

K Nearest Neighbours

Dr. Rahul Kottath

Instance-Based Learning

Idea:

Similar examples have similar label.

Classify new examples like similar training examples.

Algorithm:

Given some new example x for which we need to predict its class y

Find most similar training examples

Classify x “like” these most similar examples

Questions:

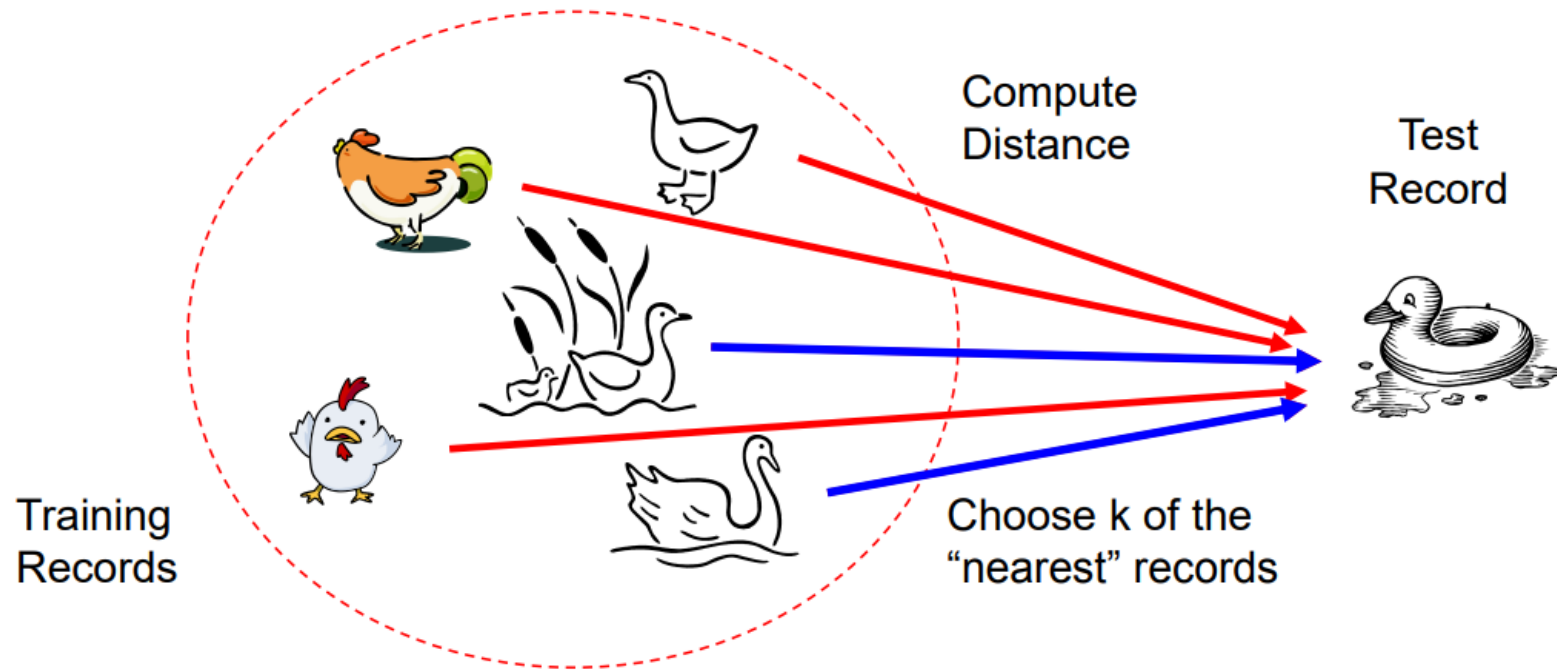
How to determine similarity?

How many similar training examples to consider?

Nearest Neighbor Classifiers

Basic idea:

- If it walks like a duck, quacks like a duck, then it's probably a duck



Nearest Neighbor Classification

0, 6, 7, 9
8, 6

In a single sentence, nearest neighbor classifiers are defined by their characteristic of classifying unlabeled examples by assigning them the class of the most similar labeled examples. Despite the simplicity of this idea, nearest neighbor methods are extremely powerful. They have been used successfully for:

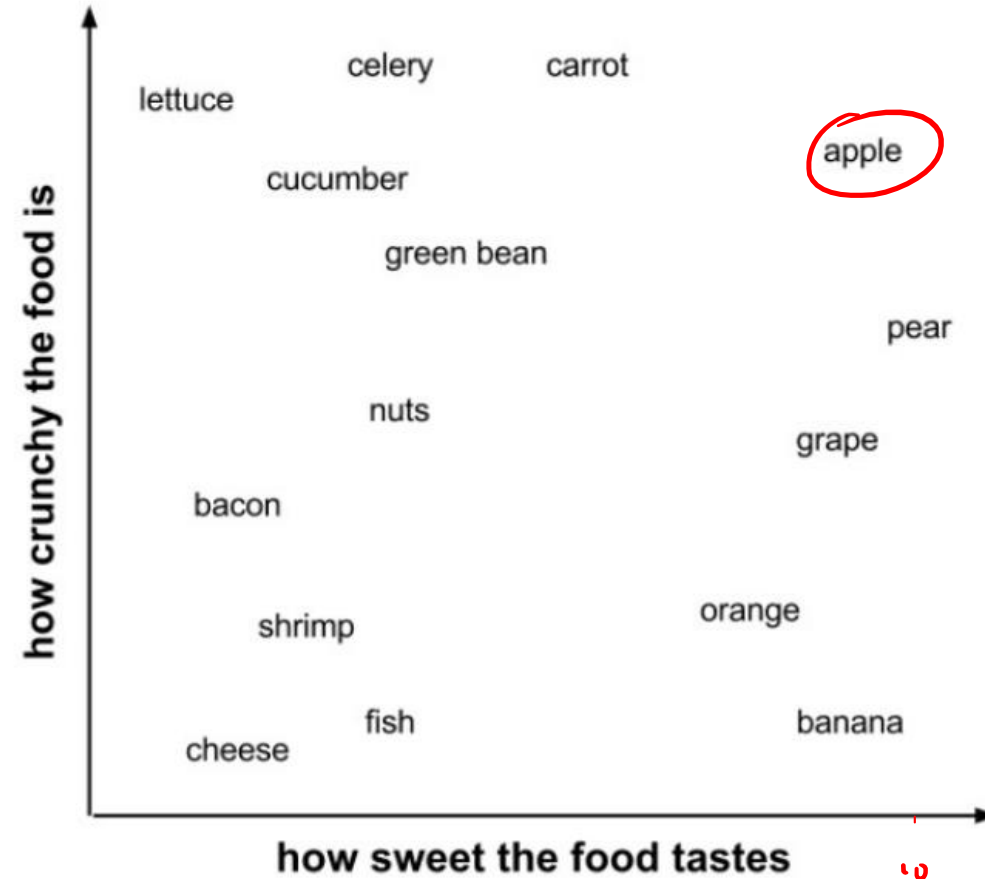
- Computer vision applications, including optical character recognition and facial recognition in both still images and video
- Predicting whether a person enjoys a movie which he/she has been recommended (as in the Netflix challenge)
- Identifying patterns in genetic data, for use in detecting specific protein or diseases

Example:

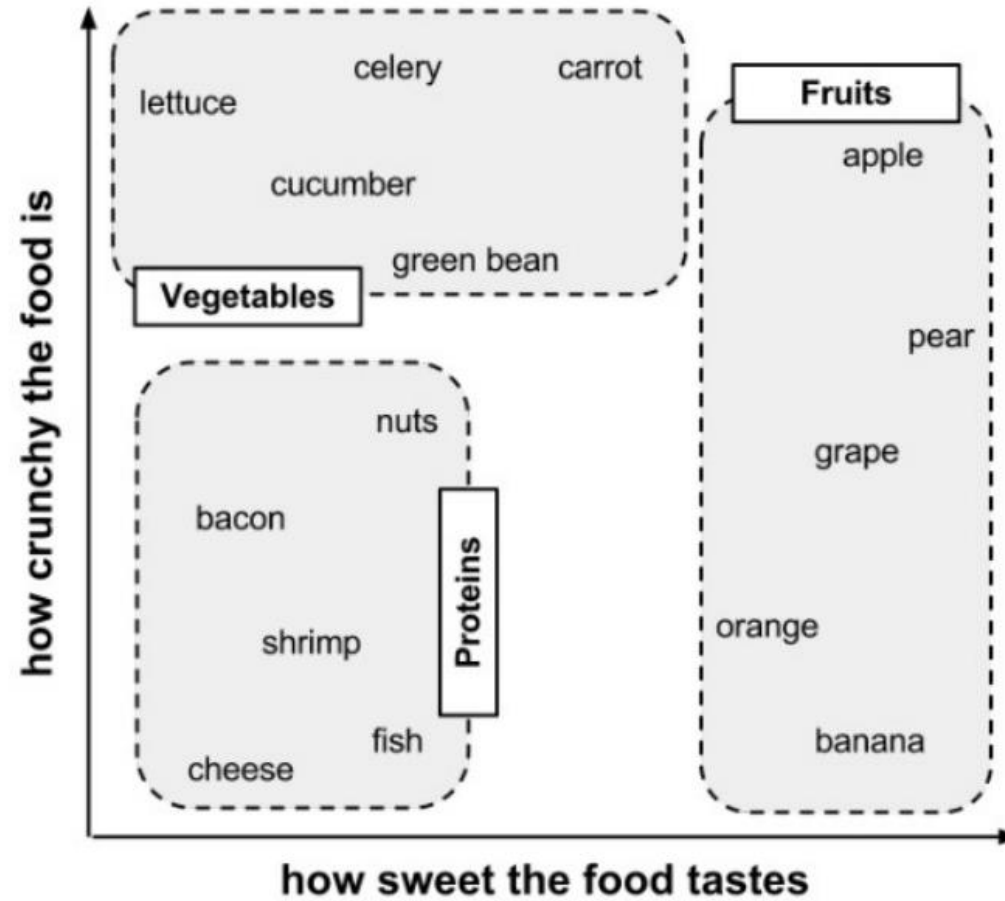
ingredient	sweetness ✓	crunchiness ✓	food type
apple	10	9	fruit —
bacon	1	4	protein
banana	10	1	fruit —
carrot	7	10	vegetable —
celery	3	10	vegetable ✓
cheese	1	1	protein

Reference: Machine Learning with R, Brett Lantz, Packt Publishing

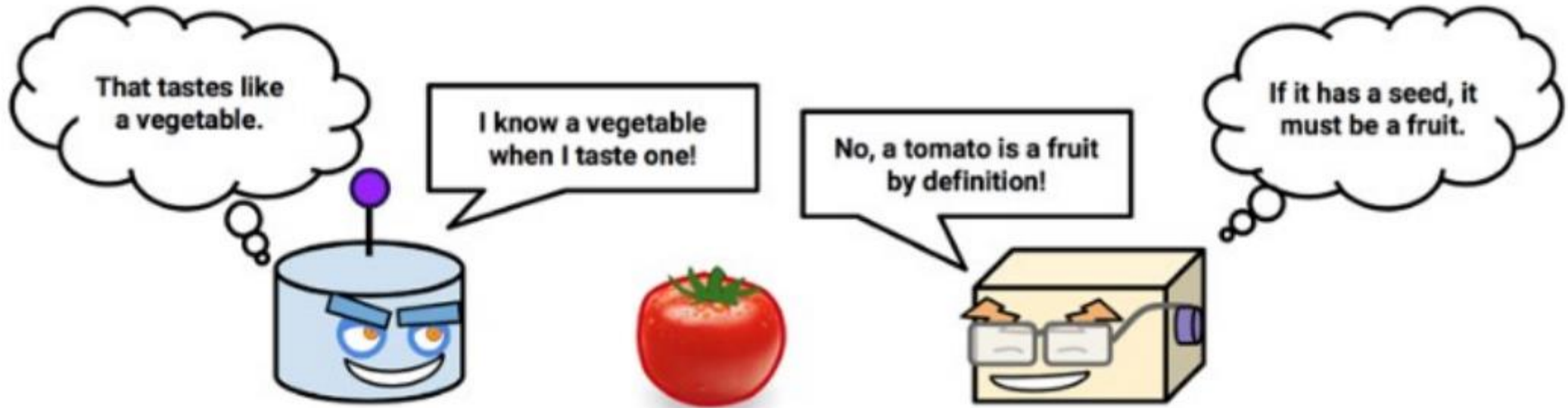
Example:



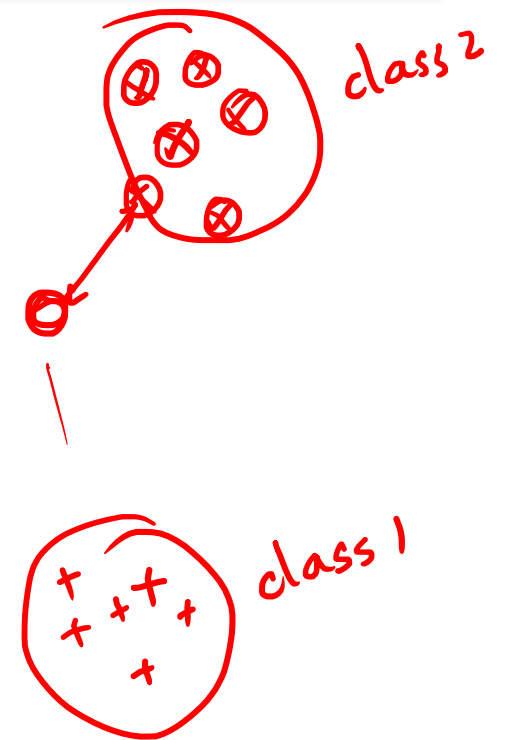
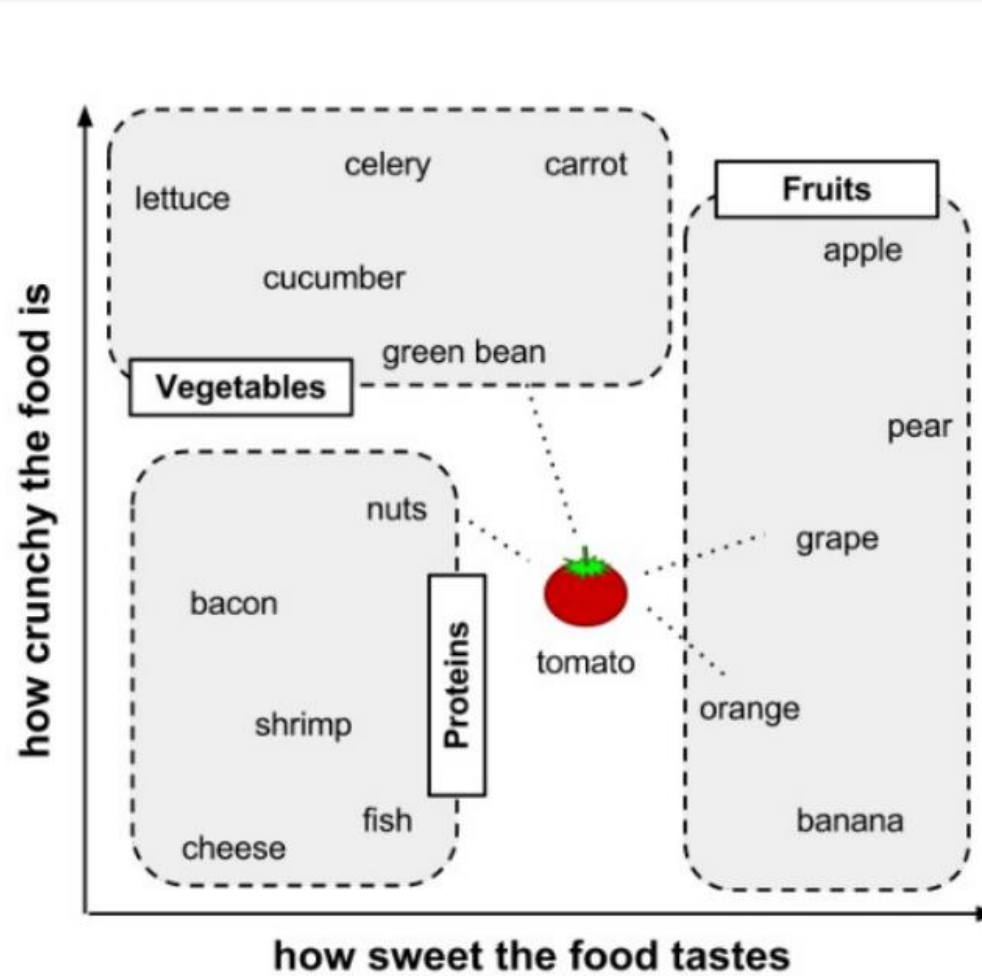
Example:



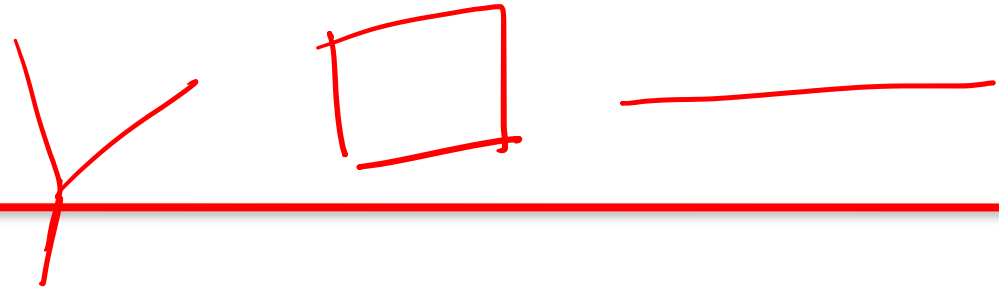
Classify me now!



Example:



Calculating Distance



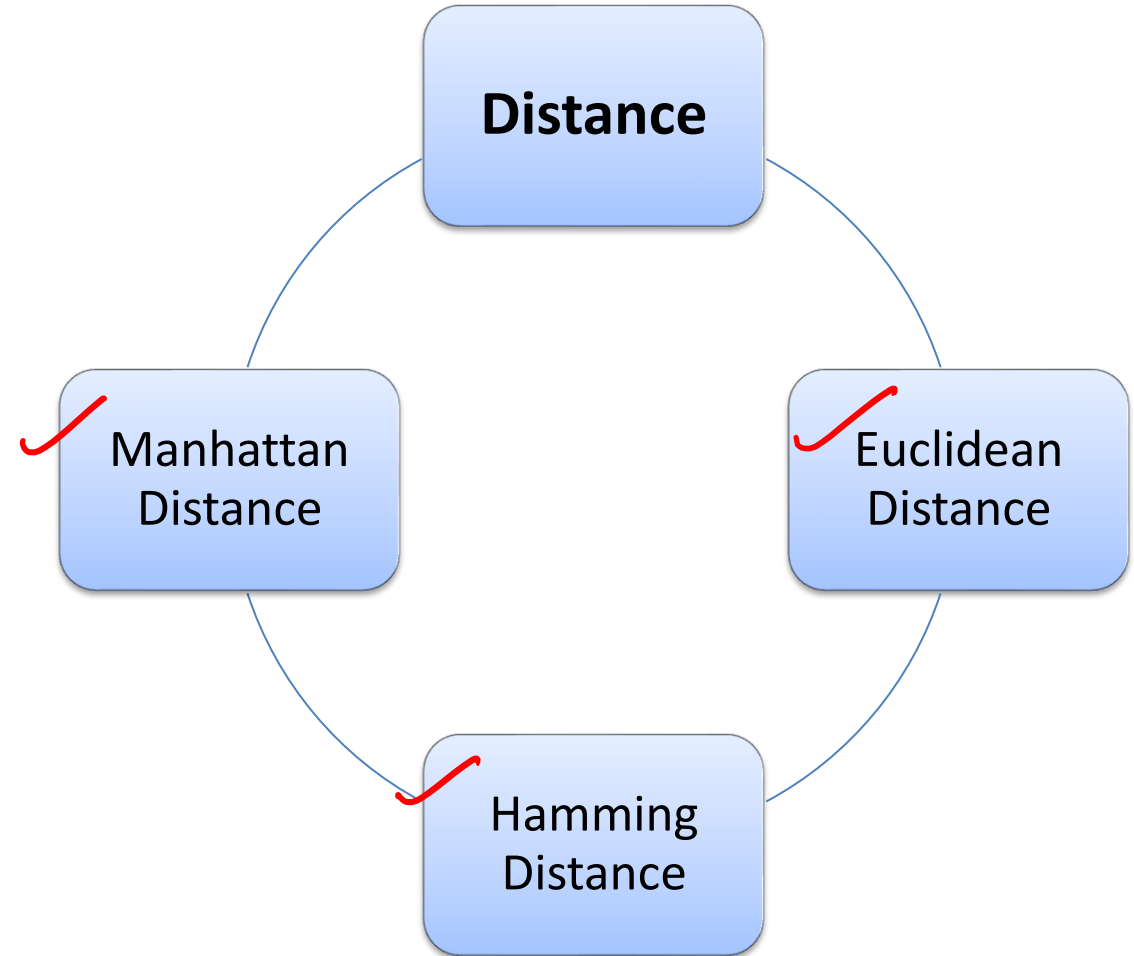
- Locating the tomato's nearest neighbors requires a distance function, or a formula that measures the similarity between two instances.
- There are many different ways to calculate distance.
- Traditionally, the kNN algorithm uses Euclidean distance, which is the distance one would measure if you could use a ruler to connect two points, illustrated in the previous figure by the dotted lines connecting the tomato to its neighbors.



