Distance-based Classifiers

Department of Computer Languages and Systems

Introduction: general context

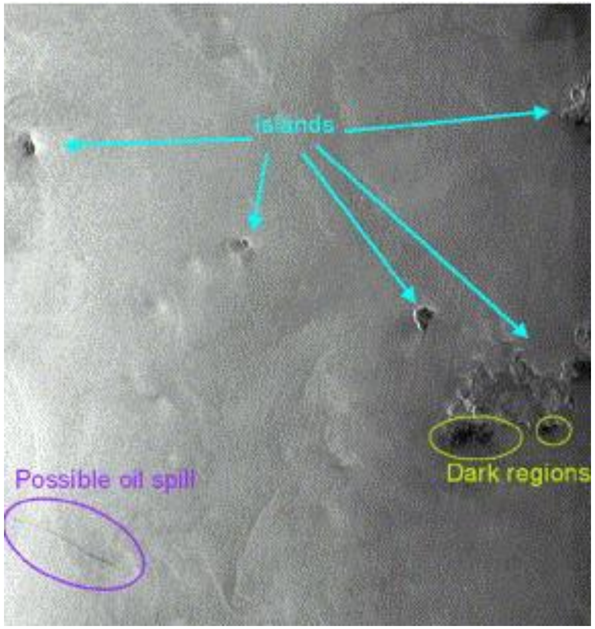
Non-parametric, supervised models can be applied if there exist …

- *objects* capable of being described by some formal representation (a vector, a string, ...)
- *classes* or categories that group the objects
- need to *automate the classification* (or identification) of new objects into some known class

Introduction: an example of classification

A real problem: identification of oil-spills in the ocean

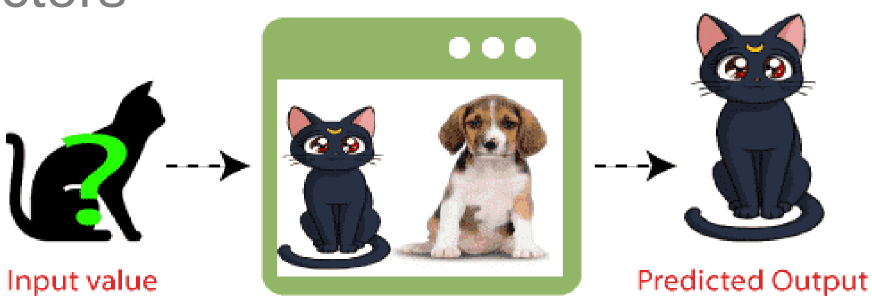
- objects: satellite images
- formal representation: vector of features or attributes (image *pixels, image features, etc.*)
- classes: oil-spills, look-alikes (ships, islands, etc.)



Distance-based classifiers

Motivation:

- the best model is not always the most complicated
- these classifiers are based on the idea that objects of the same class are the most similar
- they use some similarity (distance) measures defined on attribute vectors

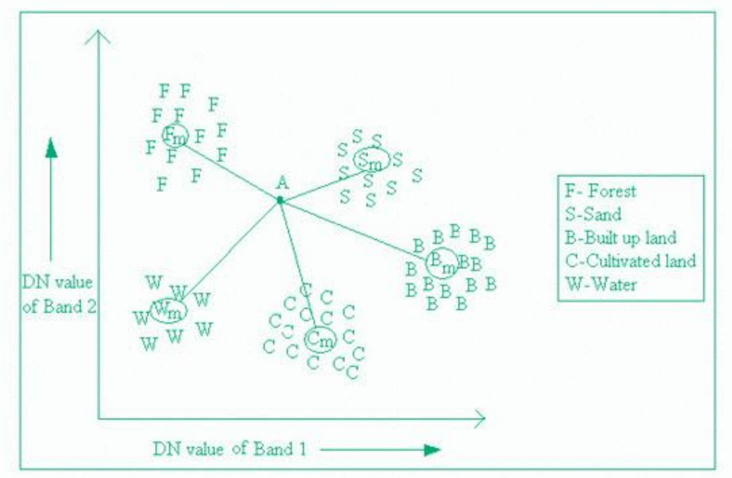


Minimum distance classifier

This is by far the simplest classification model

- it does not take all the training data into account
- it measures the similarity between test data and some representative prototype of each class
- a common prototype is the centroid of each class
- some distance is calculated for each prototype and the shortest distance is determined
- a test object is assigned to the class of the nearest centroid (the minimum distance)

