k-nearest neighbours

MACHINE LEARNING

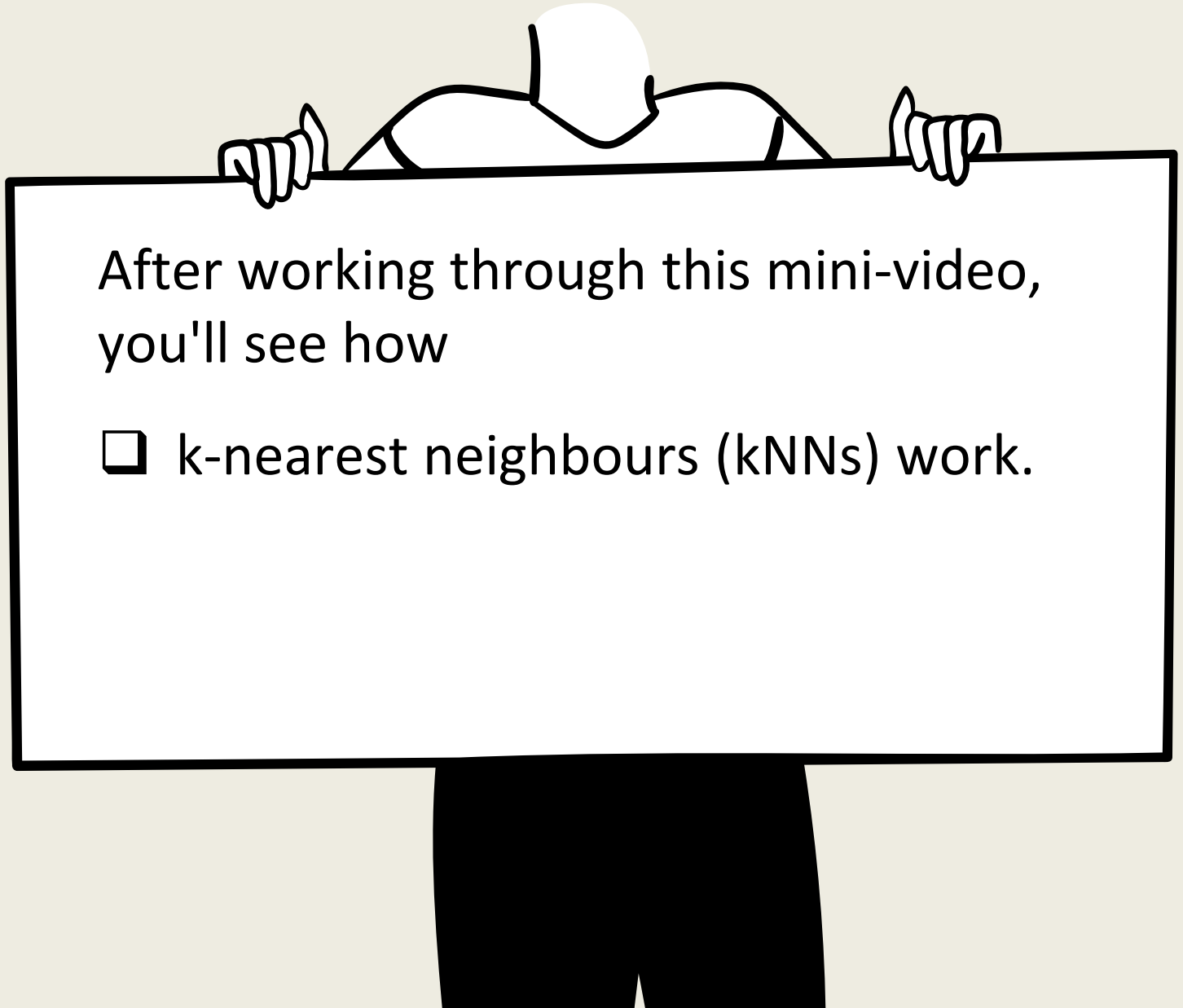
Dr. Temitayo Olugbade

UNIVERSITY
OF SUSSEX

Recall from Week 1

1. The most basic element of **machine learning** is a **model** that learns from data.
2. Training a ML model involves optimizing the model parameters based on a **loss function**.

