Introduction to Classification: Naïve Bayes and Nearest Neighbour

Dr. Akinul Islam Jony

Assistant Professor

Dept. of Computer Science
Faculty of Science & Technology
American International University – Bangladesh

akinul@aiub.edu

Outline

- Introduction to Classification
- Naïve Bayes classification
- Nearest Neighbour Classification

Objectives & Outcomes

- To understand the concepts of classification
- To get familiar with two common classification methods:
  - Naïve Bayes
  - Nearest Neighbour
- To understand the advantages and disadvantages of Naïve Bayes and Nearest Neighbour classifiers

What is Classification?

- Classification is a task that occurs very frequently in everyday life.
- Essentially it involves dividing up objects so that each is assigned to one of a number of *mutually exhaustive and exclusive* categories known as *classes*.
- The term '*mutually exhaustive and exclusive*' simply means that each object must be assigned to precisely one *class*, i.e., never to more than one and never to no class at all.

What is Classification?

- Many practical decision-making tasks can be formulated as *classification*
- problems, i.e. , assigning people or objects to one of a number of categories, for example
  - customers who are likely to buy or not buy a particular product in a supermarket
  - people who are at high, medium or low risk of acquiring a certain illness
  - student projects worthy of a distinction, merit, pass or fail grade
  - houses that are likely to rise in value, fall in value or have an unchanged value in 12 months' time
  - people who are at high, medium or low risk of a car accident in the next 12 months
  - the likelihood of rain the next day for a weather forecast (very likely, likely,
  - unlikely, very unlikely).

What is Classification? (*Recall*)

- If the designated attribute is *categorical*, the task is called *classification*.
- Classification is one form of prediction, where the value to be predicted is a *label*.
- For example,
  - a hospital may want to classify medical patients into those who are at high, medium or low risk of acquiring a certain illness,
  - an opinion polling company may wish to classify people interviewed into those who are likely to vote for each of a number of political parties or are undecided, or
  - we may wish to classify a student project as distinction, merit, pass or fail.

Classification Methods (Classifiers)

- In classification, there are some classifiers (classification algorithms/methods) deal only with *categorical* attributes, some only with *continuous* attributes while some work on *both.*
- In this lecture, we will introduce two common classification algorithms/methods:
  - Naïve Bayes
  - Nearest Neighbour

Naïve Bayes Classifiers

- Naïve Bayes is a method of classification that does not use rules, a decision tree or any other explicit representation of the classifier.
- Rather, it uses the branch of Mathematics known as *probability theory* to find the most likely of the possible classifications.
  - *A detailed discussion of probability theory would be substantially outside the scope of this course*.
- It relies on all attributes being ***categorical***.
- The probability of an event:
  - Suppose an event **E** can happen in **r** ways out of a total of **n** possible equally likely ways
  - Then the probability of occurrence of the event (called its success) is denoted by:
    $P(E)=r/n$

Naïve Bayes Classifiers: Example

- A train arrival example, where four mutually exclusive and exhaustive events:
  - E1 – cancelled (train cancelled)
  - E2 – very late (train ten minutes or more late)
  - E3 – late (train less than ten minutes late)
  - E4 – on time (train on time or early).
- Mutually exclusive and exhaustive means that one and only one must always occur among the defined possible events.

Naïve Bayes Classifiers: Example

- Let's consider this table as a sample *train* dataset.
- Each row of the dataset is called an *instance*.
- An *instance* comprises the values of a number of attributes and the corresponding classification.
- That means, our training set consists of **20** instances, each recording the value of **four** attributes as well as the classification (**class**).
- The training set constitutes the results of a sample of trials that we can use to predict the classification of other (unclassified) instances.

day	season	wind	rain	class
weekday	spring	none	none	on time
weekday	winter	none	slight	on time
weekday	winter	none	slight	on time
weekday	winter	high	heavy	late
saturday	summer	normal	none	on time
weekday	autumn	normal	none	very late
holiday	summer	high	slight	on time
sunday	summer	normal	none	on time
weekday	winter	high	heavy	very late
weekday	summer	none	slight	on time
saturday	spring	high	heavy	cancelled
weekday	summer	high	slight	on time
saturday	winter	normal	none	late
weekday	summer	high	none	on time
weekday	winter	normal	heavy	very late
saturday	autumn	high	slight	on time
weekday	autumn	none	heavy	on time
holiday	spring	normal	slight	on time
weekday	spring	normal	none	on time
weekday	spring	normal	slight	on time

The ***train*** Dataset

Naïve Bayes Classifiers: Example

- The probability of an event is usually indicated by a capital letter *P* , so we might have
  - *P* (E1) = 0.05
  - *P* (E2) = 0.1
  - *P* (E3) = 0.15
  - *P* (E4) = 0.7
  - (Read as 'the probability of event E1 is 0.05' etc.)
- Each of these probabilities is between 0 and 1 inclusive, as it has to be to qualify as a probability.
- They also satisfy a second important condition: the sum of the four probabilities has to be 1, because precisely one of the events must always occur. In this case
  - *P* (E1) + *P* (E2) + *P* (E3) + *P* (E4) = 1
- In general, the sum of the probabilities of a set of *mutually exclusive and exhaustive* events must always be 1.

