Machine Learning W6 Tutorial

COMP30027 | Sandy Luo

Overview

Measuring Distance

Euclidean, Manhattan...

KNN

Theory, code

SVM

Theory, code

01

Euclidean Distance

02

Manhattan Distance

03

Cosine Similarity

Distance Measures

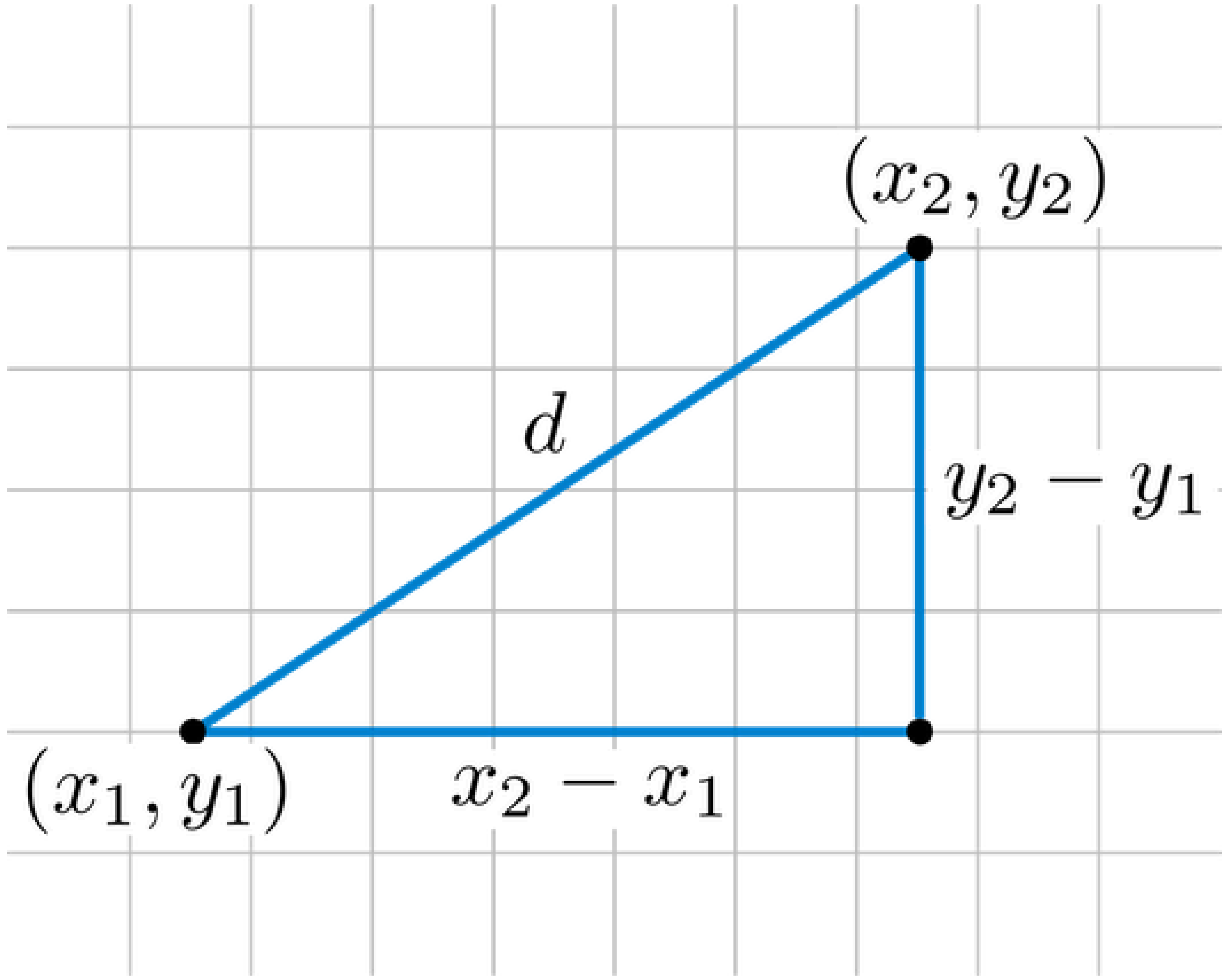
1. Euclidean Distance

- "Straight-line distance" (in 2D space)

- Given 2 items {A,B} and their features **a** and **b:**

In *n*-dimensional space:

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$



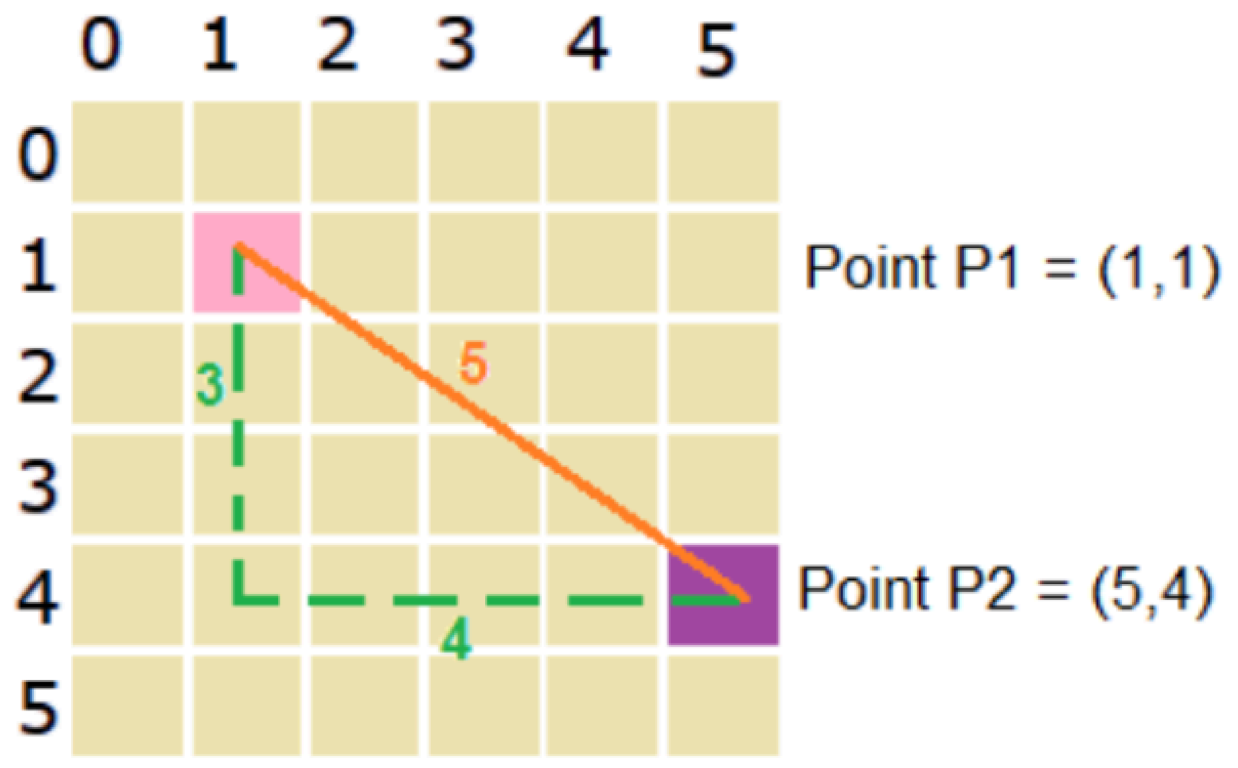
2. Manhattan Distance

- "Count the grid"

- Given 2 items {A,B} and their features **a** and **b:**

In *n*-dimensional space:

$$d(A,B) = \sum_{i=1}^{n} |a_i - b_i|$$

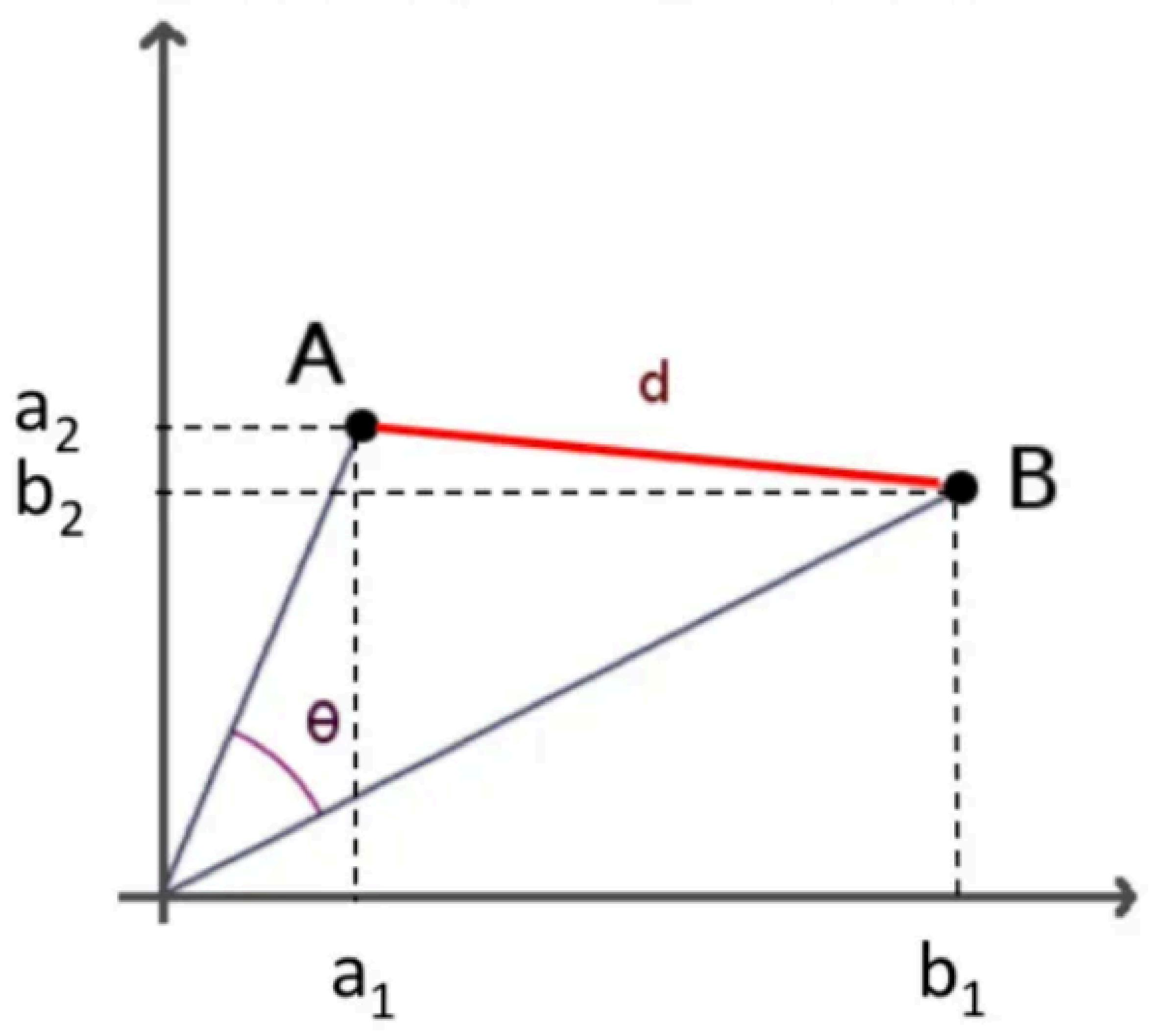


3. Cosine Similarity

- Given 2 items {A,B} and their feature vectors **a** and **b:**

In *n*-dimensional space:

$$cos(A,B) = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$



- Doesn't consider vector length, cares about direction

When to use what

- Is your data:
    - Sparse? Dense?
    - Continuous? Grid-like?
    - Do dimensions (features) have similar importance?
    - Does direction matter or magnitude?