Minimum distance classifier (ii)



minimum_distance(T_{tra}, x)

    min $\leftarrow$ MAX

    for each class ω_i

        $z_i \leftarrow$ compute_centroid(ω_i)

    for each prototype z_i

        dist $\leftarrow$ compute_distance(x, z_i)

        if dist < min

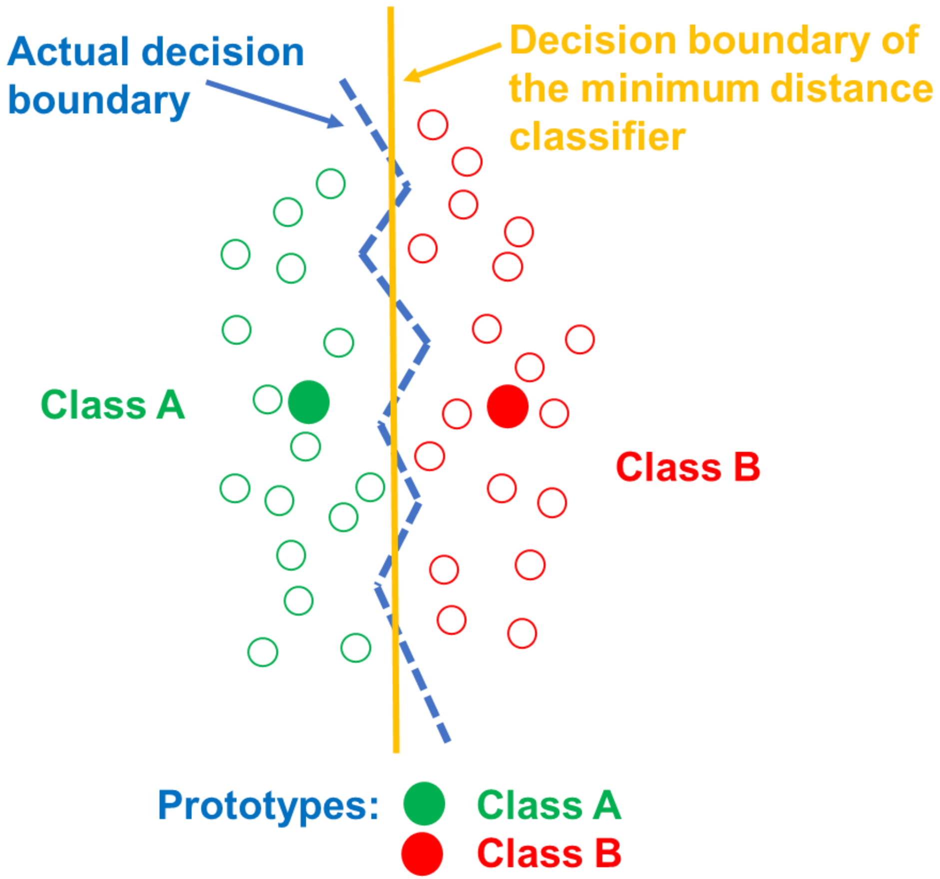
            min $\leftarrow$ dist

            class(x) $\leftarrow \omega_i$

Minimum distance classifier (iii)

Drawback:

- Loss of useful information



k Nearest neighbours (*k*-NN)

- one of the conceptually simplest yet powerful models
- it takes all the training data into consideration
- it searches for the *k* training data closest to a test point
- a class label is assigned on the basis of a majority vote: the class that is most frequently represented among the *k* neighbours
- it is said to be a lazy learner since it does not perform any training. Instead, it just stores the data

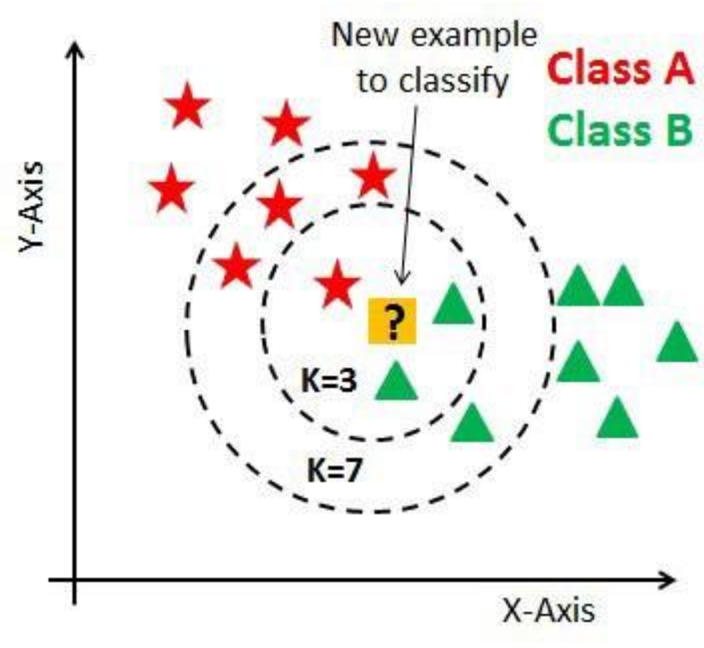
k Nearest neighbours (*k*-NN) (ii)