Distances

01011
10110 ← 4

- Distance are used to measure similarity
- There are many ways to measure the distance s between two instances



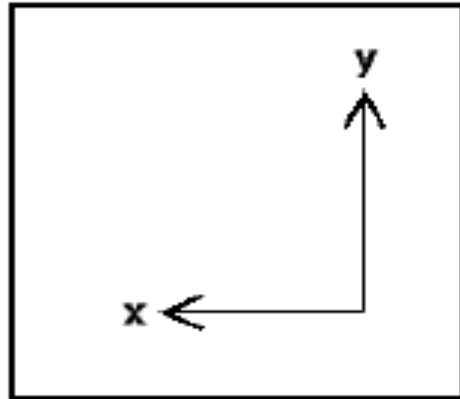
Distances

~~(x_1, y_1)~~ $\times (x_2, y_2)$

$(x_1 - x_2)^2 + (y_1 - y_2)^2 = (-1)(x_1 - x_2)^2 = (x_1 - x_2)^2$

Manhattan Distance

$$|X1-X2| + |Y1-Y2|$$

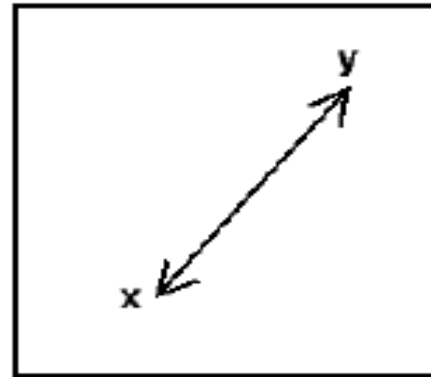


Manhattan

Euclidean Distance

$$\sqrt{(x1-x2)^2 + (y1-y2)^2}$$

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



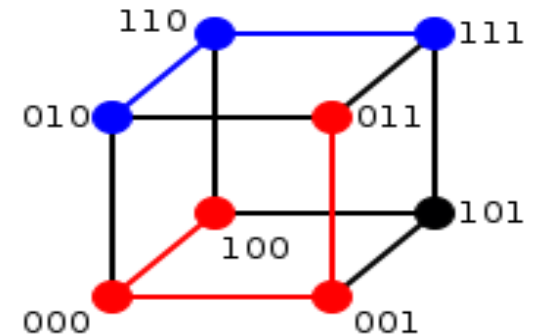
Euclidean

Hamming Distance

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

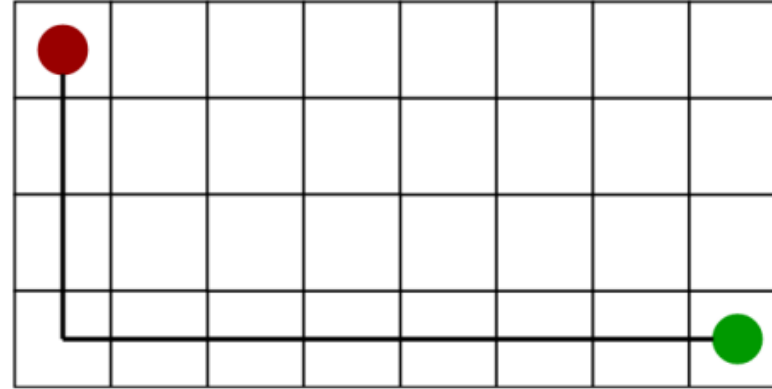
$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

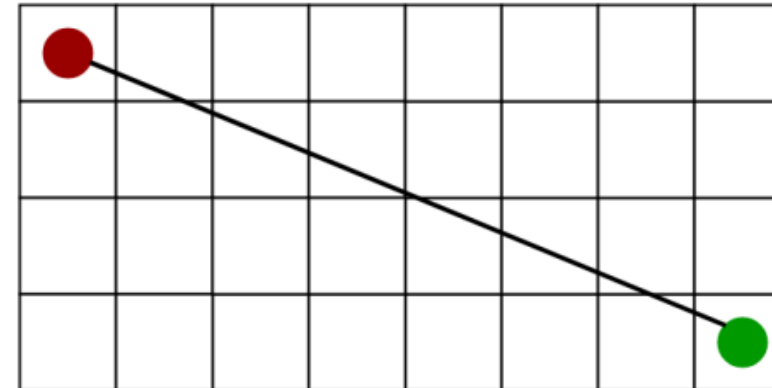


Distances

Manhattan Distance



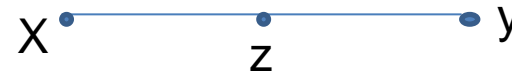
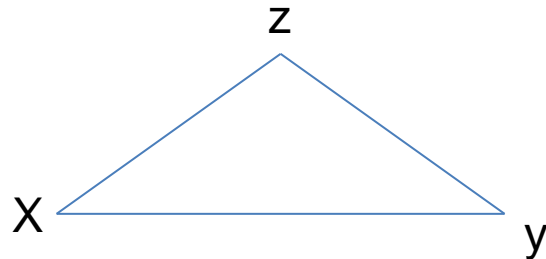
Euclidean Distance



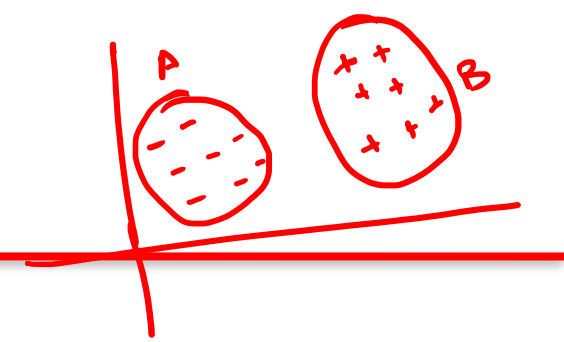
Properties of Distance

- $\text{Dist}(x,y) \geq 0$
- $\text{Dist}(x,y) = \text{Dist}(y,x)$ are Symmetric
- Detours can not Shorten Distance