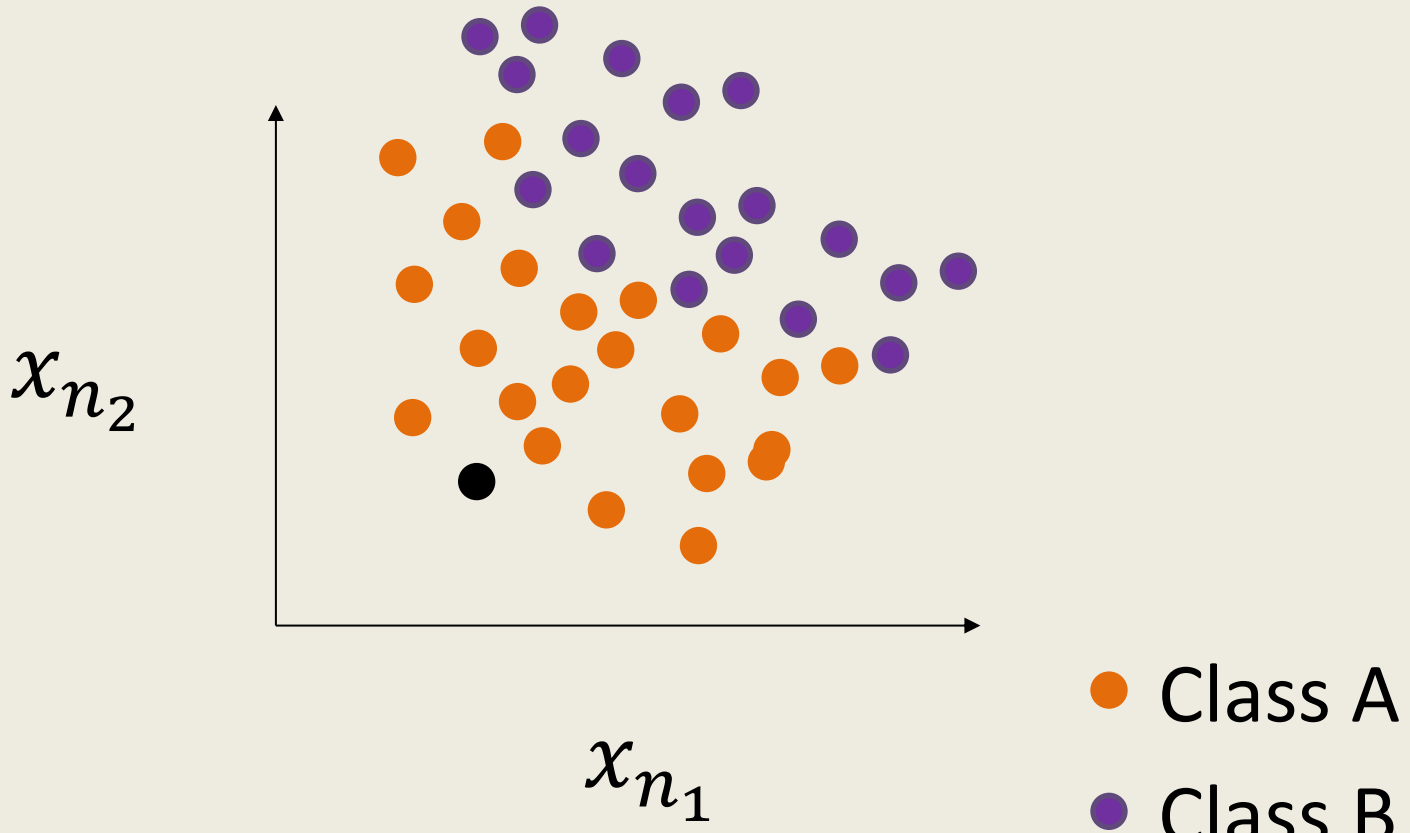
Learning outcome



After working through this mini-video, you'll see how

- ❑ k-nearest neighbours (kNNs) work.

A toy data for illustration



k-nearest neighbours (kNN)

- Consider that you can and desire to automatically obtain some answer $\boldsymbol{y}_l$
- You would need to use N training data instances ~~to find a model/function $f(\cdot)$ such that…~~
- With kNN, you use the class(es) of *k* nearest neighbours of $\boldsymbol{x}_i$ to determine its output $\hat{\boldsymbol{y}}_i$

No explicit model is defined

How a kNN inference works

- **Step 1**

For each given test instance $\boldsymbol{x}_i$, find its ***k*** nearest neighbours based on a **distance metric** $dist(\boldsymbol{x}_i, \boldsymbol{x}_n)\ \forall n = 1, 2, \cdots, N$

- **Step 2**

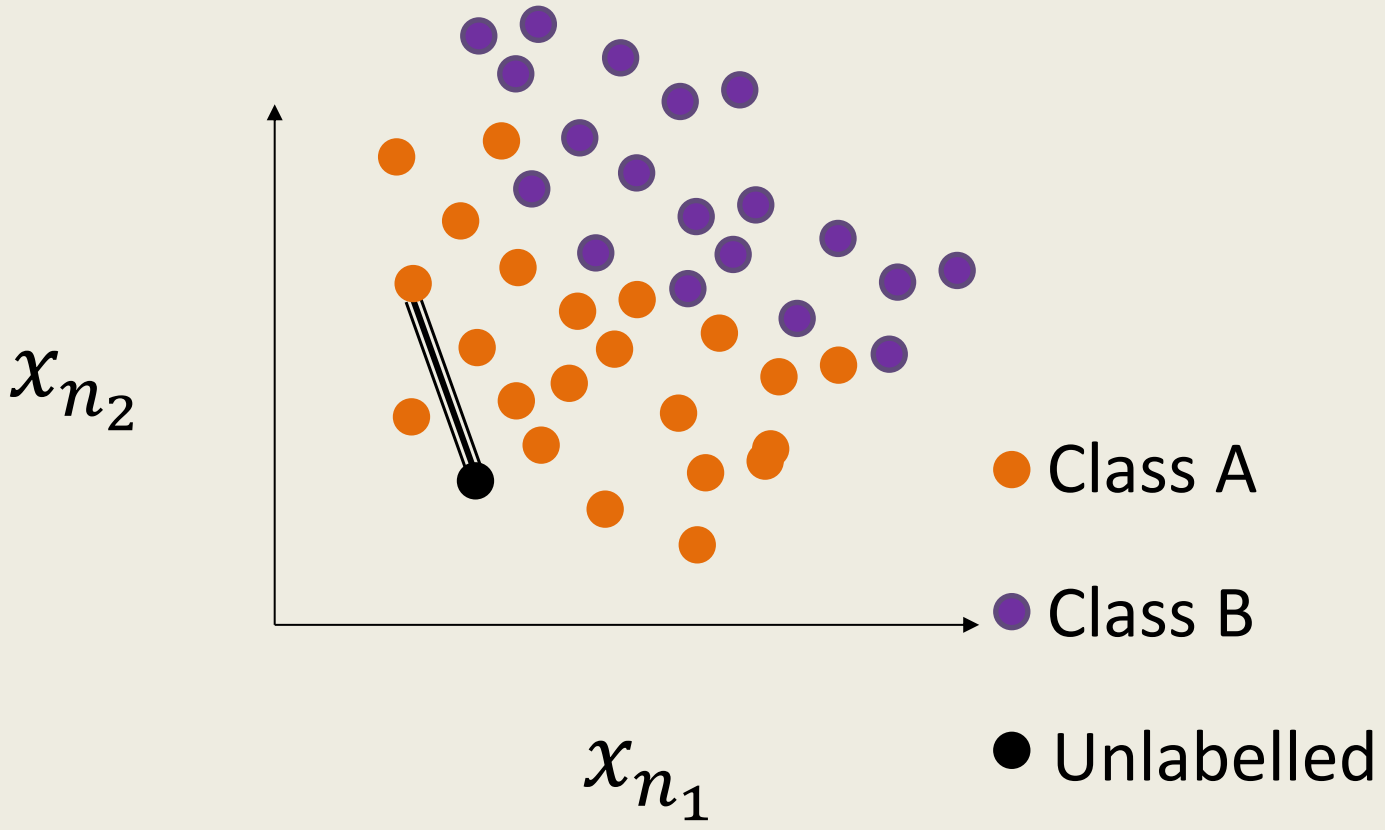
Determine the class $\hat{y}_i$ of $\boldsymbol{x}_i$ from the labels $y_1, y_2, \cdots, y_k$ of these *k* nearest neighbours using a **voting strategy**

How a kNN inference works

- **Step 1**

For each given test instance $\boldsymbol{x}_i$, find its ***k*** nearest neighbours based on a **distance metric** $dist(\boldsymbol{x}_i, \boldsymbol{x}_n)\ \forall n = 1, 2, \cdots, N$