We need to define two hyperparameters

- the **distance metric**, to determine which training data are the closest to a given test point
- the ***k* value**, which defines how many neighbours will be checked to guess the classification of a test point

k Nearest neighbours (*k*-NN) (iii)

Selecting the *k* value is the most critical question



k Nearest neighbours (*k*-NN) (iv)

Using error curves is a common method to choose the value of *k*

- Training error is zero at $k = 1$ because the nearest neighbour to a point is that point itself
- Test error is high at low values of k due to variance (overfitting), it subsequently lowers and stabilises, and with further increase in the value of k, the test error increases again due to bias (underfitting)

The value of *k* at which the test error stabilises and is low is taken as the optimal value of *k*

Similarity measures (i)

The distance metric:

- Minkowski distance: the generalized form of the Euclidean (p = 2), Manhattan (p = 1) and Chebyshev ((p =∝) distance metrics
  - Euclidean (or L2 norm) distance: the most commonly used distance measure, and it is limited to real-valued vectors
  - Manhattan (or city block or L1 norm) distance: measures the absolute value between two points
  - Chebyshev (or chessboard) distance: the minimum number of moves needed by a king to go from one square on a chessboard to another equals the Chebyshev distance

Similarity measures (ii)

The distance metric:

- Hamming distance: typically used with Boolean and string vectors
- Cosine similarity: defined as the cosine of the angle between two attribute vectors

Similarity measures (iii)

Minkowski distance:

$$d_p(x,y) = \left(\sum_{i=1}^{d} |y_i - x_i|\right)^{1/p}$$

Euclidean distance:

$$d_2(x,y) = \sqrt{\sum_{i=1}^{d} (y_i - x_i)^2}$$

Manhattan distance:

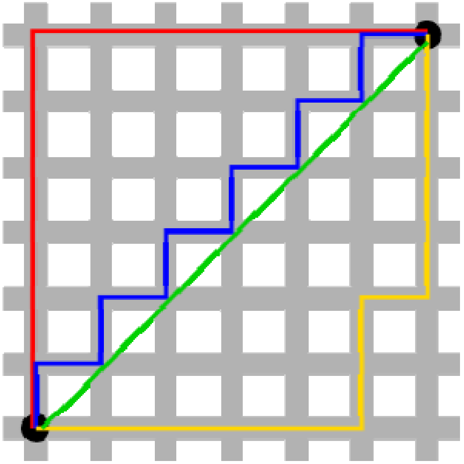
$$d_1(x,y) = \sum_{i=1}^{d} |y_i - x_i|$$

Chebyshev distance:

$$d_\infty(x,y) = \max_{i=1\dots d} |y_i - x_i|$$

Similarity measures (iv)

An example of the Minkowski distances:



Manhattan distance =12
(red, blue, or yellow)

Euclidean distance = 8.5
(green – continuous)

Chebyshev distance = 6
(green – d i s c r e t e)

Similarity measures (v)

- Hamming distance:

$$d_H(x, y) = \sum_{i=1}^{d} |y_i - x_i|$$

$$\text{if } x = y \Rightarrow d_H(x, y) = 0$$
$$\text{if } x \neq y \Longrightarrow d_H(x, y) = 1$$

Similarity measures (vi)

An example of the Hamming distance:

$$d_H(x, y) = \sum_{i=1}^{d} |y_i - x_i|$$

Vector 1	1	0	1	0	0	0	1	1
Vector 2	1	0	0	0	0	0	0	1

$$d_H = 0+0+1+0+0+0+1+0 = 2$$

the Hamming distance is 2 because only two elements of the vectors differ

Similarity measures (vii)