$$\text{Dist}(x,z) \leq \text{Dist}(x,y) + \text{Dist}(y,z)$$

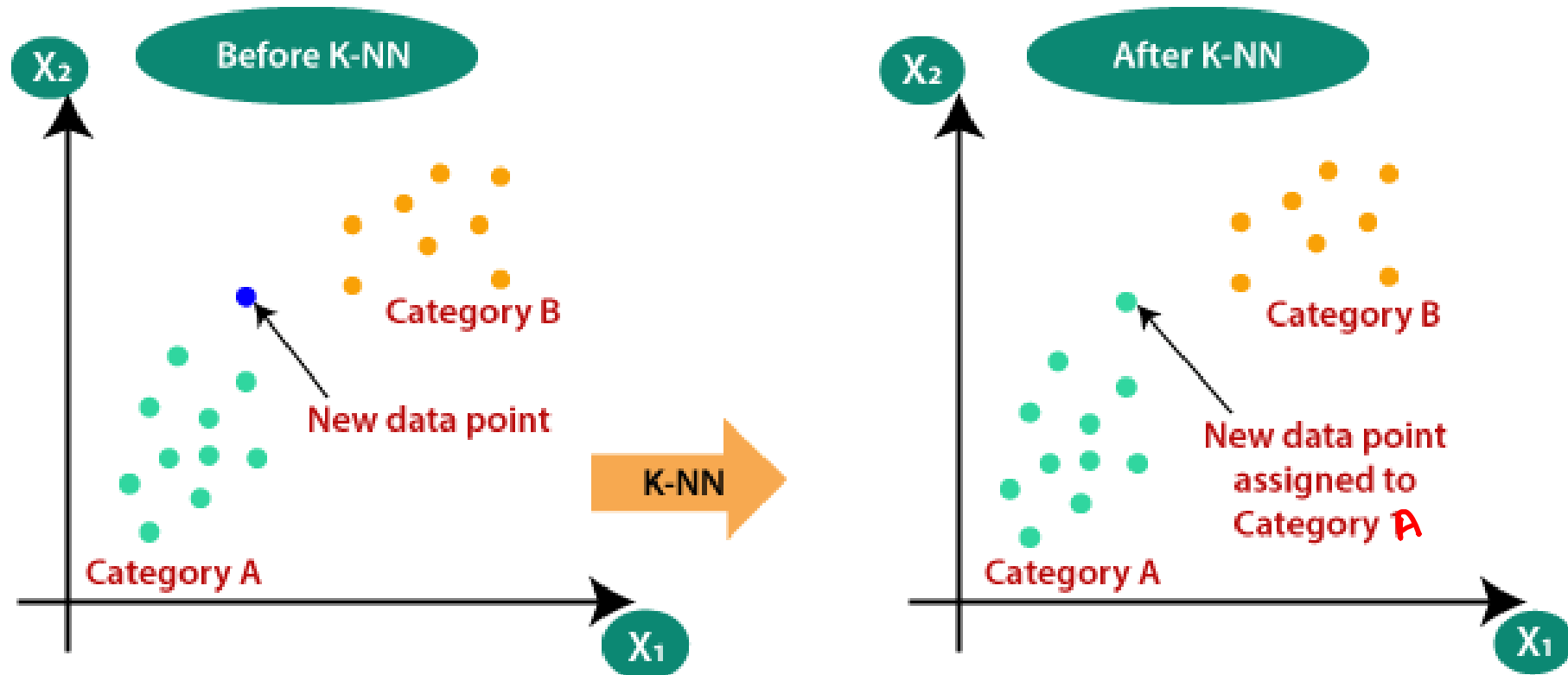


The kNN Algorithm



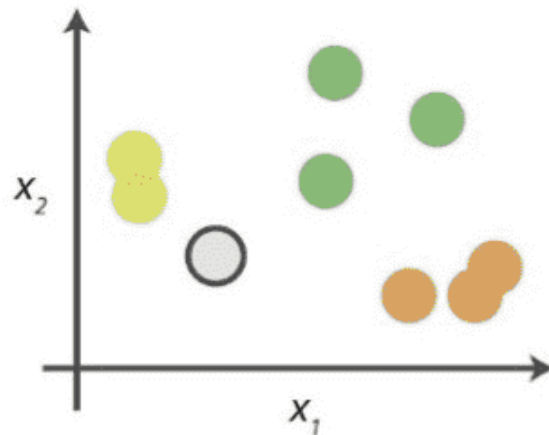
- The kNN algorithm begins with a training dataset made up of examples that are classified into several categories, as labeled by a nominal variable.
 - Assume that we have a test dataset containing unlabeled examples that otherwise have the same features as the training data.
 - For each record in the test dataset, kNN identifies k records in the training data that are the "nearest" in similarity, where k is an integer specified in advance.
 - The unlabeled test instance is assigned the class of the majority of the k nearest neighbors
-

The KNN Algorithm



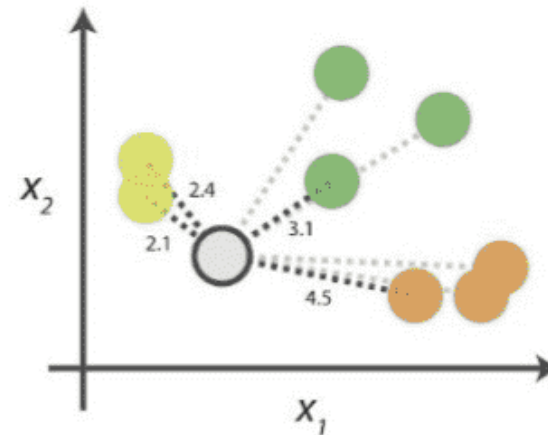
The KNN Algorithm

0. Look at the data



Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

1. Calculate distances



Start by calculating the distances between the grey point and all other points.









$$d = \sqrt{(1-3)^2 + (2-4)^2}$$

$(1, 2)$
 $(3, 4)$

The KNN Algorithm







2 class → 3 NN
- 5 NN
7 NN

2. Find neighbours

Point Distance		
 .. 	2.1	→ 1st NN
 .. 	2.4	→ 2nd NN
 .. 	3.1	→ 3rd NN
 .. 	4.5	→ 4th NN

Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

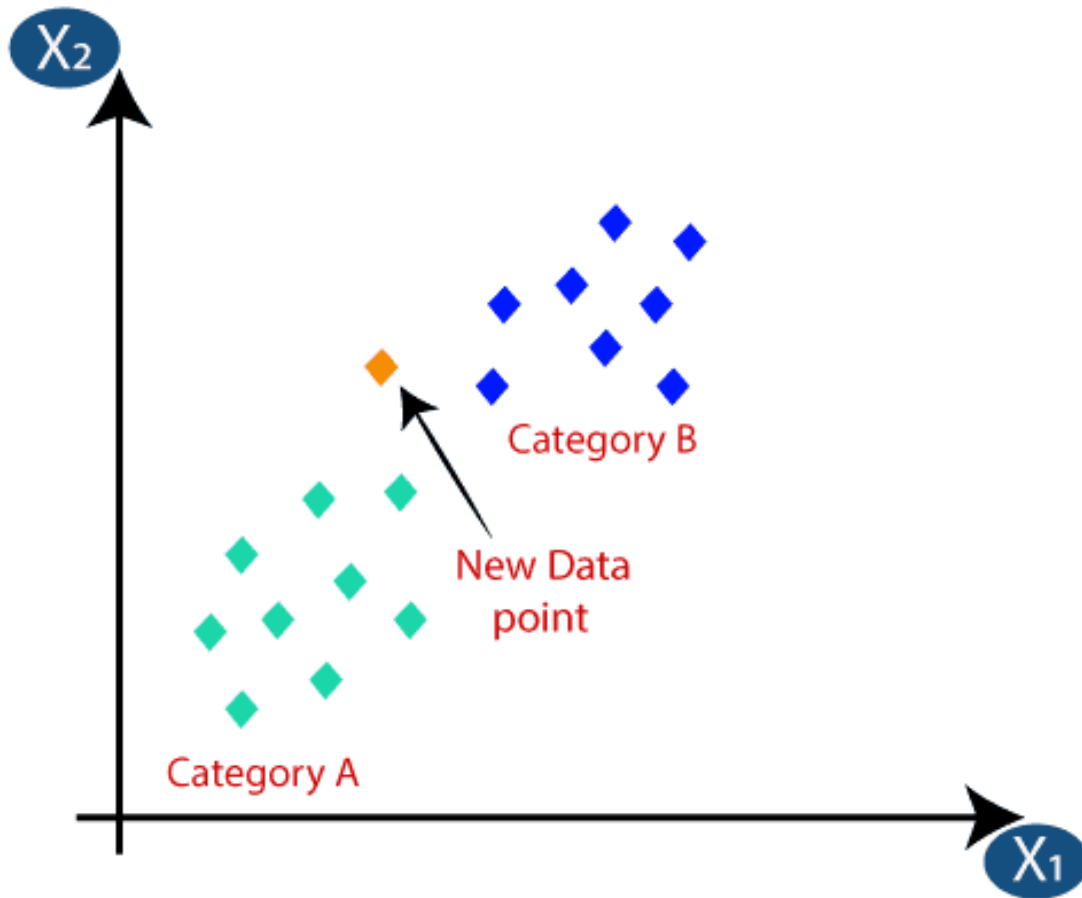
3. Vote on labels

Class	# of votes	
	2	→ Class  wins the vote! Point  is therefore predicted to be of class  .
	1	
	1	