01

Concept

02

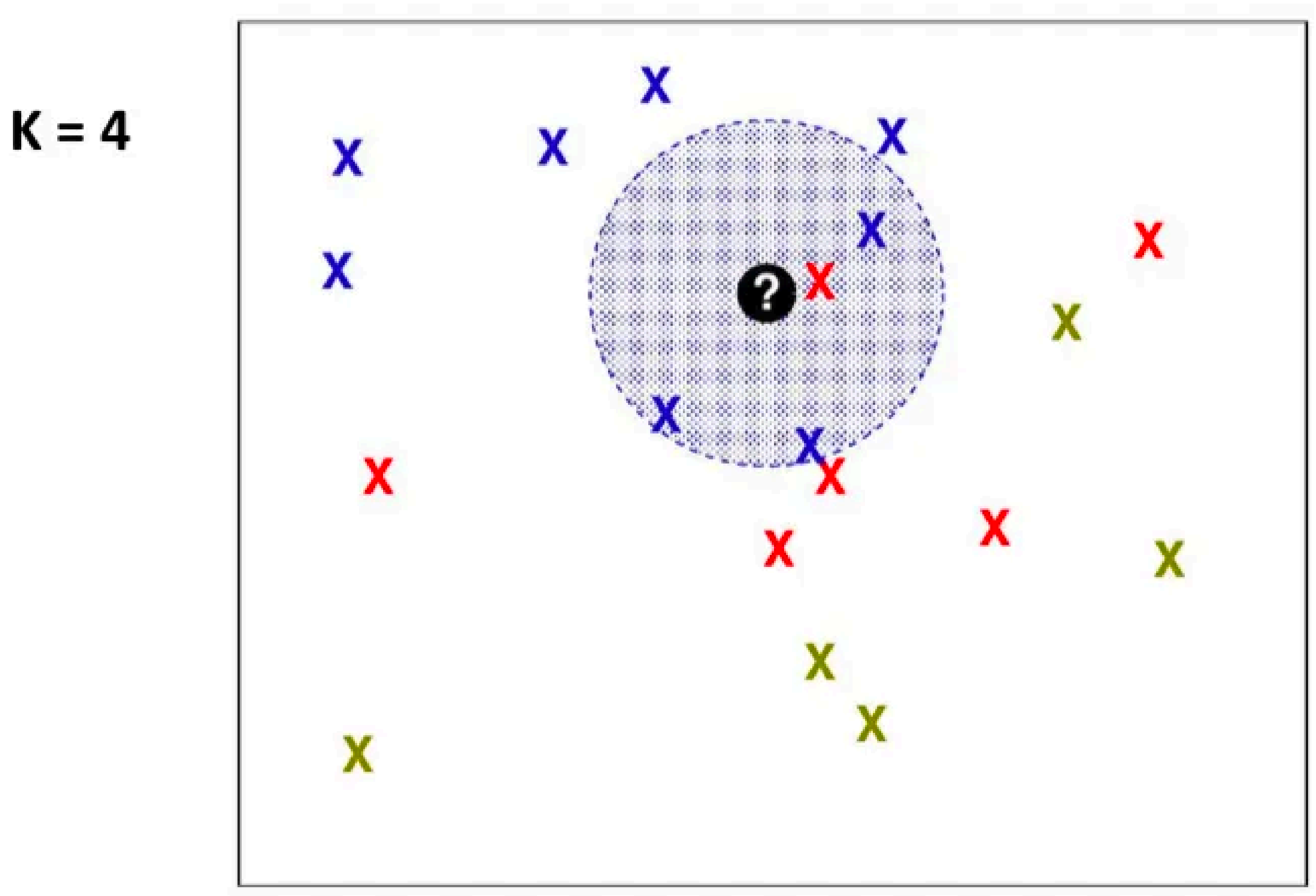
Case Study

KNN

K-Nearest Neighbour

Classify test instance by:

1. Find the k nearest data points from the test instance
2. Assign the test instance by the majority class of the k points



K-Nearest Neighbour

2. Assign the test instance by the **majority class** of the k points

Variations: Assign by the highest-weighted class (sum k point weights)

1. **Inverse linear distance**: $w_j = \frac{d_{max} - d_j}{d_{max} - d_{min}}$

d_{min} = d from nearest neighbour, d_{max} = d from furthest

2. **Inverse distance**: $w_j = \frac{1}{d_j + \epsilon}$

Q1:

Classify the test instances using 1-NN and 3-NN with the three distance measures.

For 3-NN, consider both majority vote and weighted voting (cosine similarity can be weighted by simply summing the similarities of the 3 neighbours).

Training set:

APPLE	IBM	LEMON	SUN	CLASS
4	0	1	1	fruit
5	0	5	2	fruit
2	5	0	0	computer
1	2	1	7	computer

Test set:

APPLE	IBM	LEMON	SUN	CLASS
2	0	3	1	?
1	2	1	0	?