Naïve Bayes Classifiers: Example

- Now, if we want,
  - How should we use probabilities to find the most likely classification for an unseen instance such as the one below?

weekday	winter	high	heavy	????

Naïve Bayes Classifiers: Example

- One straightforward (but flawed) way is just to look at the frequency of each of the classifications in the training set and choose the most common one.
- In this case the most common classification is ***on time***, so we would choose that.
- The flaw in this approach is that all unseen instances will be classified in the same way, in this case as ***on time***.
- Therefore, a more sophisticated approach is required.
- The instances in the training set record not only the ***classification*** but also the values of four attributes: ***day***, ***season***, ***wind*** and ***rain***. Presumably, they are recorded because we believe that in some way the values of the four attributes affect the outcome.
- To make effective use of the additional information represented by the attribute values we first need to introduce the notion of ***conditional probability***.

Naïve Bayes Classifiers: Example

- Prior probability:
  - The probability of the train being on time, calculated using the frequency of ***on time*** in the training set divided by the total number of instances is known as the ***prior probability***.
  - In this case P (class = on time) = 14/20 = 0.7.
  - If we have no other information this is the best, we can do. If we have other (relevant) information, the position is different.
- What is the probability of the train being on time if we know that the season is ***winter***?
  - We can calculate this as the number of times ***class = on time and season = winter*** (in the same instance), divided by the number of times the season is ***winter***, which comes to ***2/6 = 0.33***. This is considerably less than the prior probability of ***0.7*** and seems intuitively reasonable. Trains are less likely to be on time in winter.
  - That's where the ***conditional probability*** comes in the scene.

Naïve Bayes Classifiers: Example

- Conditional probability:
  - The probability of an event occurring if we know that an attribute has a particular value (or that several variables have particular values) is called the ***conditional probability*** of the event occurring and is written as, e.g.

    P (class = on time | season = winter).
  - The vertical bar can be read as 'given that', so the whole term can be read as 'the probability that the class is *on time* given that the season is *winter*'.
  - It is also called a ***posterior probability***.
- *Posterior probability* is the probability that we can calculate for the classification ***after*** we have obtained the information that the season is winter. By contrast, the *prior probability* is that estimated ***before*** any other information is available.

Naïve Bayes Classifiers: Example

- To calculate the most likely classification for the following instance

weekday	winter	high	heavy	????

  - we could calculate the probability of
    - P (class = on time | day = weekday and season = winter and wind = high and rain = heavy)
  - and do similarly for the other three possible classifications.
    - P (class = very late| day = weekday and season = winter and wind = high and rain = heavy)
    - P (class = late | day = weekday and season = winter and wind = high and rain = heavy)
    - P (class = cancelled | day = weekday and season = winter and wind = high and rain = heavy)
  - However, there are only two instances in the training set with that combination of attribute values and basing any estimates of probability on these is unlikely to be helpful.

Naïve Bayes Classifiers: Example

- Therefore, to obtain a reliable estimate of the four classifications a more indirect approach is needed. We could start by using ***conditional probabilities*** based on a **single attribute**.
  - For the train dataset
    - P (class = on time | rain = heavy) = 1/5 = 0.2
    - P (class = late | rain = heavy) = 1/5 = 0.2
    - P (class = very late | rain = heavy) = 2/5 = 0.4
    - P (class = cancelled | rain = heavy) = 1/5 = 0.2
  - The third of these has the largest value, so we could conclude that the most likely classification is ***very late***, a different result from using the *prior probability* as before.
- We could do a similar calculation with attributes ***day***, ***season*** and ***wind***. This might result in other classifications having the largest value.
- **Which is the best one to take?**

Naïve Bayes Classifiers: Example

- The **Naïve Bayes** algorithm gives us a way of combining the *prior probability* and *conditional probabilities* in a single formula, which we can use to calculate the probability of each of the possible classifications in turn. Having done this, we choose the classification with the largest value.
- The method uses *conditional probabilities*, but the other way round from before.
  - (This may seem a strange approach but is justified by the method that follows, which is based on a well-known Mathematical result known as ***Bayes Rule***.)

