}}$

$$\text{sb} = \left|\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)\right|$$
$$\text{sw} = \sum \left(\boldsymbol{w}(\boldsymbol{x}_i^1 - \boldsymbol{m}_1)\right)^2 + \sum \left(\boldsymbol{w}(\boldsymbol{x}_i^2 - \boldsymbol{m}_2)\right)^2$$

So where $\boldsymbol{w}^T = (-1, 5)$, $\text{sb} = 324, \text{sw} = 140$, $J(\boldsymbol{w}) = 2.3143$.
So where $\boldsymbol{w}^T = (2, -3)$, $\text{sb} = 20.25, \text{sw} = 38.5$, $J(\boldsymbol{w}) = 0.526$.

Hence, the first weights are more effective.

INCOMPLETE ex. 11–13

# 8. Tutorial 8

A support vector machine is employed to classify the following points:

$$\text{class 1: } x_1 = \begin{pmatrix}1\\1\end{pmatrix}, \quad x_2 = \begin{pmatrix}1\\-1\end{pmatrix}$$

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

**Are these two points linearly separable?** Yes.
**Design an SVM classifier to correctly classify all given points, identifying support vectors.**

INCOMPLETE ex. 1–5

# 9. Tutorial 9

INCOMPLETE ex. 1–3

## 10. Tutorial 10

INCOMPLETE ex. 1–6