Q1:

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Test instance 1 (E): **k=1, Euclidean**

1. Find closest k = 1 neighbour

$$\sum_{i=1}^{n}(a_i - b_i)^2$$ <- a = A features, b = E features

= (4-2)^2 + (0-0)^2 + (1-3)^2 + (1-1)^2

= 2^2 + 0 + (-2)^2 + 0

= 8

→ d(A,E) = sqrt(8) ~= 2.828

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Q1:

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Test instance 1 (E): **k=1, Euclidean**

1. Find closest k = 1 neighbour

$$\sum_{i=1}^{n}(a_i - b_i)^2$$ **<- a = B features, b = E features**

= (5-2)^2 + (0-0)^2 + (5-3)^2 + (2-1)^2

= 3^2 + 0 + (2)^2 + 1^2

= 9 + 4 + 1 = 14

→ d(B,E) = sqrt(14) ~= 3.742

Q1:

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Test instance 1 (E): **k=1, Euclidean**

1. Find closest k = 1 neighbour

$$\sum_{i=1}^{n}(a_i - b_i)^2$$ **<- a = C features, b = E features**

= (2-2)^2 + (5-0)^2 + (0-3)^2 + (0-1)^2

= 0 + 5^2 + (-3)^2 + (-1)^2

= 25 + 9 + 1 = 35

→ d(C,E) = sqrt(35) ~= 5.916

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Q1:

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Test instance 1 (E): **k=1, Euclidean**

1. Find closest k = 1 neighbour

$$\sum_{i=1}^{n}(a_i - b_i)^2$$ <- a = D features, b = E features

= (1-2)^2 + (2-0)^2 + (1-3)^2 + (7-1)^2

= (-1)^2 + 2^2 + (-2)^2 + (6)^2

= 1 + 4 + 4 + 36 = 45

→ d(D,E) = sqrt(45) ~= 6.708

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Q1:

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Test instance 1 (E): **k=1, Euclidean**

1. Find closest k = 1 neighbour

→ **d(A,E) = sqrt(8) ~= 2.828** **closest!**

→ **d(B,E) = sqrt(14) ~= 3.742**

→ **d(C,E) = sqrt(35) ~= 5.916**

→ **d(D,E) = sqrt(45) ~= 6.708**

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

- **Assign test instance E the class of the closest neighbour → class = "fruit"**

Q1:

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Test instance 1 (E):

- **k=3, Euclidean, inverse distance**

1. Find closest k = 3 neighbours

closest 3!

→ d(A,E) = sqrt(8) ~= 2.828

→ d(B,E) = sqrt(14) ~= 3.742

→ d(C,E) = sqrt(35) ~= 5.916

→ d(D,E) = sqrt(45) ~= 6.708

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Q1:

$w_j = \frac{d_{max} - d_j}{d_{max} - d_{min}}$

d_{min} = d from nearest neighbour, d_{max} = d from furthest

→ d(A,E) = sqrt(8) ~= 2.828

→ d(B,E) = sqrt(14) ~= 3.742

→ d(C,E) = sqrt(35) ~= 5.916

Test instance 1 (E):

- **k=3, Euclidean, inverse distance**

2. Calculate weighted inverse distance of the k=3 points

- w_A = (d_max - d_A) / (d_max - d_min)
  = (5.916 - 2.828) / (5.916 - 2.828) = 1
- w_B = (d_max - d_B) / (d_max - d_min)
  = (5.916 - 3.742) / (5.916 - 2.828) = 0.704
- w_C = (d_max - d_C) / (d_max - d_min)
  = (5.916 - 5.916) / (5.916 - 2.828) = 0

Q1:

$w_j = \frac{d_{max} - d_j}{d_{max} - d_{min}}$

d_{min} = d from nearest neighbour, d_{max} = d from furthest

→ d(A,E) = sqrt(8) ~= 2.828

→ d(B,E) = sqrt(14) ~= 3.742

→ d(C,E) = sqrt(35) ~= 5.916

Test instance 1 (E):

- **k=3, Euclidean, inverse distance**

3. Assign label by the class w/ heightest weight

- w_A = 1, w_B = 0.704, w_C = 0
- w_{fruits} = w_A + w_B = 1 + 0.704 = 1.704
- w_{computer} = w_C = 0
- 1.704 > 0 → class = "fruit"

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Q1:

$$d(A,B) = \sum_{i=1}^{n} |a_i - b_i|$$

Test instance 1 (E):

- **k=3, Manhattan, inverse linear distance**

1. Find closest k = 3 neighbours

d(A,E) = |4-2| + |0-0| + |1-3| + |1-1|
= 2 + 0 + 2 + 0 = 4

d(B,E) = |5-2| + |0-0| + |5-3| + |2-1|
= 3 + 0 + 2 + 1 = 6

d(C,E) = |2-2| + |5-0| + |0-3| + |0-1|
= 0 + 5 + 3 + 1 = 9

d(D,E) = |1-2| + |2-0| + |1-3| + |7-1|
= 1 + 2 + 2 + 6 = 11

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Q1:

$$w_j = \frac{1}{d_j + \epsilon}$$

Test instance 1 (E):

- **k=3, Manhattan, inverse linear distance**

2. Calculate weighted inverse linear distance of the k=3 points

d(A,E) = 4, d(B,E) = 6, d(C,E) = 9

w_A = 1 / (d_A + e)
= 1 / (4 + e)

w_B = 1 / (d_B + e)
= 1 / (6 + e)

w_C = 1 / (d_C + e)
= 1 / (9 + e)

Q1:

$$w_j = \frac{1}{d_j + \epsilon}$$

- Here, e is not given.
- Assume e → 0

Test instance 1 (E):

- **k=3, Manhattan, inverse linear distance**

3. Assign label by the class w/ heightest weight

- w_A = 1 / (4 + e), w_B = 1 / (6 + e), w_C = 1 / (9 + e)
- w_{fruit} = w_A + w_B = 1 / (4 + e) + 1/ (6 + e) ~= 0.621
- w_{computer} = w_C = 1 / (9 + e) = 0.169
- 0.621 > 0.169 → class = "fruit"

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Q1:

$$cos(A,B) = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test instance 1 (E):

- **k=3, Cosine similarity, majority vote**

1. Find closest k = 3 neighbours

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

$\sum_i a_i b_i$ = (A_{apple}*E_{apple}) + (A_{ibm} *E_{ibm}) + ...