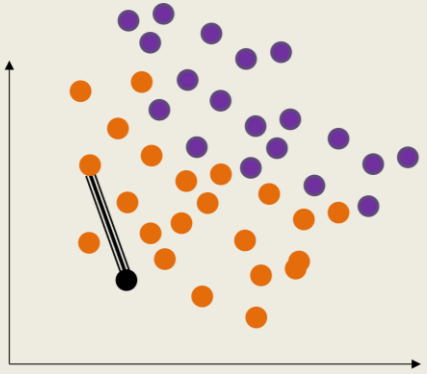
Euclidean distance



$$dist(x_i, x_n) = \sqrt{\sum_{d=1}^{D} \left(x_{i_d} - x_{n_d}\right)^2} = \|x_i - x_n\|$$

see math proof in last slide pages

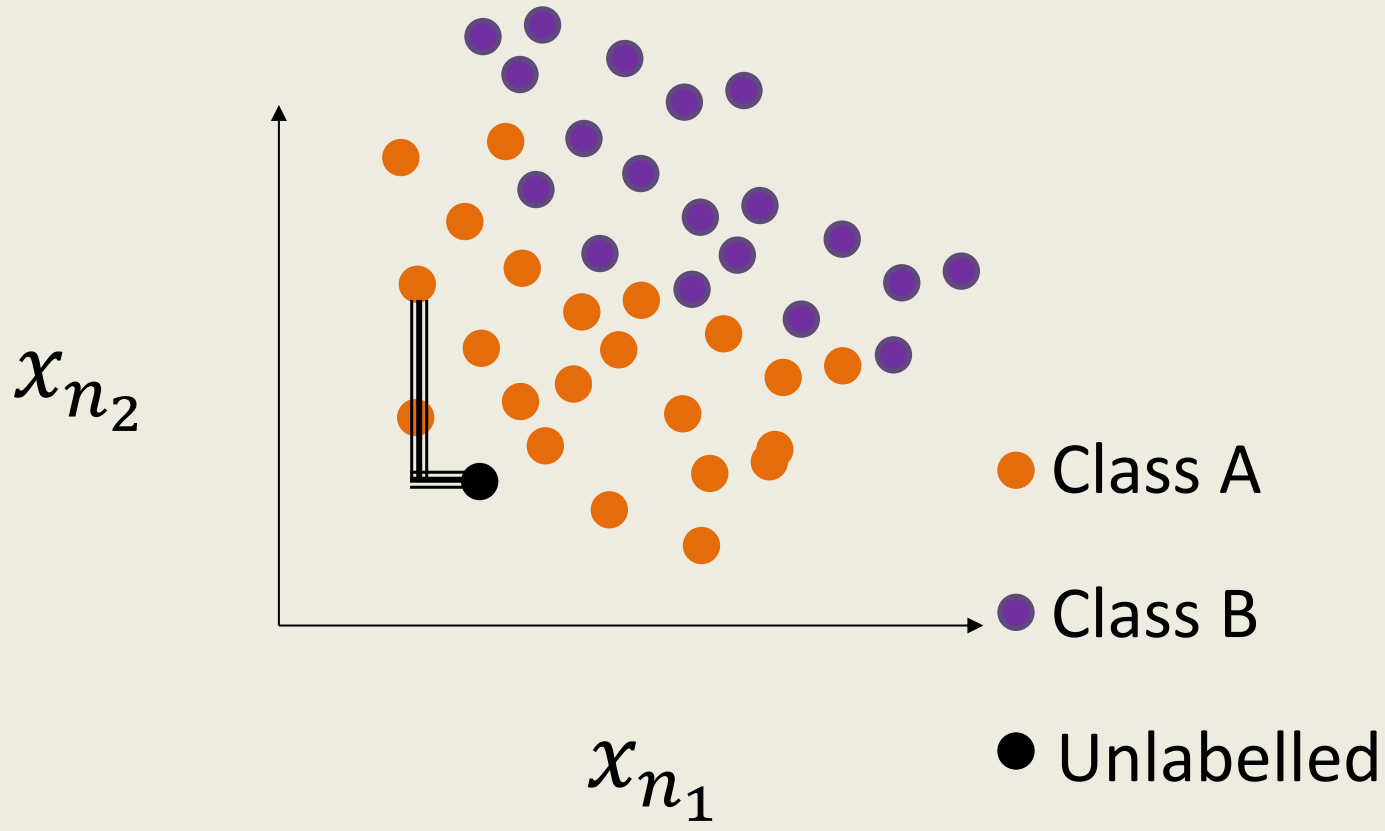
Euclidean distance use cases



Differentiating:

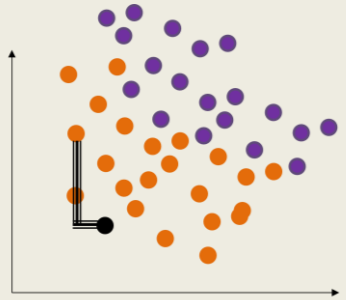
- objects in the sky 
- football fans in a pub
- sediment particles vs floating particles in water

Manhattan (city block) distance



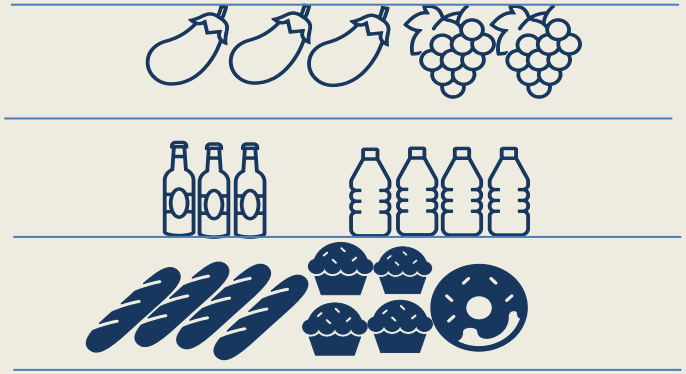
$$dist(x_i, x_n) = \sum_{d=1}^{D} \left| x_{i_d} - x_{n_d} \right| = |x_i - x_n|$$

Manhattan distance use cases



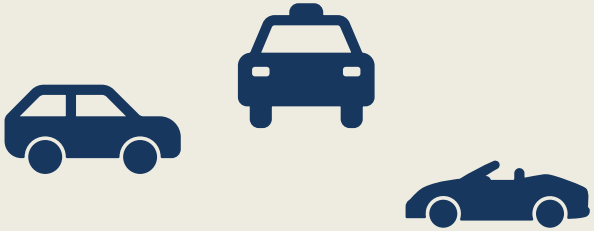
Predicting:

- prices of houses in a region

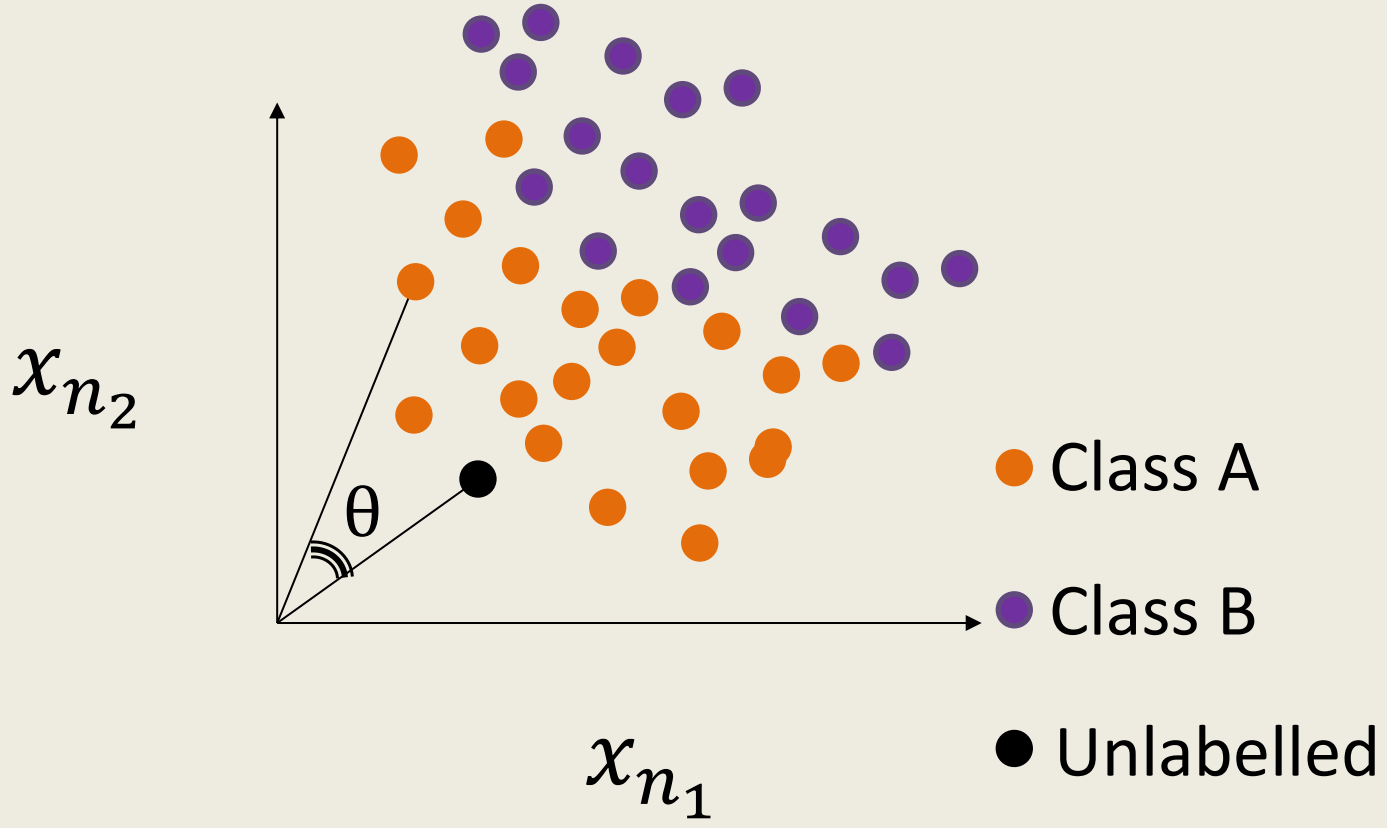


- categories of items in a supermarket

- trajectories of cars in traffic



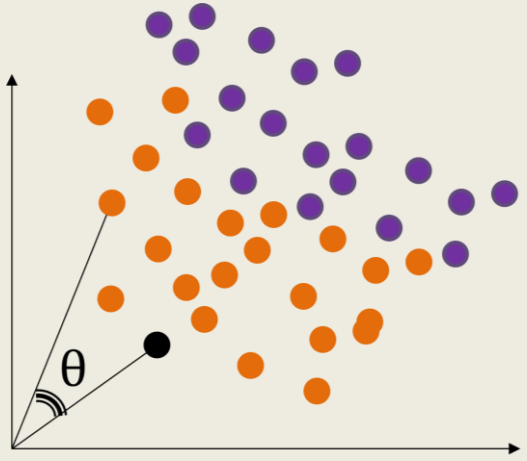
Cosine similarity



$$dist(x_i, x_n) = 1 - \frac{\sum_{d=1}^{D} x_{i_d} \cdot x_{j_d}}{\sqrt{\sum_{d=1}^{D} x_{i_d}{}^2} \cdot \sqrt{\sum_{d=1}^{D} x_{n_d}{}^2}}$$

see math proof in last slide pages

Cosine similarity use cases



Similarity between

- text documents (represented by the number of occurrences of given words)

Other ‘distance’ metrics

- Correlation (trend, e.g. Pearson’s for normally distributed data, Spearman’s otherwise)
  e.g. between house prices over two time periods

- Tchebychev (maximum difference)
  e.g. between the longest subtask for each of two processes

- Metric learning (optimization, e.g. based on Linear Discriminant Analysis, or Generalized Mahalanobis Distance)

How a kNN inference works

- **Step 1**

For each given test instance $\boldsymbol{x}_i$, find its ***k*** nearest neighbours based on a **distance metric** $dist(\boldsymbol{x}_i, \boldsymbol{x}_n)\ \forall n = 1, 2, \cdots, N$

- **Step 2**

Determine the class $\hat{y}_i$ of $\boldsymbol{x}_i$ from the labels $y_1, y_2, \cdots, y_k$ of these *k* nearest neighbours using a **voting strategy**

Voting strategy

- Majority voting (for classification)

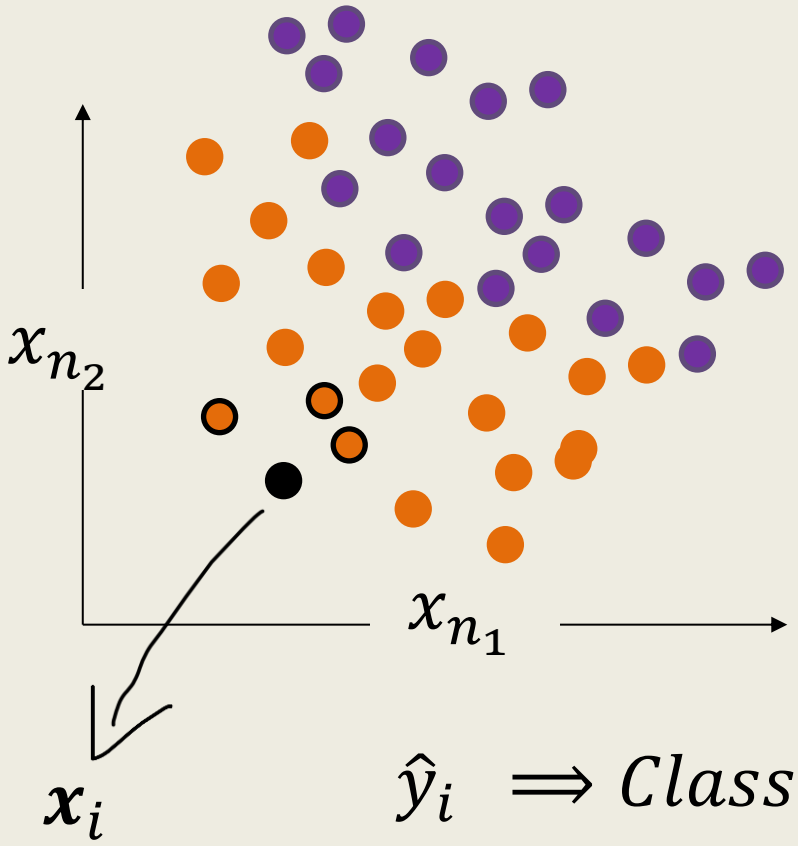
$$\hat{y}_i = \max_c \left\{ \sum_{j=1}^{k} 1_{y_j=c}, \qquad \forall c \in C \right\}$$