- Cosine similarity:

$$S_C(\vec{x},\vec{y}) = \cos(\theta) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2}\sqrt{\sum_{i=1}^{d} y_i^2}}$$

$$\begin{aligned}
&\text{if } S_C(\vec{x},\vec{y}) = -1 \Longrightarrow \vec{x} \text{ and } \vec{y} \text{ are exactly opposite}\\
&\text{if } S_C(\vec{x},\vec{y}) = +1 \Longrightarrow \vec{x} \text{ and } \vec{y} \text{ are exactly the same}\\
&\text{if } S_C(\vec{x},\vec{y}) = 0 \quad \Longrightarrow \vec{x} \text{ and } \vec{y} \text{ are orthogonal}
\end{aligned}$$

intermediate values indicate intermediate similarity

Normalization

- **Continuous numerical attributes:** to avoid that some attributes dominate over others, we should generally "normalize" the attributes in a common range such as [0,1] or [-1, 1]

- Normalization is generally required when we are dealing with attributes on a different scale as this may lead to poor model performance

person_name	Salary	Year_of_experience	Expected Position Level
Aman	100000	10	2
Abhinav	78000	7	4
Ashutosh	32000	5	8
Dishi	55000	6	7
Abhishek	92000	8	3
Avantika	120000	15	1
Ayushi	65750	7	5

The attributes salary and year_of_experience are on different scale and hence attribute salary can take high priority over attribute year_of_experience in the model.

Normalization (ii): typical methods

- z-score (or zero-mean) normalization or standardization: it re-scales a feature value so that it has distribution with 0 mean value and variance equals to 1

$$z_i = \frac{x_i - \bar{x}_i}{\sigma_i}$$

where $\bar{x}_i$ and σ_i are the mean value and the standard deviation of attribute *i*, respectively

- min-max normalization: it re-scales a feature or observation value with distribution value between 0 and 1

$$z_i = \frac{x_i - \min_i X}{\max_i X - \min_i X}$$

Normalization (iii): typical methods

- decimal scaling: it normalizes by moving the decimal point of values of the data. We divide each value of the data by the maximum absolute value of data

$$z_i = \frac{x_i}{10^j}$$

where *j* is the smallest integer such that $\max_i(|z_i|) < 1$

Example: our input data is -10, 201, 301, -401, 501, 601, 701

Step 1: Maximum absolute value in input data: 701

Step 2: Divide the input data by 1000 (i.e., $j = 3$, 10^3)

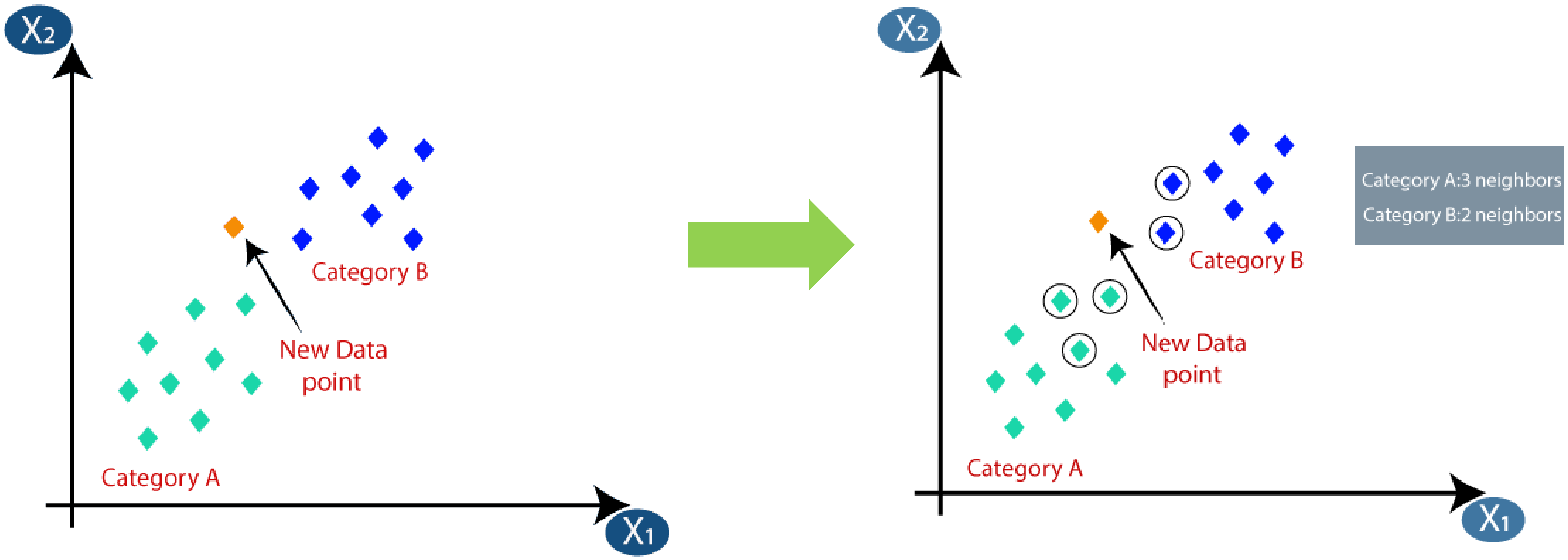
Result: The normalized data is: -0.01, 0.201, 0.301, -0.401, 0.501, 0.601, 0.701