= (4*2) + (0*0) + (1*3) + (1*1)

= 8 + 0 + 3 + 1 = 12

$\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}$ = sqrt(A_{apple}^2 + A_{ibm}^2 + ...) * sqrt(E_{apple}^2 + E_{ibm}^2 + ...)

= sqrt(4^2 + 0^2 + 1^2 + 1^2) * sqrt(2^2 + 0^2 + 3^2 + 1^2)

= sqrt(16 + 0 + 1 + 1) * sqrt(4 + 0 + 9 + 1) = sqrt(18 * 14)

→ cos(A,E) = 12 / sqrt(18*14) ~= 0.756

Q1:

$$cos(A,B) = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Test instance 1 (E):

- **k=3, Cosine similarity, majority vote**

1. Find closest k = 3 neighbours

$\sum_i a_i b_i$ = (B_{apple}*E_{apple}) + (B_{ibm} *E_{ibm}) + ...

= (5*2) + (0*0) + (5*3) + (2*1)

= 10 + 0 + 15 + 2 = 27

$\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}$ = sqrt(B_{apple}^2 + B_{ibm}^2 + ...) * sqrt(E_{apple}^2 + E_{ibm}^2 + ...)

= sqrt(5^2 + 0^2 + 5^2 + 2^2) * sqrt(2^2 + 0^2 + 3^2 + 1^2)

= sqrt(25 + 0 + 25 + 4) * sqrt(4 + 0 + 9 + 1) = sqrt(54 * 14)

→ cos(B,E) = 27 / sqrt(54*14) ~= 0.982

Q1:

$$cos(A,B) = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

Test instance 1 (E):

- **k=3, Cosine similarity, majority vote**

1. Find closest k = 3 neighbours

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

$\sum_i a_i b_i$ = (C_{apple}*E_{apple}) + (C_{ibm} *E_{ibm}) + ...

= (2*2) + (5*0) + (0*3) + (0*1)

= 4 + 0 + 0 + 0 = 4

$\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}$ = sqrt(C_{apple}^2 + C_{ibm}^2 + ...) * sqrt(E_{apple}^2 + E_{ibm}^2 + ...)

= sqrt(2^2 + 5^2 + 0^2 + 0^2) * sqrt(2^2 + 0^2 + 3^2 + 1^2)

= sqrt(4 + 25 + 0 + 0) * sqrt(4 + 0 + 9 + 1) = sqrt(29 * 14)

→ cos(C,E) = 4 / sqrt(29*14) ~= 0.199

Q1:

$$cos(A,B) = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test instance 1 (E):

- **k=3, Cosine similarity, majority vote**

1. Find closest k = 3 neighbours

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

$\sum_i a_i b_i$ = (D_{apple}*E_{apple}) + (D_{ibm} *E_{ibm}) + ...
= (1*2) + (2*0) + (1*3) + (7*1)
= 2 + 0 + 3 + 7 = 12

$\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}$ = sqrt(D_{apple}^2 + D_{ibm}^2 + ...) * sqrt(E_{apple}^2 + E_{ibm}^2 + ...)
= sqrt(1^2 + 2^2 + 1^2 + 7^2) * sqrt(2^2 + 0^2 + 3^2 + 1^2)
= sqrt(1 + 4 + 1 + 49) * sqrt(4 + 0 + 9 + 1) = sqrt(55 * 14)

→ cos(D,E) = 12 / sqrt(55*14) ~= 0.432

Q1:

$$cos(A,B) = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test instance 1 (E):

- **k=3, Cosine similarity, majority vote**

1. Find closest k = 3 neighbours

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

- **cos(A,E) = 12 / sqrt(18*14) ~= 0.756**
- **cos(B,E) = 27 / sqrt(54*14) ~= 0.982**
- **cos(C,E) = 4 / sqrt(29*14) ~= 0.199**
- **cos(D,E) = 12 / sqrt(55*14) ~= 0.432**

similarity → higher the score the more similar (closer) it is!

Q1:

$$cos(A,B) = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

Test instance 1 (E):

- **k=3, Cosine similarity, majority vote**

2. Assign by the **majority class** of the k points

- **cos(A,E) = 12 / sqrt(18*14) ~= 0.756**
- **cos(B,E) = 27 / sqrt(54*14) ~= 0.982**
- **cos(D,E) = 12 / sqrt(55*14) ~= 0.432**

- **2 * fruit, 1 * computer**
- **→ majority class = "fruit"**
- **→ assign instance E's class as "fruit"**

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Q1:

$$cos(A,B) = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

Test instance 1 (E):

- **k=3, Cosine similarity, weighted voting**

2. Assign by the **weighted vote** of the k points

- **cos(A,E) = 12 / sqrt(18*14) ~= 0.756**
- **cos(B,E) = 27 / sqrt(54*14) ~= 0.982**
- **cos(D,E) = 12 / sqrt(55*14) ~= 0.432**

- **w_{fruit} = 0.983 + 0.756 = 1.739**
- **w_{computer} = 0.432**

- **1.739 > 0.432 → class = "fruit"**

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Q1:

Fill in the rest of the table!