- Averaging (for regression)

$$\hat{y}_i = \frac{1}{k} \sum_{j=1}^{k} y_j$$

Majority voting example

Consider number of neighbours *k*=3,

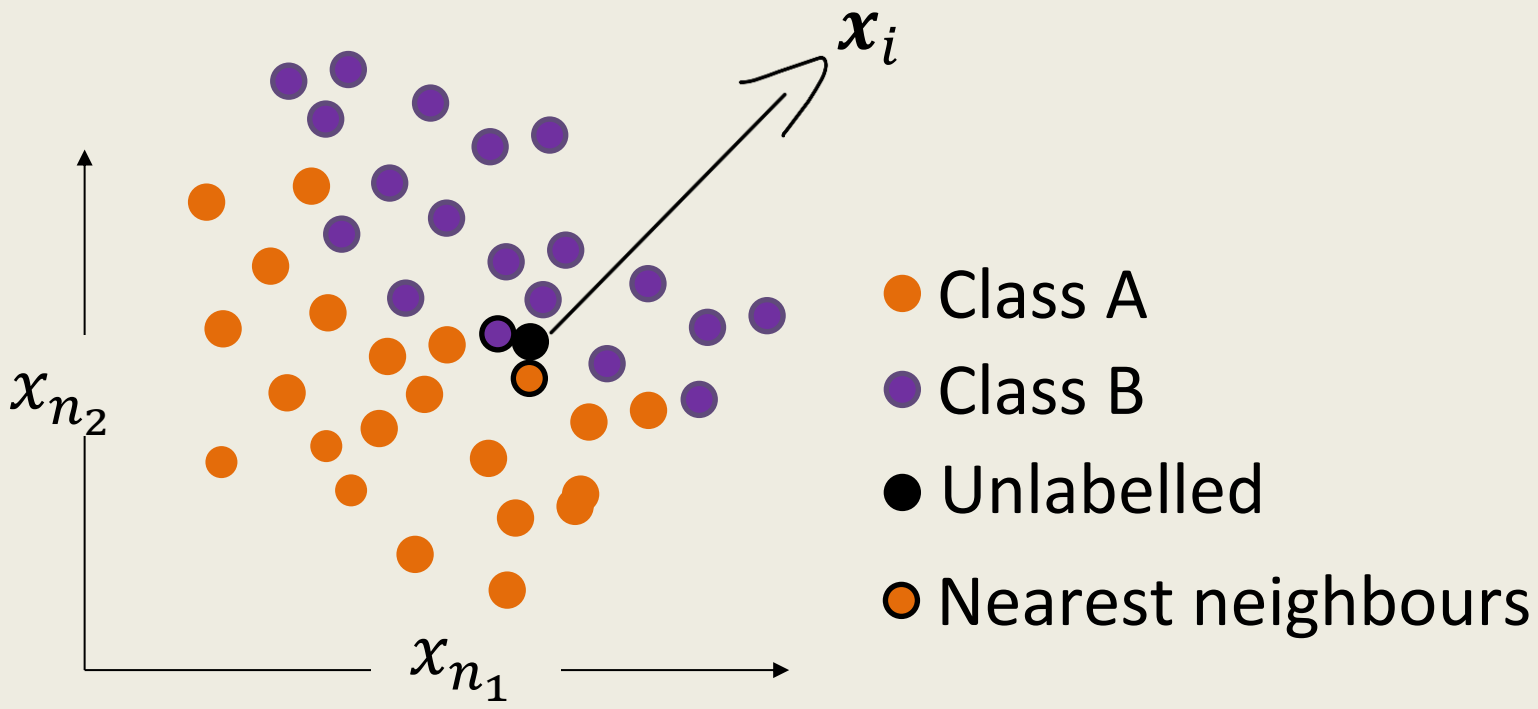


- Class A
- Class B
- Unlabelled
- Nearest neighbours

see math details in last slide pages

Another example

Consider *k*=2



$\hat{y}_i \Longrightarrow$ a tie between Class A & Class B

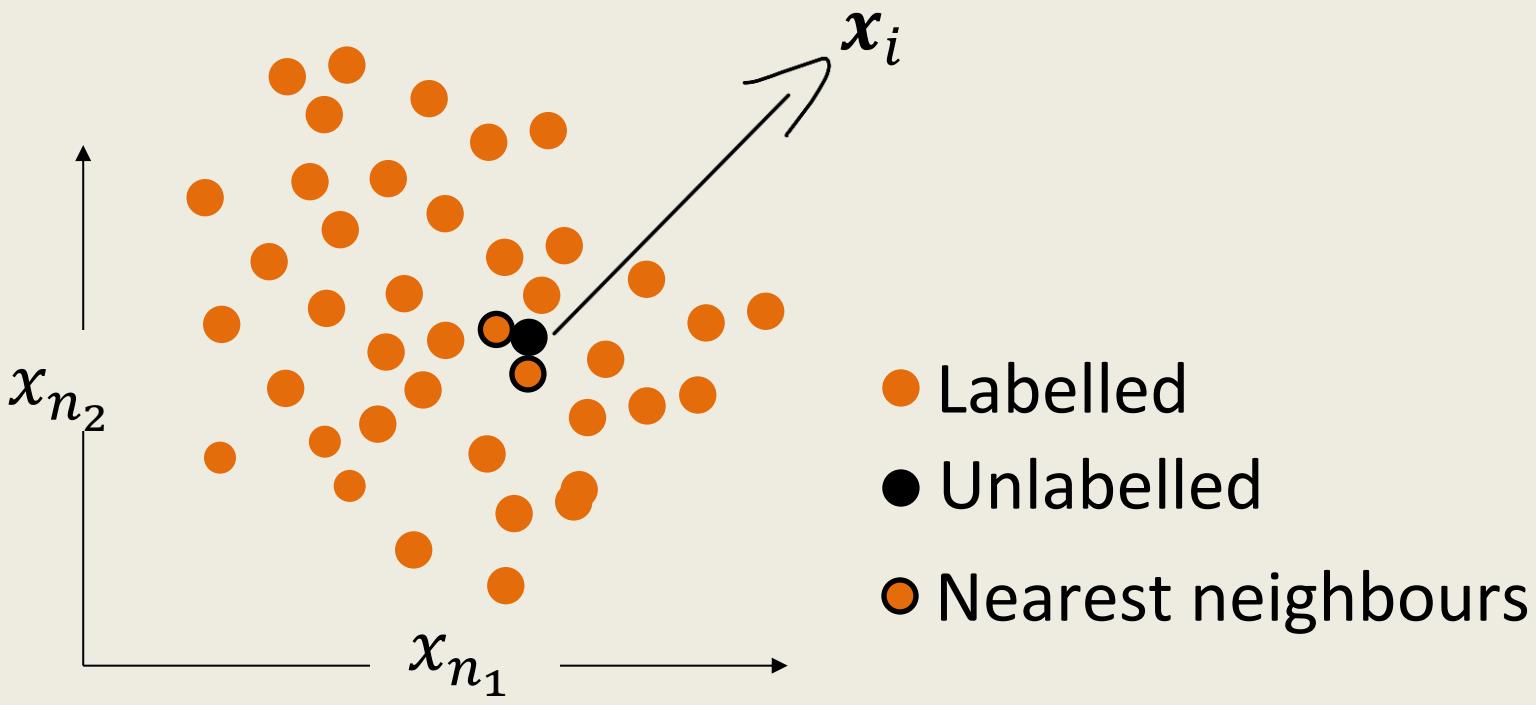
see math details on last slide pages

Averaging example

Consider a regression task with k=2

$$\Rightarrow \hat{y}_i = \frac{1}{k}\sum_{j=1}^{k} y_j$$

$$= \frac{1}{2}\left(y_{n.neighbour1} + y_{n.neighbour2}\right)$$



Other voting strategies

- Distance-weighted voting (for classification)

$$\hat{y}_i = \max_c \left\{ \sum_{j=1}^{k} \frac{1}{d(x_i, x_j)^r} \cdot 1_{y_j = c}, \quad \forall c \in C \right\}, \; r \geq 1$$

- Distance-weighted averaging (for regression)

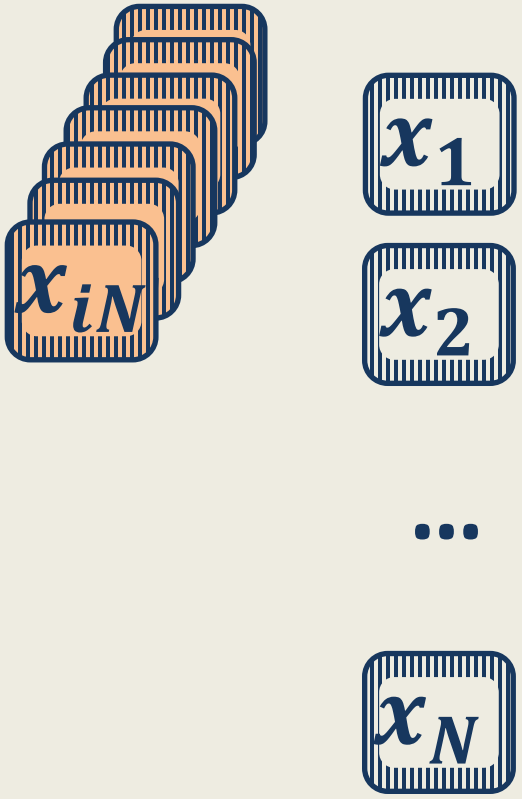
$$\hat{y}_i = \frac{1}{k} \sum_{j=1}^{k} \frac{1}{d(x_i, x_j)^r} \cdot y_j, \; r \geq 1$$

Inference time complexity for kNN

$$N^2 D$$