The *k*-NN algorithm

1. Select the number *k* of neighbours
2. Calculate the distance between the training data and the test sample
3. Take the *k* nearest neighbours as per the calculated distance
4. Among these *k* neighbours, count the number of the training data in each class
5. Assign the test sample to that category (class) with a majority of neighbours

The *k*-NN algorithm (ii)

Example: *k*-NN classifier (***k* = 5** and **Euclidean distance**)



Advantages *k*-NN

Advantages:

- Easy to understand and implement: given its simplicity, it is one of the first classifiers that a new data scientist will learn
- Adapts easily: as new training samples are added, the algorithm adjusts to account for any new data since all training data are stored into memory
- Few hyperparameters: only requires a *k* value and a distance metric, which is low when compared to other machine learning models

Disadvantages *k*-NN

Disadvantages:

- Does not scale well: since *k*-NN is a lazy algorithm, it uses all the training data at the runtime and hence is slow
- Curse of dimensionality: the *k*-NN algorithm does not perform well with high-dimensional data
- Complexity: $O(n)$ for each instance to be classified
- Computationally expensive: it takes up more memory and data storage compared to other classifiers

The (k,l)-NN algorithm

Let l be a positive integer $\lceil k/2 \rceil < l \leq k$, a threshold for the majority in the voting of the k nearest neighbours

- This classifier consists of applying the k-NN model and make a decision based on:
  - if some class has received at least l votes, then assign the test sample to the most voted class
  - otherwise, reject the classification of the test sample (i.e., assign it to a dummy class, ω_0)
- This is a **classifier with reject option**

Application of *k*-NN: the IRIS data set

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa		
5.1	3.8	Setosa		
7.2	3.0	Virginica		
5.4	3.4	Setosa		
5.1	3.3	Setosa		
5.4	3.9	Setosa		
7.4	2.8	Virginica		
6.1	2.8	Versicolor		
7.3	2.9	Virginica		
6.0	2.7	Versicolor		
5.8	2.8	Virginica		
6.3	2.3	Versicolor		
5.1	2.5	Versicolor		
6.3	2.5	Versicolor		

Application of *k*-NN: the IRIS data set (ii)

Sepal Leght	Sepal Width	Species
5.2	3.1	?

Compute the distances between the given test sample and all training data:

$$d_2(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} =$$
$$= \sqrt{(5.2 - 5.3)^2 + (3.1 - 3.7)^2} = 0.608$$

Application of *k*-NN: the IRIS data set (iii)

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	
5.1	3.8	Setosa	0.707	
7.2	3.0	Virginica	2.002	
5.4	3.4	Setosa	0.360	
5.1	3.3	Setosa	0.220	
5.4	3.9	Setosa	0.820	
7.4	2.8	Virginica	2.220	
6.1	2.8	Versicolor	0.940	
7.3	2.9	Virginica	2.100	
6.0	2.7	Versicolor	0.890	
5.8	2.8	Virginica	0.670	
6.3	2.3	Versicolor	1.360	
5.1	2.5	Versicolor	0.600	
6.3	2.5	Versicolor	1.250	

Application of *k*-NN: the IRIS data set (iv)

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	**3**
5.1	3.8	Versicolor	0.707	**6**
7.2	3.0	Virginica	2.002	**12**
5.4	3.4	Setosa	0.360	**2**
5.1	3.3	Setosa	0.220	**1**
5.4	3.9	Setosa	0.820	**7**
7.4	2.8	Virginica	2.220	**14**
6.1	2.8	Versicolor	0.940	**9**
7.3	2.9	Virginica	2.100	**13**
6.0	2.7	Versicolor	0.890	**8**
5.8	2.8	Virginica	0.670	**5**
6.3	2.3	Versicolor	1.360	**11**
5.1	2.5	Versicolor	0.600	**4**
6.3	2.5	Versicolor	1.250	**10**

Find ranks

Application of *k*-NN: the IRIS data set (v)

Sepal Leght	Sepal Width	Species
5.2	3.1	?

Find the k nearest neighbours:

If k = 1	Rank = 1	Setosa
If k = 5	Rank = 1, 2, ..., 5	Setosa
If k = 10	Rank = 1, 2, ..., 10	Versicolor