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

$$d(A,B) = \sum_{i=1}^{n}|a_i - b_i|$$

$$w_j = \frac{1}{d_j + \epsilon}$$

$$cos(A,B) = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$

$$w_j = \frac{d_{max} - d_j}{d_{max} - d_{min}}$$

d_{min} = d from nearest neighbour, d_{max} = d from furthest

Training set:

APPLE	IBM	LEMON	SUN	CLASS	
4	0	1	1	fruit	A
5	0	5	2	fruit	B
2	5	0	0	computer	C
1	2	1	7	computer	D

Test set:

APPLE	IBM	LEMON	SUN	CLASS	
2	0	3	1	?	E
1	2	1	0	?	F

Q1:

Test instance 1:

Measure	K	Weight	Prediction
Euclidean	1	N/A	fruit
Euclidean	3	Majority vote	fruit
Euclidean	3	Inverse dist	fruit
Euclidean	3	Inverse linear dist	fruit
Manhattan	1	N/A	fruit
Manhattan	3	Majority vote	fruit
Manhattan	3	Inverse dist	fruit
Manhattan	3	Inverse linear dist	fruit
Cosine	1	N/A	fruit
Cosine	3	Majority vote	fruit
Cosine	3	Sum	fruit

Test instance 2:

Measure	K	Weight	Prediction
Euclidean	1	N/A	computer
Euclidean	3	Majority vote	fruit
Euclidean	3	Inverse dist	fruit
Euclidean	3	Inverse linear dist	computer
Manhattan	1	N/A	computer
Manhattan	3	Majority vote	computer
Manhattan	3	Inverse dist	computer
Manhattan	3	Inverse linear dist	computer
Cosine	1	N/A	fruit
Cosine	3	Majority vote	computer
Cosine	3	Sum	fruit

Q2:

How does the k value in k-NN algorithm affect the decision boundary between classes?

- K increase:
  - More neighbours → smoother decision boundary → less sensitive
- Issues with big K:
  - Too big → may miss rare classes
  - May become 0-R / majority class classifier
- Need to find balance!

01

Concept

02

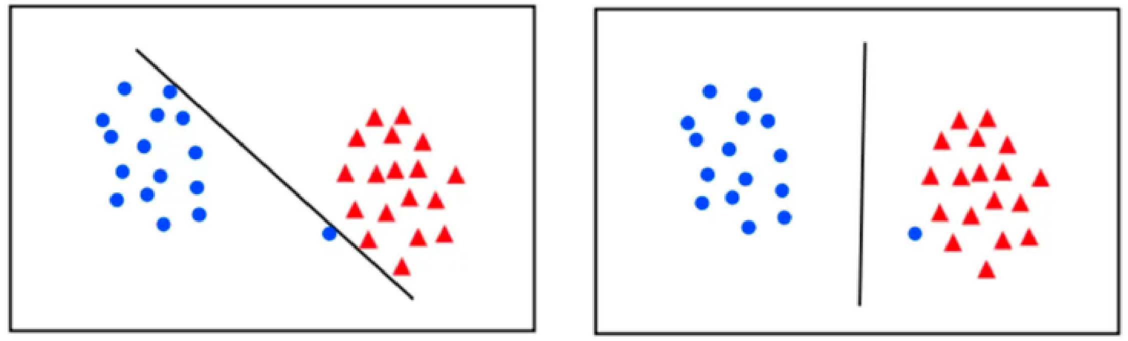
Questions

SVM

Support Vector Machine

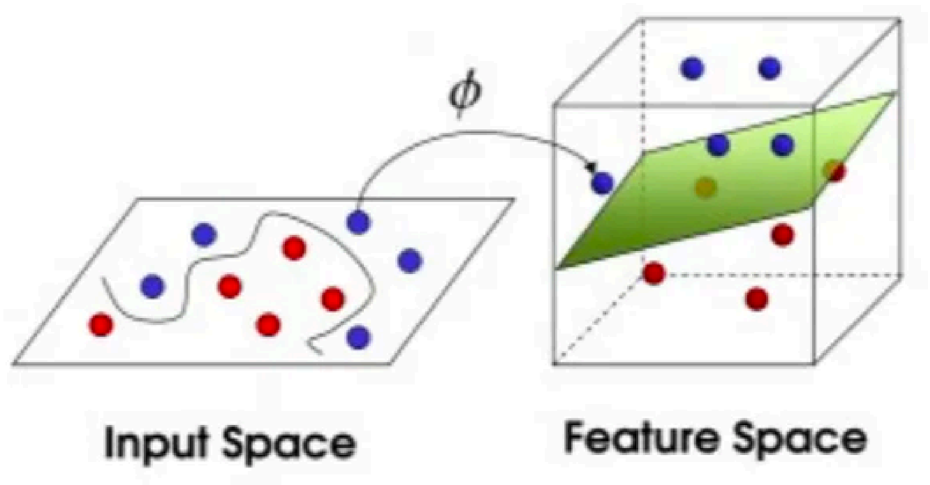
- Hyperplane-based classifier
  - Tries to find the hyperplane that best separates the classes
  - Hyperplane = the high-dimensional equivalent of a line

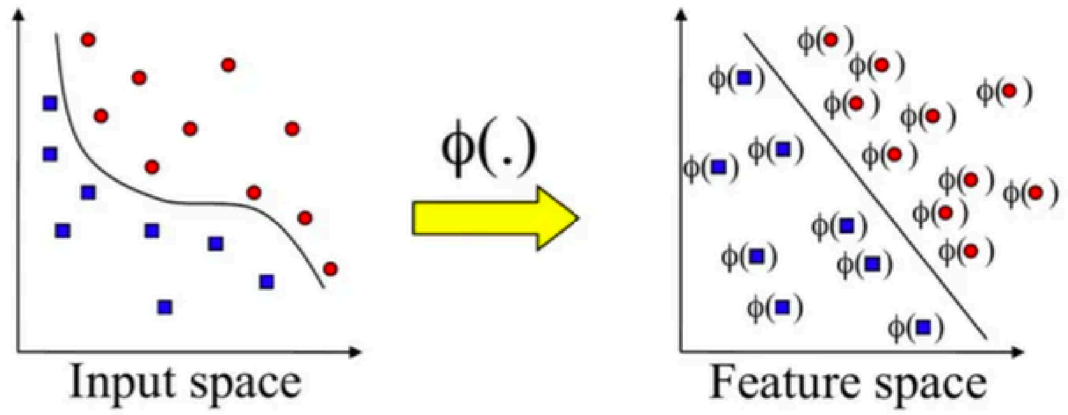
- Best separator = hyperplane w/ maximum margin
  - Maximise distance b/w closest point & boundary
  - Soft margins: allow some data points to violate the boundary