N for the total number of data instances
D for the total number of features

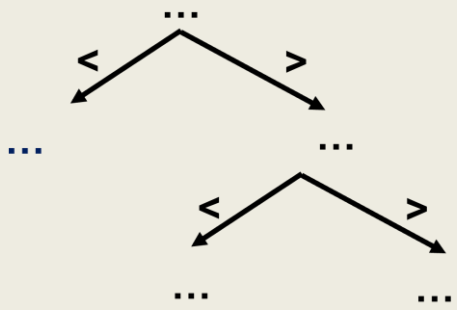
see math details in last slide pages



Strategy for minimizing time complexity

- Indexing for faster search

e.g. k-dimension tree (kd tree)



- kd tree – Create a tree-based indexing of the data
  - Repeatedly partition the data region
  - Similar to creating a decision tree
    - The partition median is the preferred split point
    - Leaf nodes are data instances
    - Sorting of values per feature – time complexity $NlogN$
    - Overall time complexity given D features – $NDlogN$

- kNN search – Goes through the tree from root node
  - Values to the left of a node are smaller than the node
  - Aim is to find the leaf node closest to x_i

Other strategies: Dimensionality reduction

- Time complexity of the ordinary kNN (i.e. kNN with brute force search) is

$$N^2 D$$

- **Dimensionality reduction** – Transforming the feature set into a smaller number of features,
  i.e. from a larger number of dimensions D to a smaller number of dimensions D^*