Naïve Bayes Classifiers: Example

- For example,
  - instead of the probability that the class is *very late* given that the season is *winter*, i.e.,

    ***P* (class = very late | season = winter)**,
  - we use the *conditional probability* that the season is *winter* given that the class is *very late*, i.e.,

    ***P* (season = winter | class = very late)**.
  - We can calculate this as the number of times that **season = winter and class = very late** occur in the same instance, divided by the number of instances for which the class is ***very late***.
- In a similar way we can calculate other *conditional probabilities*, for example,

  ***P* (rain = none | class = very late)**.

Naïve Bayes Classifiers: Example

- For the ***train*** data we can tabulate all the ***conditional*** and ***prior*** probabilities as shown in this table.
- For example,
  - the conditional probability ***P* (day = weekday|class = on time)** is the number of instances in the train dataset for which **day = weekday and class = on time** (i.e., **9**), divided by the total number of instances for which **class = on Time** (i.e., **14**). So, the conditional probability is **9/14 = 0.64**.
  - The prior probability ***P* (class = very late)** is the number of instances in the dataset for which **class = very late** divided by the total number of instances, i.e., **3/20 = 0.15**.
- We can now use these values to calculate the probabilities of real interest to us.

	class = on time	class = late	class = very late	class = cancelled
day = weekday	**9/14 = 0.64**	**1/2 = 0.5**	**3/3 = 1**	**0/1 = 0**
day = saturday	2/14 = 0.14	1/2 = 0.5	0/3 = 0	1/1 = 1
day = sunday	1/14 = 0.07	0/2 = 0	0/3 = 0	0/1 = 0
day = holiday	2/14 = 0.14	0/2 = 0	0/3 = 0	0/1 = 0
season = spring	4/14 = 0.29	0/2 = 0	0/3 = 0	1/1 = 1
season = summer	6/14 = 0.43	0/2 = 0	0/3 = 0	0/1 = 0
season = autumn	2/14 = 0.14	0/2 = 0	1/3 = 0.33	0/1 = 0
season = winter	**2/14 = 0.14**	**2/2 = 1**	**2/3 = 0.67**	**0/1 = 0**
wind = none	5/14 = 0.36	0/2 = 0	0/3 = 0	0/1 = 0
wind = high	**4/14 = 0.29**	**1/2 = 0.5**	**1/3 = 0.33**	**1/1 = 1**
wind = normal	5/14 = 0.36	1/2 = 0.5	2/3 = 0.67	0/1 = 0
rain = none	5/14 = 0.36	1/2 =0.5	1/3 = 0.33	0/1 = 0
rain = slight	8/14 = 0.57	0/2 = 0	0/3 = 0	0/1 = 0
rain = heavy	**1/14 = 0.07**	**1/2 = 0.5**	**2/3 = 0.67**	**1/1 = 1**
Prior Probability	**14/20 = 0.70**	**2/20 = 0.10**	**3/20 = 0.15**	**1/20 = 0.05**

Conditional and Prior Probabilities: *train* Dataset

Naïve Bayes Classifiers: Example

- These are the posterior probabilities of each possible class occurring for a specified instance, i.e., for known values of all the attributes.
- We can calculate these posterior probabilities using the method ***Naïve Bayes*** Classification.

Naïve Bayes Classifiers

Naïve Bayes Classification

Given a set of k mutually exclusive and exhaustive classifications $c_1, c_2, \ldots, c_k$, which have prior probabilities $P(c_1), P(c_2), \ldots, P(c_k)$, respectively, and n attributes $a_1, a_2, \ldots, a_n$ which for a given instance have values $v_1, v_2, \ldots, v_n$ respectively, the posterior probability of class c_i occurring for the specified instance can be shown to be proportional to

$$P(c_i) \times P(a_1 = v_1 \text{ and } a_2 = v_2 \ldots \text{ and } a_n = v_n \mid c_i)$$

Making the assumption that the attributes are independent, the value of this expression can be calculated using the product

$$\boldsymbol{P(c_i) \times P(a_1 = v_1 \mid c_i) \times P(a_2 = v_2 \mid c_i) \times \ldots \times P(a_n = v_n \mid c_i)}$$

We calculate this product for each value of i from 1 to k and choose the classification that has the largest value.