Support Vector Machine

- Non-linear SVM:
  - Transform data to a new feature space
  - Find hyperplane separating two classes in the new space





Kernel Trick

- Kernel trick:
  - Compute inner products directly instead of computing high-dimensional coordinates

- Kernel functions:
  - Functions to transform data efficiently

Common kernel functions:

1. **Linear Kernel:**

$$K(x_i, x_j) = x_i^T x_j$$

2. **Polynomial Kernel:**

$$K(x_i, x_j) = (x_i^T x_j + \theta)^d$$

3. **Radial Basis Kernel:**

$$K(x_i, x_j) = exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$$

Q3(a):

What does it mean for a classification dataset to be "**linearly separable**"?

- Linearly separable:
  - Can completely separate the classes with a single hyperplane

Q3(b):

If a dataset isn't linearly separable, an SVM learner has two major options. What are they, and why might we prefer one to the other?

1. **Soft margin**:
   - Allow some mistakes to be made w/ penalty to find better boundary
   - Better choice if largely linearly separable → kernel methods too powerful for few outliers, may generate a spurious transformation
2. **Kernel methods**:
   - Transform the data into a higher-dimensional space
   - Better if the instance topology isn't linear, but polynomial / circular…

Q4:

Unlike other geometric methods such as K-NN, SVMs work better with large attribute sets. Why might this be true?

- Do all attributes contribute equally to the label prediction?
- SVM hyperplanes:
    - Implicitly capture relationships b/w features w/o feature engineering
    - Can effectively learn from a large number of features w/o prior knowledge
- SVMs only need support vectors (small subset of points) to define boundary
    - Sparse representation → efficient & scalable w.r.t no. of input features.*

Q5:

How do changes in data points affect the decision boundary of an SVM?

- Decision boundary is determined by the support vectors
  - Support vectors = data points closest to the boundary
  - If data point not support vector, doesn’t affect hyperplane

- Hence, **change support vectors → change decision boundary**
  - + / - a support vector → decision boundary may shift / rotate

Q5:

How do changes in data points affect the decision boundary of an SVM?

Mathematical explanation of SVM:

- Solve contrained optimisation problem using Lagrange multipliers
  a. Introduce a value a_i for each contraint (training data point)
  b. Solve a_i
  c. Eventually, most a_i = 0; non-zero values → support vectors
- As such, non-support vectors have a_i = 0, don’t contribute to solution

Q5:

How do changes in data points affect the decision boundary of an SVM?

Solve by minimizing w.r.t. <u>w, b</u> and maximizing w.r.t. <u>a_i (\lambda_i)</u>

The Lagrangian

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

KKT conditions:

* Feasibility: $y_i\big((\boldsymbol{w}^*)'\boldsymbol{x}_i + b^*\big) - 1 \geq 0$ for $i = 1, \dots, n$
* Feasibility: $\lambda_i^* \geq 0$ for $i = 1, \dots, n$
* Complementary slackness: $\lambda_i^*\big(y_i\big((\boldsymbol{w}^*)'\boldsymbol{x}_i + b^*\big) - 1\big) = 0$
* Stationarity: $\nabla_{\boldsymbol{w},b}\mathcal{L}(\boldsymbol{w}^*, b^*, \boldsymbol{\lambda}^*) = \mathbf{0}$

If nothing makes sense: https://www.youtube.com/watch?v=bM4_AstaBZo&t=512s

Q6(a):

What is the value of slack variables for data points that are correctly classified in SVMs?

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N} \xi_i$$

$$\text{subject to } y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) + \xi_i - 1 \geq 0,$$

$$\xi_i \geq 0, \forall\, i \in \{1,2,\ldots,N\}$$

- Slack variables in soft-margin SVM allows for some classification errors
    - Value indicates how much a data point violates the margin
    - e.g. For correct classification, slack variable = 0 → doesn't contribute
    - e.g. For incorrect instances inside the margin, slack variable = 1
    - For incorrect instances outside the margin?

Q6(b):

What should the slack penalty C be to make a soft-margin SVM function as a hard-margin SVM?