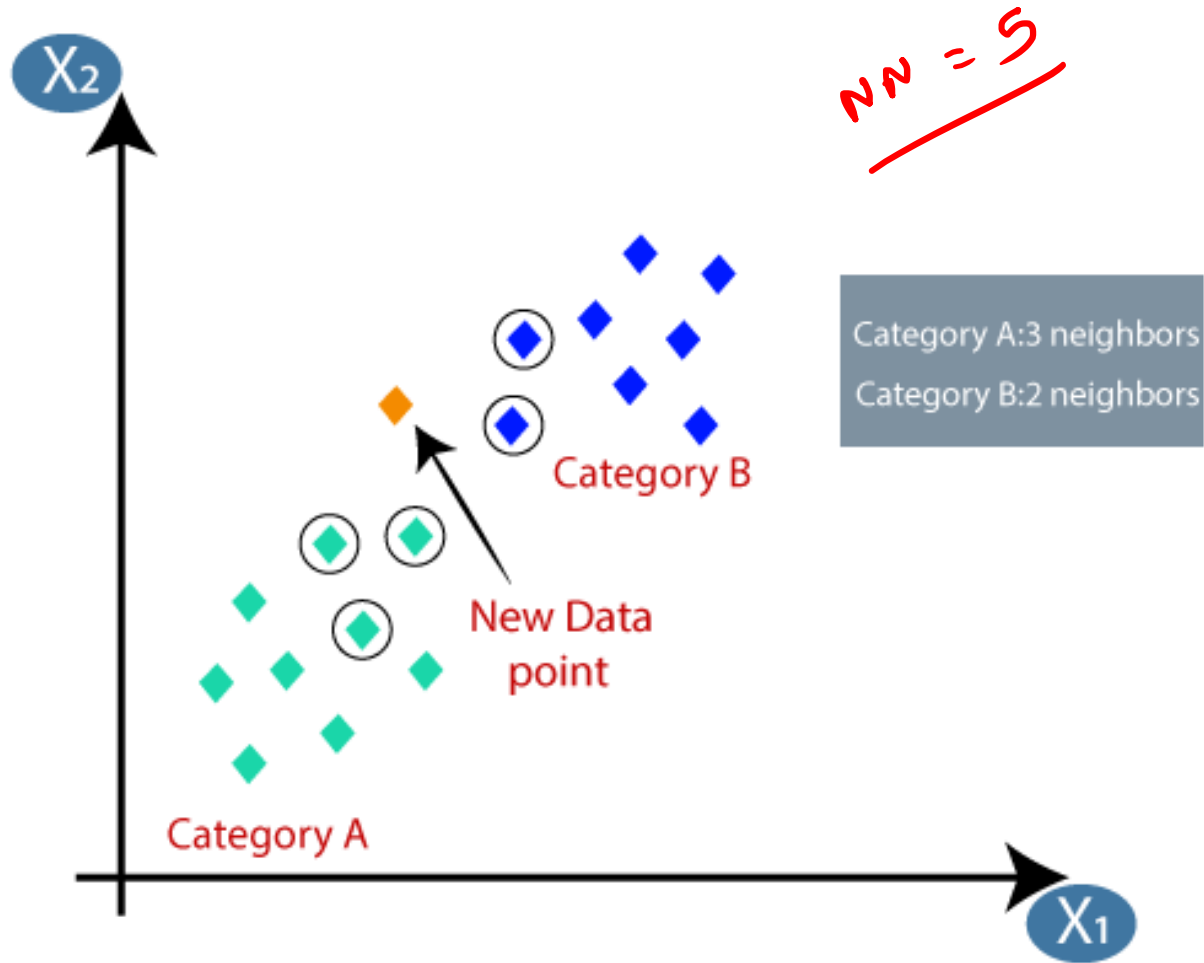
Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