The Naïve Bayes Classification Algorithm

Naïve Bayes Classifiers

- The formula shown in bold for ***Naïve Bayes*** classification combines the *prior* probability of $\boldsymbol{c_i}$ with the values of the ***n*** possible *conditional* probabilities involving a test on the value of a single attribute.
- It is often written as

$$P(c_i) \times \prod_{j=1}^{n} P(a_j = v_j \mid class = c_i).$$

Note that the Greek letter $\prod$ (pronounced pi) in the above formula is not connected with the mathematical constant 3.14159.... It indicates the product obtained by multiplying together the n values $P(a_1 = v_1 \mid c_i)$, $P(a_2 = v_2 \mid c_i)$ etc.

Naïve Bayes Classifiers

- When using the ***Naïve Bayes*** method to classify a series of unseen instances the most efficient way to start is by calculating all the ***prior*** probabilities and, also all the ***conditional*** probabilities involving one attribute, though not all of them may be required for classifying any particular instance.

Naïve Bayes Classifiers: Example

- Now, by using the ***prior*** probabilities and the ***conditional*** probabilities values for the ***train*** dataset, we obtain the following posterior probabilities for each possible classification for the unseen instance:

weekday	winter	high	heavy	????

  - **class = on time**
    - 0.70 × 0.64 × 0.14 × 0.29 × 0.07 = 0.0013
  - **class = late**
    - 0.10 × 0.50 × 1.00 × 0.50 × 0.50 = 0.0125
  - **class = very late**
    - 0.15 × 1.00 × 0.67 × 0.33 × 0.67 = 0.0222
  - **class = cancelled**
    - 0.05 × 0.00 × 0.00 × 1.00 × 1.00 = 0.0000
  - The largest value is for **class = very late**.

Note that the four values calculated are not themselves probabilities, as they do not sum to **1**. However, Each value can be '***normalised***' to a valid posterior probability simply by dividing it by the sum of all four values.

In practice, we are interested only in finding the largest value, so the ***normalisation*** step is not necessary.

Naïve Bayes Classifiers: Advantages

- Easy to implement
- Good results obtained in most of the cases

Naïve Bayes Classifiers: Disadvantages

- The most obvious problem is that it relies on all attributes being *categorical*.
  - In practice, many datasets have a combination of categorical and continuous attributes, or even only continuous attributes.
  - This problem can be overcome by converting the continuous attributes to categorical ones.
- A second problem is that estimating probabilities by relative frequencies can give a poor estimate if the number of instances with a given attribute/value combination is small.
  - In the extreme case where it is zero, the posterior probability will inevitably be calculated as zero. This happened for **class = cancelled** in the train example.
- Another problem is that Naive Bayes assumes that all variables (or predictors) are independent (rarely happening in real life). That means, dependencies among variables cannot be modeled by Naïve Bayes Classifier.
  - This limits the applicability of this algorithm in real-world use cases. Because, practically, in most cases dependencies exist among variables.

Naïve Bayes Classifiers: Avoiding the Zero-Probability Problem

- Prediction using Naïve Bayes requires each *conditional probability* be **non-zero**. Otherwise, the predicted probability will be **zero**.
  - For example:
    - Suppose a dataset with 1000 tuples, income = low (0), income = medium (990), and income = high (10)
- Use *Laplacian correction* (or *Laplacian estimator*)
  - Adding 1 to each case
    - Prob(income = low) = 1/1003
    - Prob(income = medium) = 991/1003
    - Prob(income = high) = 11/1003
  - The "corrected" probability estimates are close to their "uncorrected" counterparts.

Nearest Neighbour Classification

- Nearest Neighbour classification is mainly used when all attribute values are continuous, although it can be modified to deal with categorical attributes.
- The idea is to estimate the classification of an unseen instance using the classification of the instance or instances that are closest to it, in some sense that we need to define.

Nearest Neighbour Classification

- Supposing we have a training set with just two instances such as the following.

a	b	c	d	e	f	Class
yes	no	no	6.4	8.3	low	negative
yes	yes	yes	18.2	4.7	high	positive

six attribute values, followed by a classification (positive or negative).

- We are then given a third instance.

yes	no	no	6.6	8.0	low	????

- **What should its classification be?**
  - Even without knowing what the six attributes represent, it seems intuitively obvious that the unseen instance is nearer to the first instance than to the second.
  - In the absence of any other information, we could reasonably predict its classification using that of the first instance, i.e., as '**negative**'.

Nearest Neighbour Classification

- In practice there are likely to be many more instances in the training set, but the same principle applies.
- It is usual to base the classification on those of the ***k*** nearest neighbours (where ***k*** is a small integer such as 3 or 5), not just the nearest one.
- The method is then known as *k-Nearest Neighbour* or just *k-NN classification.*

Basic k-Nearest Neighbour Classification Algorithm

– Find the k training instances that are closest to the unseen instance.

– Take the most commonly occurring classification for these k instances.