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

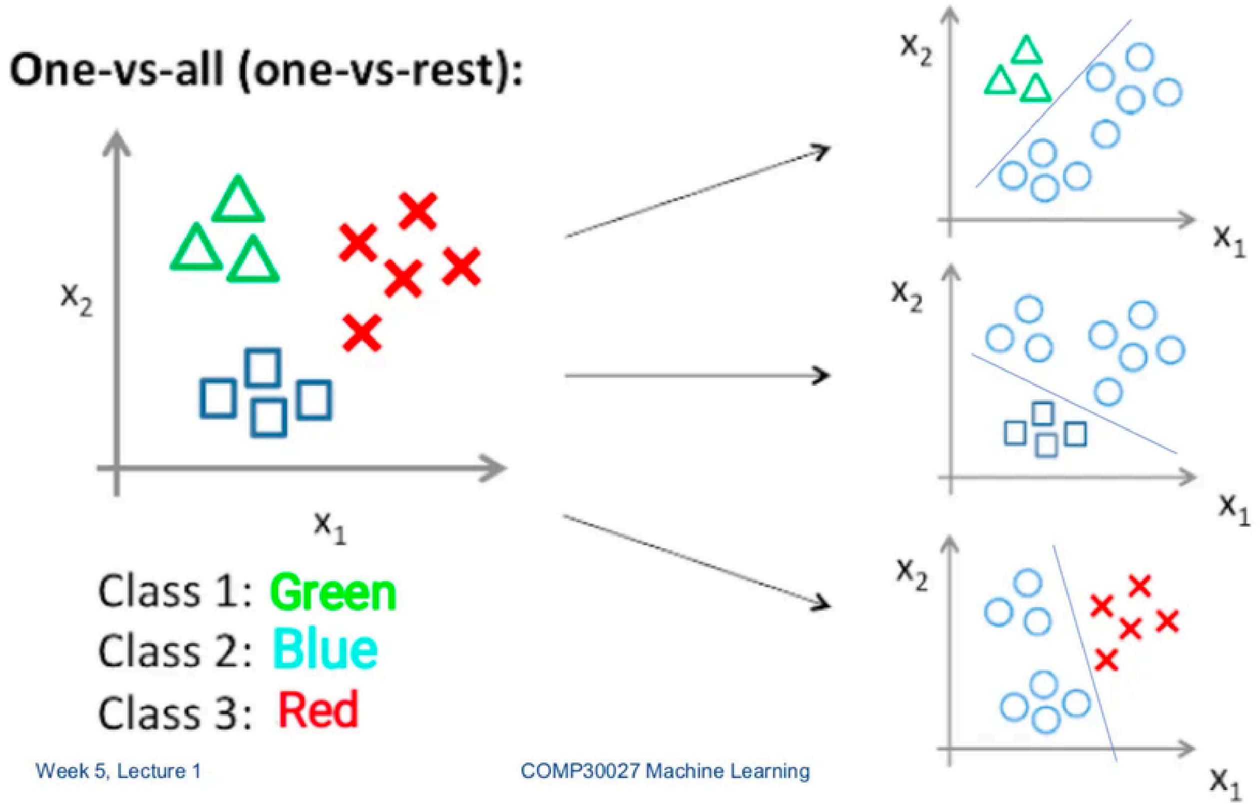
$$\text{subject to } y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) + \xi_i - 1 \geq 0,$$

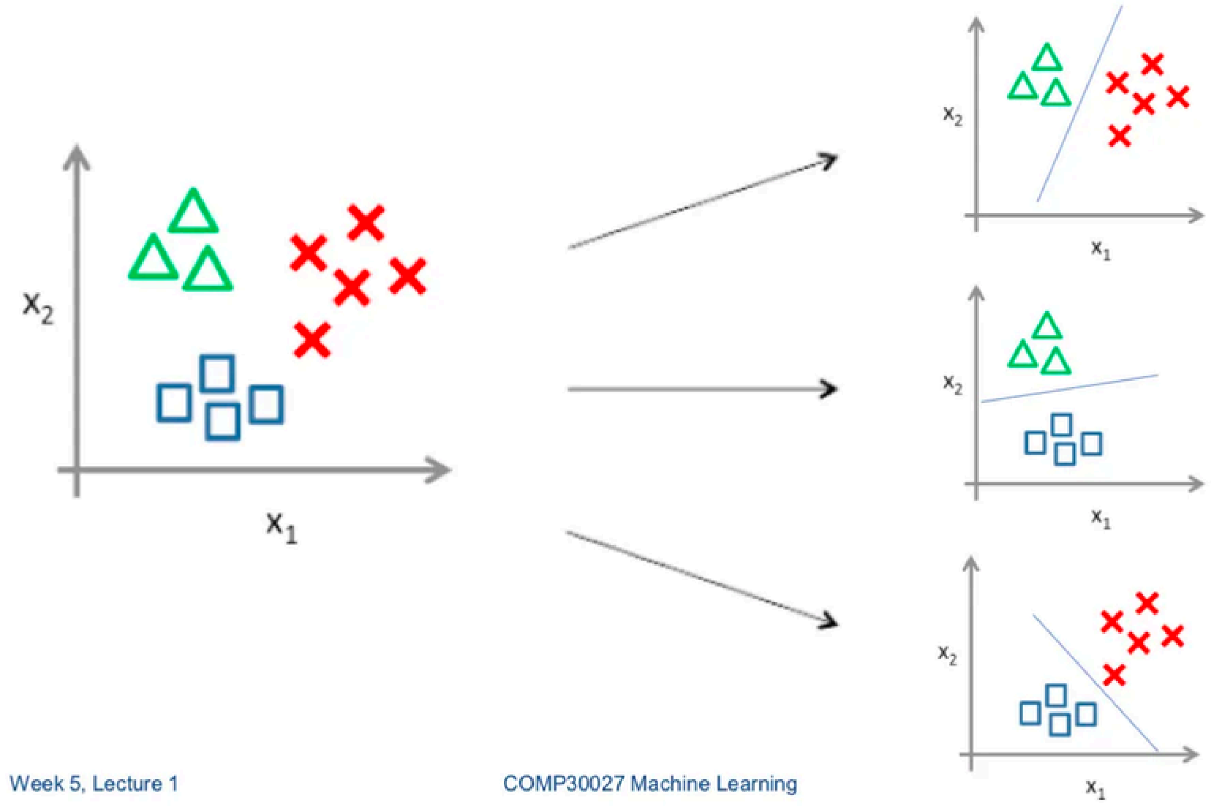
$$\xi_i \geq 0, \forall\, i \in \{1,2,\dots,N\}$$

- C: trade-off b/w maximising the margin & minimising the training error
  - If C increase, weight of training error higher → try to minimise error, so the margin will be smaller
  - If C decrease, weight of second term lower → try to get larger margin
- Hard SVM = no mistakes allowed / penalty rate = ∞
  - Set slack penalty C to be very large

Q6(b):

How many binary classifiers are needed to classify a dataset with 4 classes using one-vs-one method?





- One-vs-all: one classifier to separate one class from the rest
- One-vs-one: one classifier per pair of classes

Q6(b):

How many binary classifiers are needed to classify a dataset with 4 classes using one-vs-one method?

- 4 classes:
  - 4 choose 2 = $c(c-1)/2$
  - = (4×3)/2 = 6 unique pairs of classes