The KNN Algorithm

NN



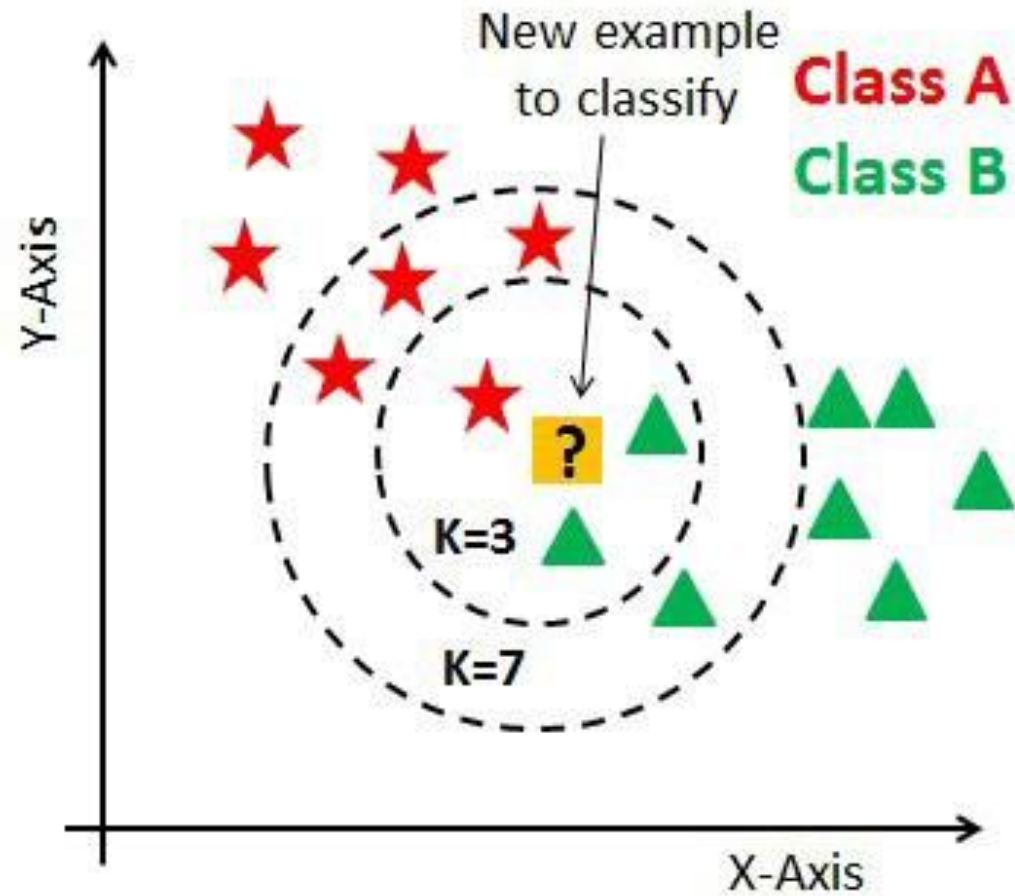
The KNN Algorithm



New data = Category A

The KNN Algorithm

ⓧ



ⓧ $\rightarrow k$

$k=3$

\rightarrow

class B

$k=7$

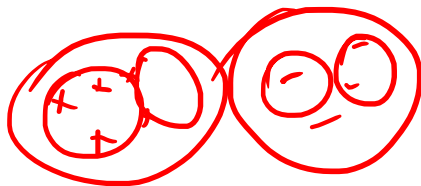
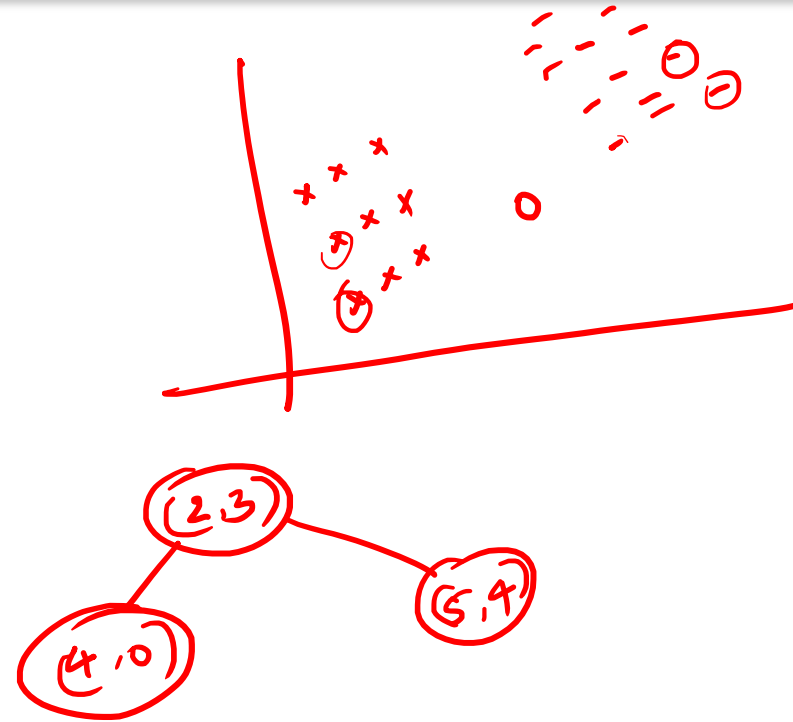
\rightarrow

class A

hyperparameter

Ways to perform K-NN

- Brute Force ✓
- ✓ k-Dimensional Tree (kd tree)
- ✓ Ball Tree



Hyperparameter tuning:

What is the best distance to use?

What is the best value of k to use?

i.e., how do we set the **hyperparameters**?

Hyperparameter tuning:

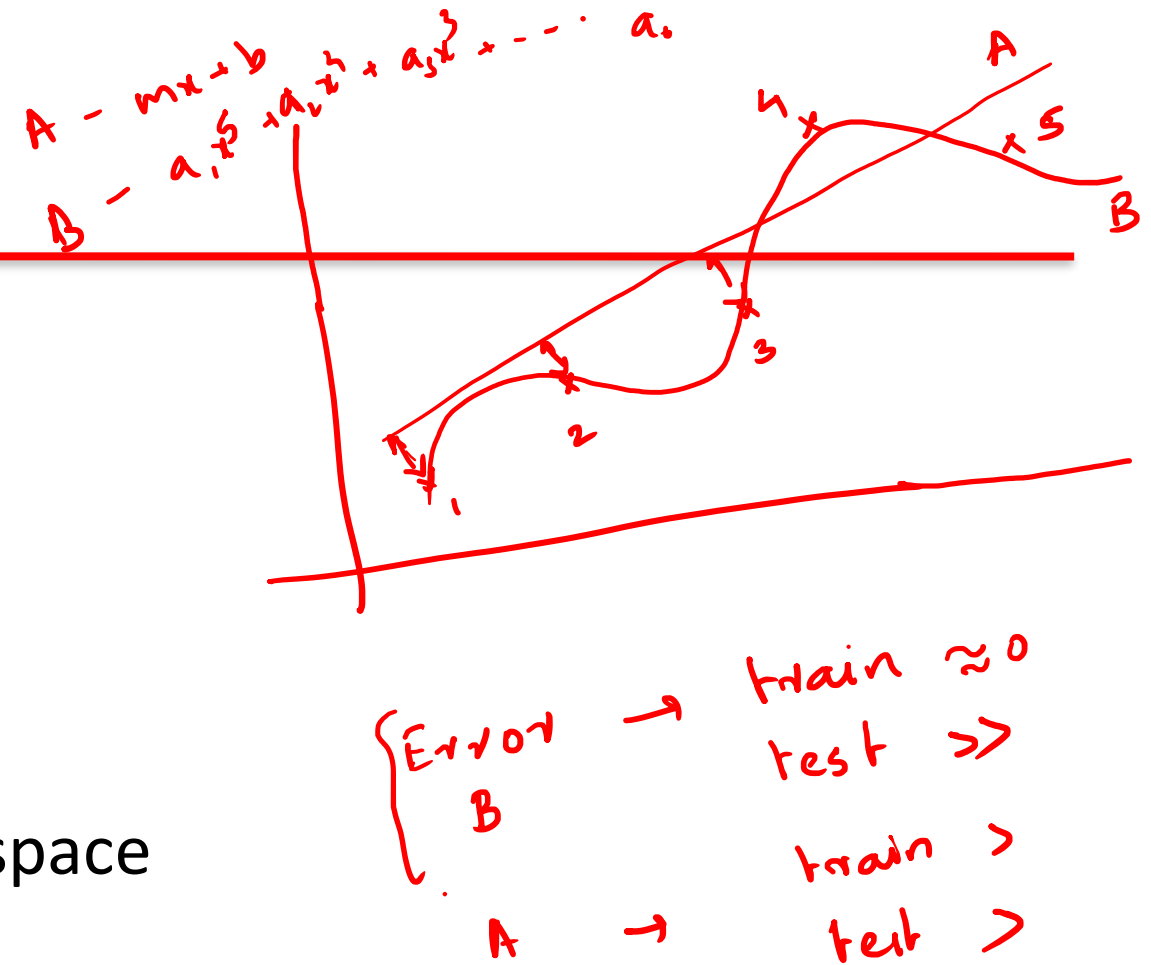
Increase k:

Makes KNN less sensitive to noise

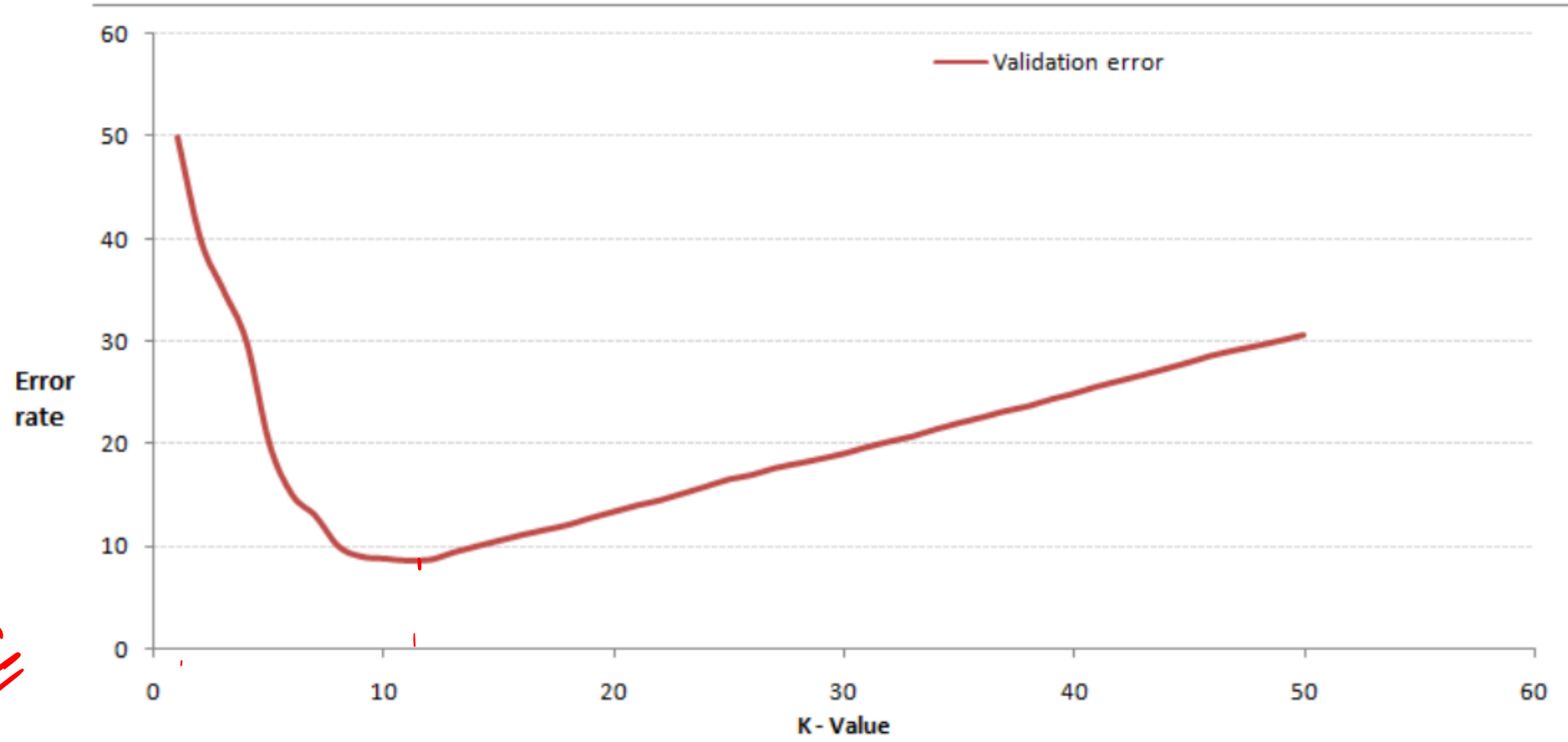
Decrease k:

Allows capturing finer structure of space

➔ Pick k not too large, but not too small (depends on data)



Choosing appropriate k



$k=11$

Hyperparameter tuning:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

What is the best **distance** to use?

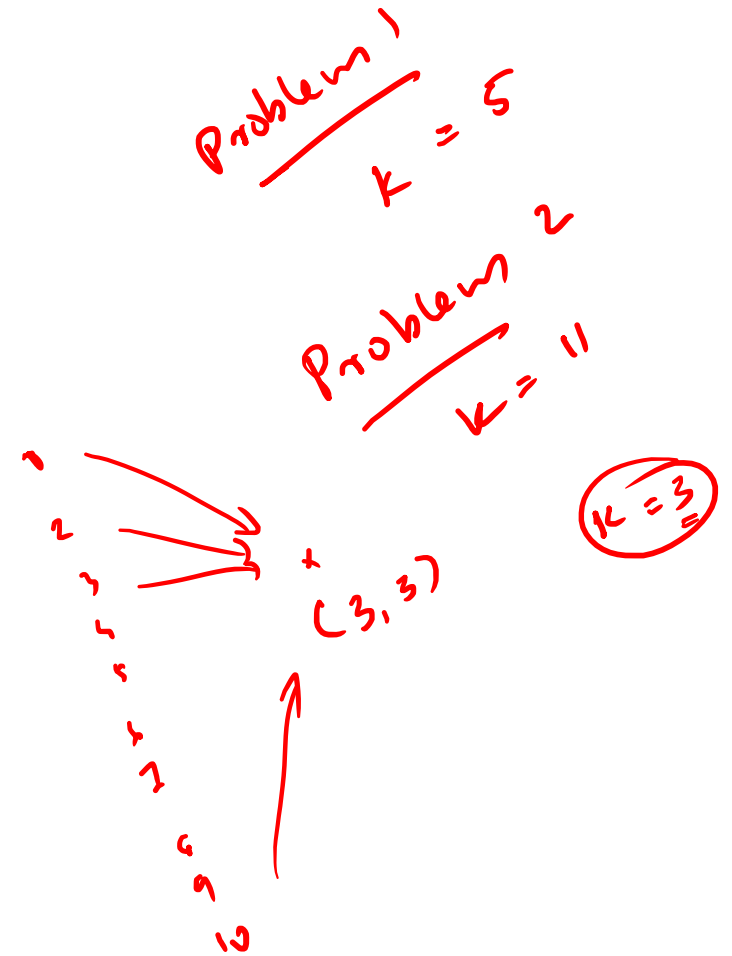
What is the best value of **k** to use?

i.e. how do we set the **hyperparameters**?

Very problem-dependent.

Must try them all out and see what works best.

✓ No. of rooms — 1-10
✓ Floor area — 500 — 5000
✓ No. of bathrooms — 1-6



✓ Min-Max normalization

$$\begin{array}{r} 1 - 10 \rightarrow 0 - 1 \\ 500 - 5000 \\ 0 - 1 \end{array}$$

- The traditional method of rescaling features for kNN is minmax normalization.
- This process transforms a feature such that all of its values fall in a range between 0 and 1. The formula for normalizing a feature is as follows. Essentially, the formula subtracts the minimum of feature X from each value and divides by the range of X:

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

$$\frac{600 - 500}{5000 - 500} = \frac{100}{4500}$$

- Normalized feature values can be interpreted as indicating how far, from 0 percent to 100 percent, the original value fell along the range between the original minimum and maximum.
-

The Lazy Learning

(KNN)

- Using the strict definition of learning, a lazy learner is not really learning anything.
 - Instead, it merely stores the training data in it. This allows the training phase to occur very rapidly, with a potential downside being that the process of making predictions tends to be relatively slow.
 - Due to the heavy reliance on the training instances, lazy learning is also known as instance-based learning or rote learning.
-

KNN Advantages

- Easy to program
 - No optimization or training required
 - Classification accuracy can be very good; can outperform more complex models.
-

KNN Disadvantages

- Need distance/similarity measure and attributes that “match” target function.
- For large training sets,
 - Must make a pass through the entire dataset for each classification. This can be prohibitive for large data sets.

1000

Python Packages needed

- pandas
 - Data Analytics
 - numpy
 - Numerical Computing
 - matplotlib.pyplot
 - Plotting graphs
 - sklearn
 - Classification and Regression Classes
-

Implementation Using sklearn

Let's go to Jupyter Notebook!