Nearest Neighbour Classification

- We can illustrate *k-NN* classification diagrammatically when the dimension (i.e., the number of attributes) is small.
- Next, we will explain an example which illustrates the case where the dimension is just **2**.
- However, in real-world data mining applications it can of course be considerably larger.

Nearest Neighbour Classification

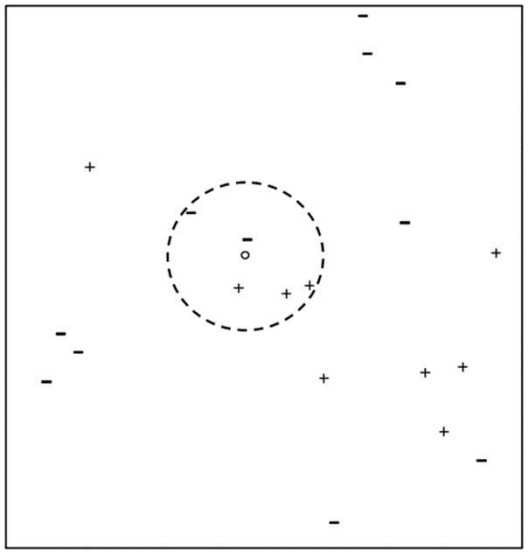
- A training set with 20 instances, each giving the values of two attributes and an associated classification.
- How can we estimate the classification for an 'unseen' instance where the first and second attributes are **9.1** and **11.0**, respectively?

Attribute 1	Attribute 2	Class
0.8	6.3	−
1.4	8.1	−
2.1	7.4	−
2.6	14.3	+
6.8	12.6	−
8.8	9.8	+
9.2	11.6	−
10.8	9.6	+
11.8	9.9	+
12.4	6.5	+
12.8	1.1	−
14.0	19.9	−
14.2	18.5	−
15.6	17.4	−
15.8	12.2	−
16.6	6.7	+
17.4	4.5	+
18.2	6.9	+
19.0	3.4	−
19.6	11.1	+

Training Set for *k*-Nearest Neighbour Example

Nearest Neighbour Classification

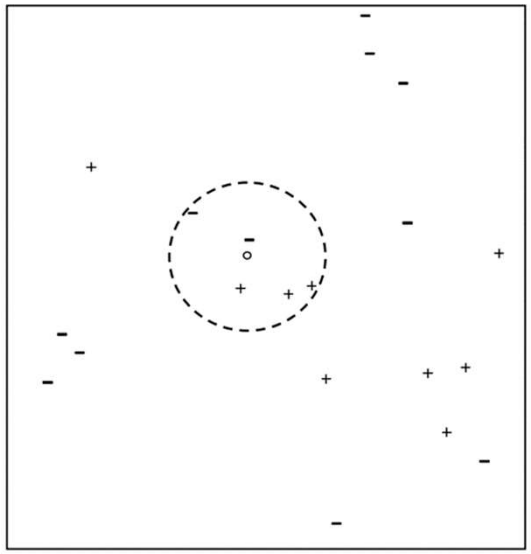
- For this small number of attributes, we can represent the training set as 20 points on a two-dimensional graph with values of the first and second attributes measured along the horizontal and vertical axes, respectively.
- Each point is labelled with a **+** or **–** symbol to indicate that the classification is positive or negative, respectively.
- The result is shown in the figure (a two-dimensional Representation of the training Data).



Two-dimensional Representation of Training Set for *k*-Nearest Neighbour Example

Nearest Neighbour Classification

- A circle has been added to enclose the five nearest neighbours of the unseen instance, which is shown as a small circle close to the centre of the larger one.
- The five nearest neighbours are labelled with three **+** signs and two **–** signs.
- So, a basic *5-NN* classifier would classify the unseen instance as '**positive**' by a form of majority voting.
- There are other possibilities, for example the 'votes' of each of the k nearest neighbours can be weighted, so that the classifications of closer neighbours are given greater weight than the classifications of more distant ones. We will not pursue this here.