- Dimensionality reduction methods to be discussed in more detail in Week 6

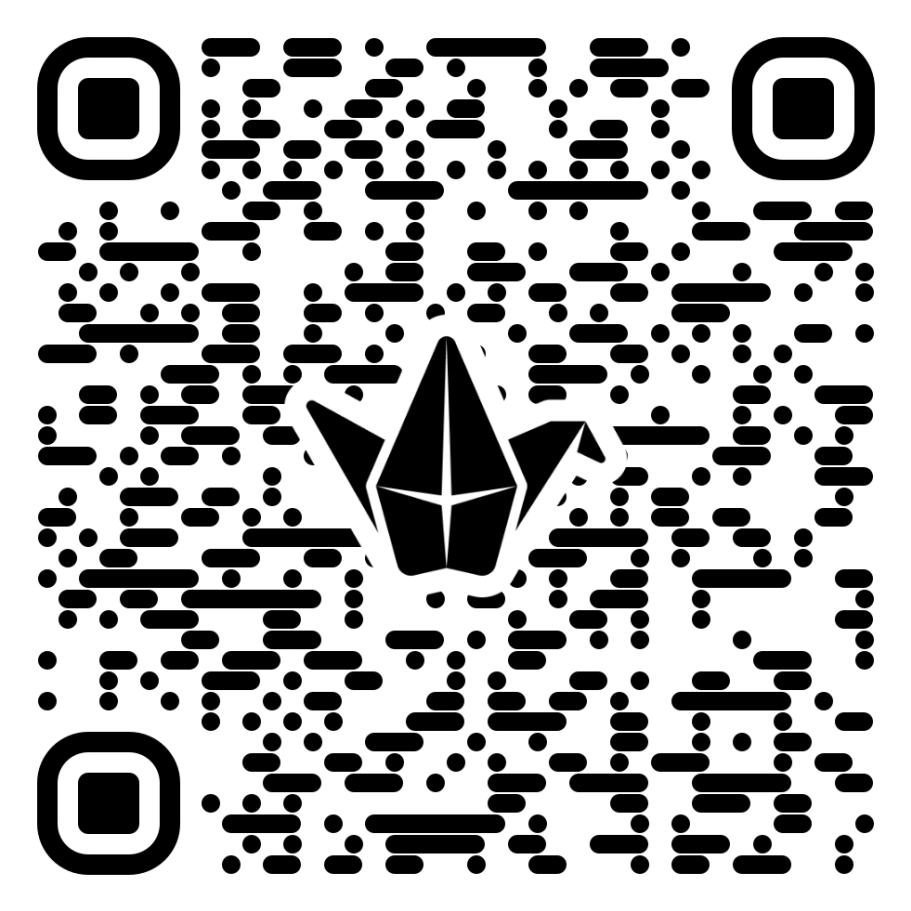
Summary

1. At inference time, the kNN finds **k-nearest neighbours** for obtaining predictions for a given data instance.

2. It has three main hyperparameters: the number of neighbours ***k***, the **distance metric**, the **voting strategy**.

3. Brute force search for nearest neighbours has high time complexity for inference time.

Any questions???



scan the QR code to ask questions

Math details and proofs

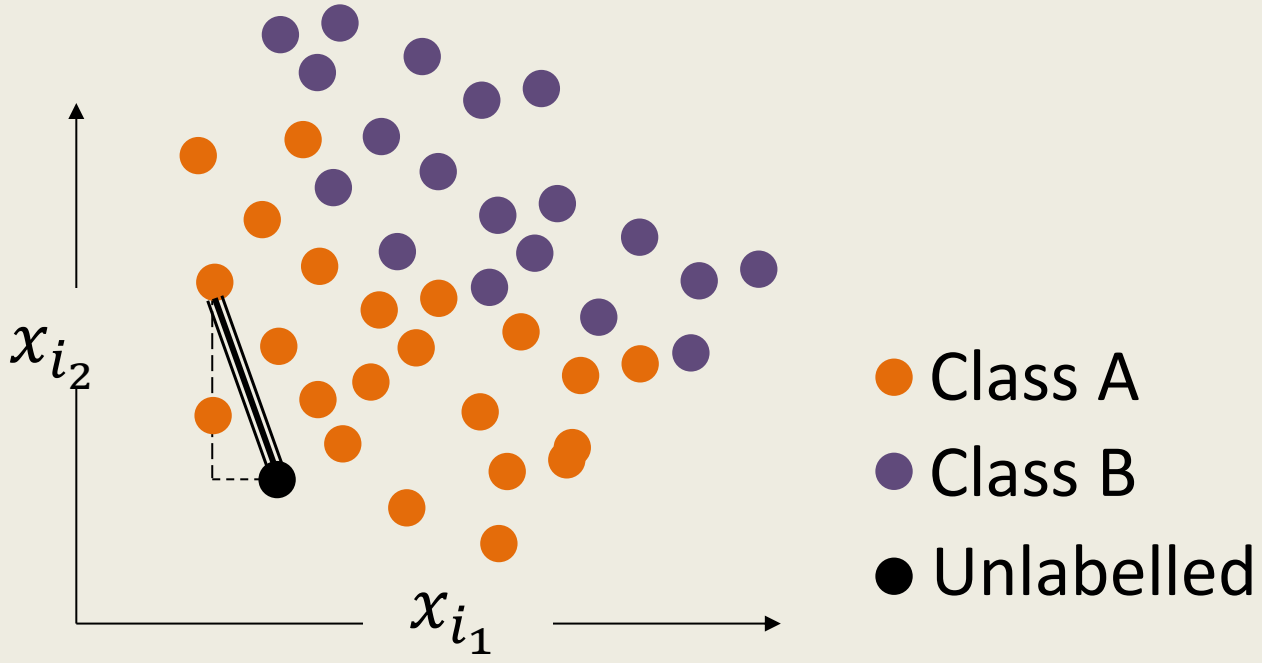
Derivation of Euclidean distance – MATH

From Pythagoras' theorem,

$$\left(dist(x_i, x_n)\right)^2 = \left(x_{i_1} - x_{n_1}\right)^2 + \left(x_{i_2} - x_{n_2}\right)^2$$

taking square root of both sides

$$\Longrightarrow dist(x_i, x_n) = \sqrt{\left(x_{i_1} - x_{n_1}\right)^2 + \left(x_{i_2} - x_{n_2}\right)^2}$$



Derivation of cosine similarity – MATH

$$\overrightarrow{x_i} \cdot \overrightarrow{x_j} = \|x_i\|\|x_j\|\cos\theta$$

making cosθ the subject of the formula

$$\cos\theta = \frac{\overrightarrow{x_i} \cdot \overrightarrow{x_j}}{\|x_i\|\|x_j\|}$$

expanding each term on the right hand side

$$\cos\theta = \frac{\sum_{d=1}^{D} x_{i_d} \cdot x_{j_d}}{\sqrt{\sum_{d=1}^{D} x_{i_d}{}^2} \cdot \sqrt{\sum_{d=1}^{D} x_{j_d}{}^2}}$$