Two-dimensional Representation of Training Set for *k*-Nearest Neighbour Example

Nearest Neighbour Classification

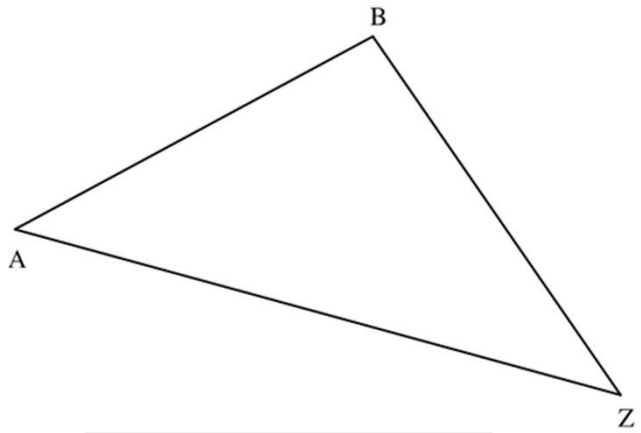
- We can represent two points in two dimensions ('in two-dimensional space' is the usual term) as (a_1, a_2) and (b_1, b_2) and visualise them as points in a plane.
- When there are three attributes we can represent the points by (a_1, a_2, a_3) and (b_1, b_2, b_3).
- When there are ***n*** attributes, we can represent the instances by the points $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$ in '***n***-dimensional space'.
- As the number of dimensions (attributes) increases it rapidly becomes impossible to visualise them.

Distance Measures

- There are many possible ways of measuring the distance between two instances with ***n*** attribute values, or equivalently between two points in ***n***-dimensional space.
- The notation **dist**(*X*, *Y*) is used to denote the distance between two points *X* and *Y*.
- Usually, distance measurement imposes three requirements.
  - The distance of any point *A* from itself is zero, i.e., **dist(*A*, *A*) = 0**.
  - The distance from *A* to *B* is the same as the distance from *B* to *A*, i.e.,
    dist(A, B) = dist(B, A) (the symmetry condition).
  - The third condition is called the *triangle inequality*. It corresponds to the intuitive idea that 'the shortest distance between any two points is a straight line'. The condition says that for any points *A*, *B* and *Z*:
    dist(A, B) ≤ dist(A, Z) + dist(Z, B).

Distance Measures

- The equality only occurs if *Z* is the same point as *A* or *B* or is on the direct route between them.



The Triangle Inequality

Distance Measures

- There are many possible distance measures:
  - Euclidean Distance
  - Manhattan Distance or City Block Distance
  - Maximum Dimension Distance

Distance Measures: Euclidean Distance

- If we denote an instance in the training set by (a_1, a_2) and the unseen instance by (b_1, b_2) the length of the straight line joining the points is

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

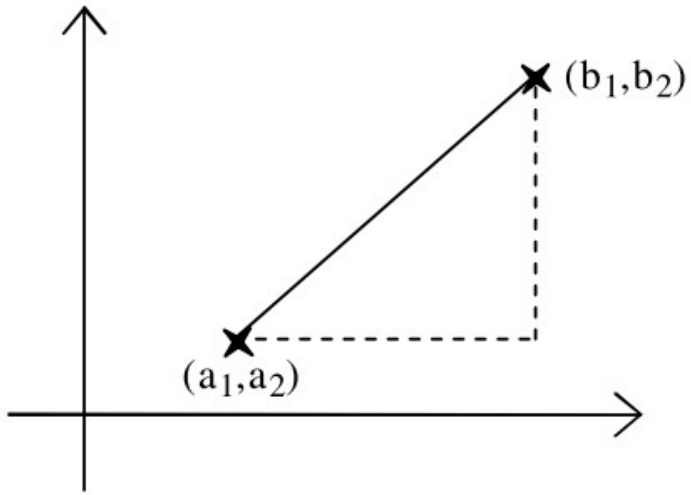
- If there are two points (a_1, a_2, a_3) and (b_1, b_2, b_3) in a three-dimensional space the corresponding formula is

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2}$$

- The formula for *Euclidean distance* between points (a_1, a_2, . . . , a_n) and (b_1, b_2, . . . , b_3) in ***n***-dimensional space is a generalisation of these two results.
- The *Euclidean distance* is given by the formula

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

Distance Measures: Euclidean Distance



Example of Euclidean Distance

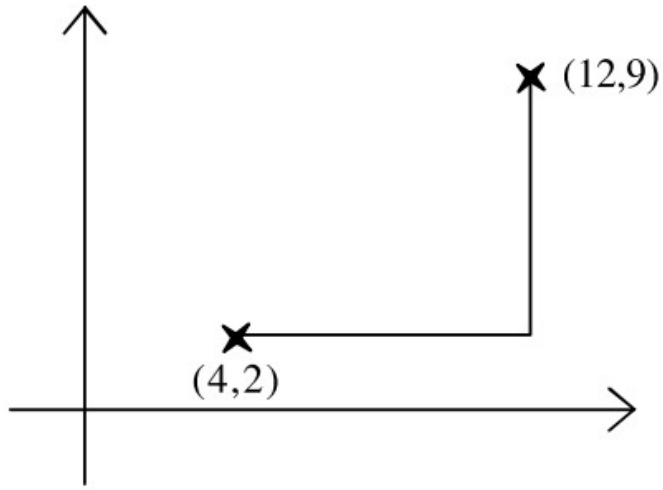
Distance Measures: Manhattan Distance

- Another measure sometimes used is called *Manhattan Distance* or *City Block Distance*.
- The analogy is with travelling around a city such as Manhattan, where you cannot (usually) go straight from one place to another but only by moving along streets aligned horizontally and vertically.