Distance is actually dissimilarity

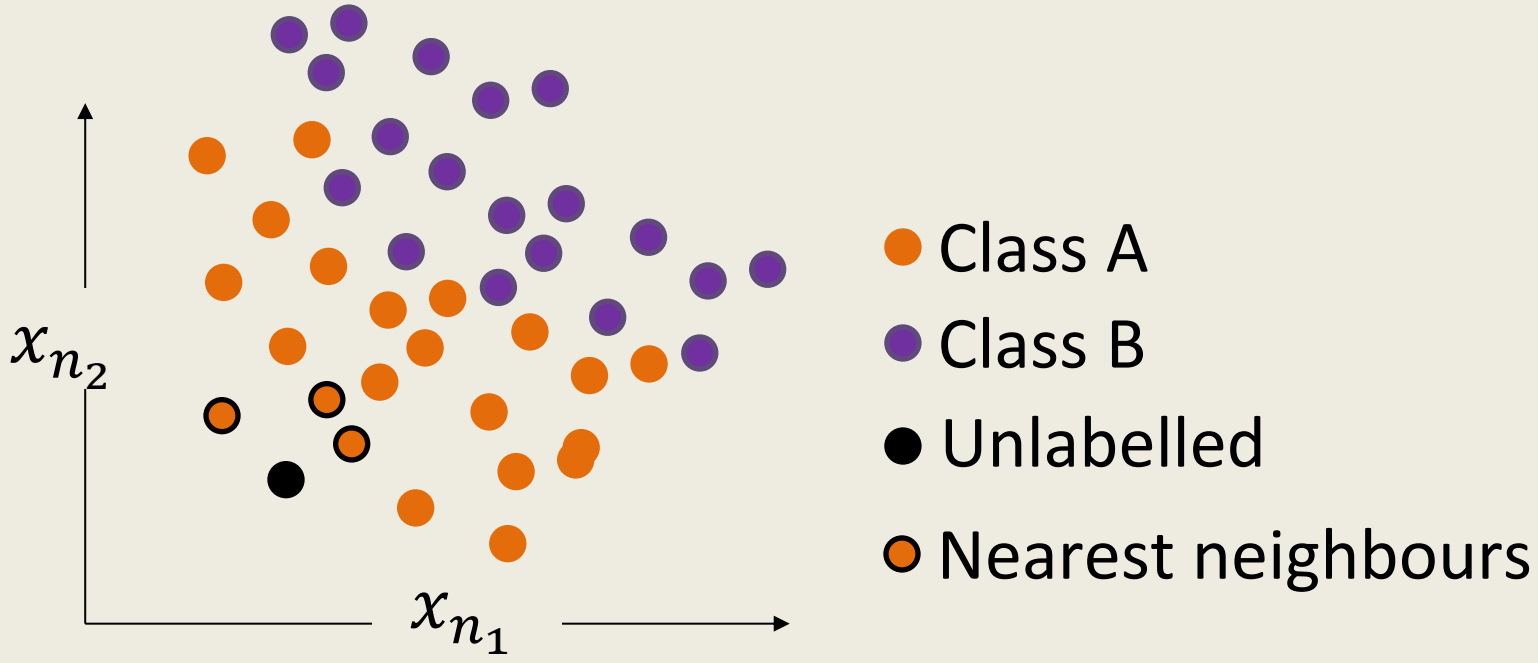
$$\Longrightarrow 1 - \cos\theta = 1 - \frac{\sum_{d=1}^{D} x_{i_d} \cdot x_{j_d}}{\sqrt{\sum_{d=1}^{D} x_{i_d}{}^2} \cdot \sqrt{\sum_{d=1}^{D} x_{j_d}{}^2}}$$

Majority voting example – MATH DETAILS

Consider number of neighbours *k*=3,

$$\Longrightarrow \hat{y}_i = \max_c \left\{ \sum_{j=1}^{3} 1_{y_j=c}\,, \qquad \forall c \in C \right\}$$

$$\Longrightarrow \hat{y}_i = \max_c \{3, 0\} \Longrightarrow Class\ A$$

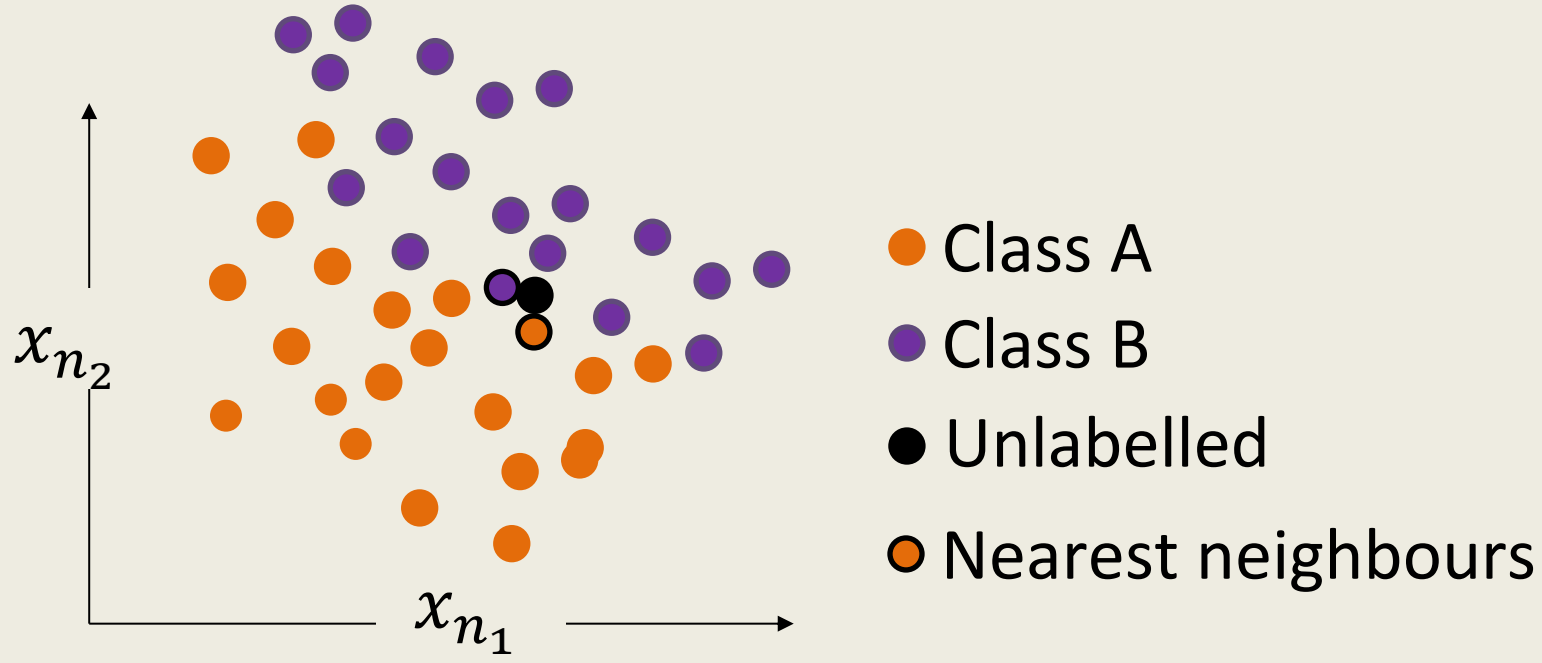


Another example – MATH DETAILS

Consider k=2

$$\Rightarrow \hat{y}_i = \max_c \left\{ \sum_{j=1}^{2} 1_{y_j=c}, \qquad \forall c \in C \right\}$$

$$\Rightarrow \hat{y}_i = \max_c \{1, 1\} \Rightarrow \text{a tie between Class A \& Class B}$$



Inference time complexity for kNN – MATH

- Time complexity

$$O(N^2 D)$$

N for the total number of data instances
D for the total number of features

- Recall that

$$dist(x_i, x_j) = \sqrt{\sum_{d=1}^{D} \left(x_{i_d} - x_{j_d}\right)^2}, \forall i, i = 1, 2, \cdots, N, j \neq i, \forall j, j = 1, 2, \cdots, M$$

i.e. each of M data instances in the test dataset is compared to each of N data instances in the training dataset, and each data instance in both sets has D dimensions that need to be included in the comparison