Distance Measures: Manhattan Distance

- The *City Block* distance between the points (4, 2) and (12, 9) is

  (12 – 4) + (9 – 2) = 8 + 7 = 15.



Example of City Block Distance

Distance Measures: Maximum Dimension Distance

- A third possibility is the *maximum dimension distance*.
- This is the largest absolute difference between any pair of corresponding attribute values. (The absolute difference is the difference converted to a positive number if it is negative.)
- For example, the *maximum dimension distance* between the following two instances:

6.2	−7.1	−5.0	18.3	−3.1	8.9

8.3	12.4	−4.1	19.7	−6.2	12.4

12.4 – (–7.1) = 19.5

Normalisation

- A major problem when using the *Euclidean distance* formula (and many other distance measures) is that the large values frequently swamp the small ones.
- Suppose that two instances are as follows for some classification problem associated with cars (the classifications themselves are omitted).

Mileage (miles)	Number of doors	Age (years)	Number of owners
18,457	2	12	8
26,292	4	3	1

- When the distance of these instances from an unseen one is calculated, the ***mileage*** attribute will almost certainly contribute a value of several thousands squared, i.e. several millions, to the sum of squares total. The number of doors will probably contribute a value less than 10.
- It is clear that in practice the only attribute that will matter when deciding which neighbours are the nearest using the *Euclidean distance* formula is the *mileage*.
- We could have chosen an alternative measure of distance travelled such as *millimetres* or perhaps *light years*. Similarly we might have measured *age* in some other unit such as *milliseconds* or *millennia*. The units chosen should not affect the decision on which are the nearest neighbours.

Normalisation

- To overcome this problem we generally *normalise* the values of continuous attributes.
- The idea is to make the values of each attribute run from 0 to 1.
- In general if the lowest value of attribute *A* is *min* and the highest value is *max*, we convert each value of *A*, say *a*, to (*a* − *min*)/(*max* − *min*).
- Using this approach all continuous attributes are converted to small numbers from 0 to 1, so the effect of the choice of unit of measurement on the outcome is greatly reduced.
- Note that it is possible that an unseen instance may have a value of *A* that is less than *min* or greater than *max*. If we want to keep the adjusted numbers in the range from 0 to 1, we can just convert any values of *A* that are less than *min* or greater than *max* to 0 or 1, respectively.

Normalisation

- Another issue that occurs with measuring the distance between two points is the *weighting* of the contributions of the different attributes.
- We may believe that the mileage of a car is more important than the number of doors it has.
- To achieve this, we can adjust the formula for Euclidean distance to

$$\sqrt{w_1(a_1 - b_1)^2 + w_2(a_2 - b_2)^2 + \ldots + w_n(a_n - b_n)^2}$$

  - where w_1, w_2, . . . , w_n are the weights. It is customary to scale the weight values so that the sum of all the weights is one.

Dealing with Categorical Attributes

- One of the weaknesses of the nearest neighbour approach to classification is that there is no entirely satisfactory way of dealing with *categorical* attributes.
- One possibility is to say that the difference between any two identical values of the attribute is *zero* and that the difference between any two different values is 1.
  - Effectively this amounts to saying (for a *colour* attribute)
    red − red = 0, red − blue = 1, blue − green = 1, etc.
- Sometimes there is an ordering (or a partial ordering) of the values of an attribute, for example we might have values good, average and bad.
  - We could treat the difference between good and average or between average and bad as 0.5 and the difference between good and bad as 1.
  - This still does not seem completely right but may be the best we can do in practice.

Weka Tool

- Get familiar with the Weka tool
  - an open-source machine learning software
  - https://www.cs.waikato.ac.nz/ml/weka/

Exercises

- Using the Naïve Bayes classification algorithm with the *train* dataset, calculate the most likely classification for the following unseen instances.

weekday	summer	high	heavy	????
sunday	summer	normal	slight	????

The ***train*** Dataset

day	season	wind	rain	class
weekday	spring	none	none	on time
weekday	winter	none	slight	on time
weekday	winter	none	slight	on time
weekday	winter	high	heavy	late
saturday	summer	normal	none	on time
weekday	autumn	normal	none	very late
holiday	summer	high	slight	on time
sunday	summer	normal	none	on time
weekday	winter	high	heavy	very late
weekday	summer	none	slight	on time
saturday	spring	high	heavy	cancelled
weekday	summer	high	slight	on time
saturday	winter	normal	none	late
weekday	summer	high	none	on time
weekday	winter	normal	heavy	very late
saturday	autumn	high	slight	on time
weekday	autumn	none	heavy	on time
holiday	spring	normal	slight	on time
weekday	spring	normal	none	on time
weekday	spring	normal	slight	on time

Exercises

- Using the training set (as shown in the Table), calculate the 5-nearest neighbours of the instance with first and second attributes **9.1** and **11.0**, respectively.

Attribute 1	Attribute 2	Class
0.8	6.3	−
1.4	8.1	−
2.1	7.4	−
2.6	14.3	+
6.8	12.6	−
8.8	9.8	+
9.2	11.6	−
10.8	9.6	+
11.8	9.9	+
12.4	6.5	+
12.8	1.1	−
14.0	19.9	−
14.2	18.5	−
15.6	17.4	−
15.8	12.2	−
16.6	6.7	+
17.4	4.5	+
18.2	6.9	+
19.0	3.4	−
19.6	11.1	+

Training Set for *k*-Nearest Neighbour Example

Assignment

- Use the *train* dataset for Building and evaluating Naive Bayes classifier with WEKA/Python/Other tools (that you find most suitable for you).
  - Prepare data for classification
  - Building a Naive Bayes model
  - Evaluate classifier with the test set

day	season	wind	rain	class
weekday	spring	none	none	on time
weekday	winter	none	slight	on time
weekday	winter	none	slight	on time
weekday	winter	high	heavy	late
saturday	summer	normal	none	on time
weekday	autumn	normal	none	very late
holiday	summer	high	slight	on time
sunday	summer	normal	none	on time
weekday	winter	high	heavy	very late
weekday	summer	none	slight	on time
saturday	spring	high	heavy	cancelled
weekday	summer	high	slight	on time
saturday	winter	normal	none	late
weekday	summer	high	none	on time
weekday	winter	normal	heavy	very late
saturday	autumn	high	slight	on time
weekday	autumn	none	heavy	on time
holiday	spring	normal	slight	on time
weekday	spring	normal	none	on time
weekday	spring	normal	slight	on time

The ***train*** Dataset

Assignment

- Use the *degrees* dataset for Building and evaluating Naive Bayes classifier with WEKA/Python/Other tools (that you find most suitable for you).
  - Prepare data for classification
  - Building a Naive Bayes model
  - Evaluate classifier with the test set

SoftEng	ARIN	HCI	CSA	Project	Class
A	B	A	B	B	SECOND
A	B	B	B	A	FIRST
A	A	A	B	B	SECOND
B	A	A	B	B	SECOND
A	A	B	B	A	FIRST
B	A	A	B	B	SECOND
A	B	B	B	B	SECOND
A	B	B	B	B	SECOND
A	A	A	A	A	FIRST
B	A	A	B	B	SECOND
B	A	A	B	B	SECOND
A	B	B	A	B	SECOND
B	B	B	B	A	SECOND
A	A	B	A	B	FIRST
B	B	B	B	A	SECOND
A	A	B	B	B	SECOND
B	B	B	B	B	SECOND
A	A	B	A	A	FIRST
B	B	B	A	A	SECOND
B	B	A	A	B	SECOND
B	B	B	B	A	SECOND
B	A	B	A	B	SECOND
A	B	B	B	A	FIRST
A	B	A	B	B	SECOND
B	A	B	B	B	SECOND
A	B	B	B	B	SECOND

The *degrees* dataset

Assignment

- Use the dataset for Building and evaluating KNN classifier with WEKA/Python/Other tools (that you find most suitable for you).
- Apply different distance measure formula and compare the outcome with different value for *K* as well.

Attribute 1	Attribute 2	Class
0.8	6.3	−
1.4	8.1	−
2.1	7.4	−
2.6	14.3	+
6.8	12.6	−
8.8	9.8	+
9.2	11.6	−
10.8	9.6	+
11.8	9.9	+
12.4	6.5	+
12.8	1.1	−
14.0	19.9	−
14.2	18.5	−
15.6	17.4	−
15.8	12.2	−
16.6	6.7	+
17.4	4.5	+
18.2	6.9	+
19.0	3.4	−
19.6	11.1	+

Training Set for *k*-Nearest Neighbour Example

Reference

- Max Bramer, “Chapter 3: Introduction to Classification: Naïve Bayes and Nearest Neighbour”, *Principles of Data Mining* (4th Edition).