COMP9033 DATA ANALYTICS

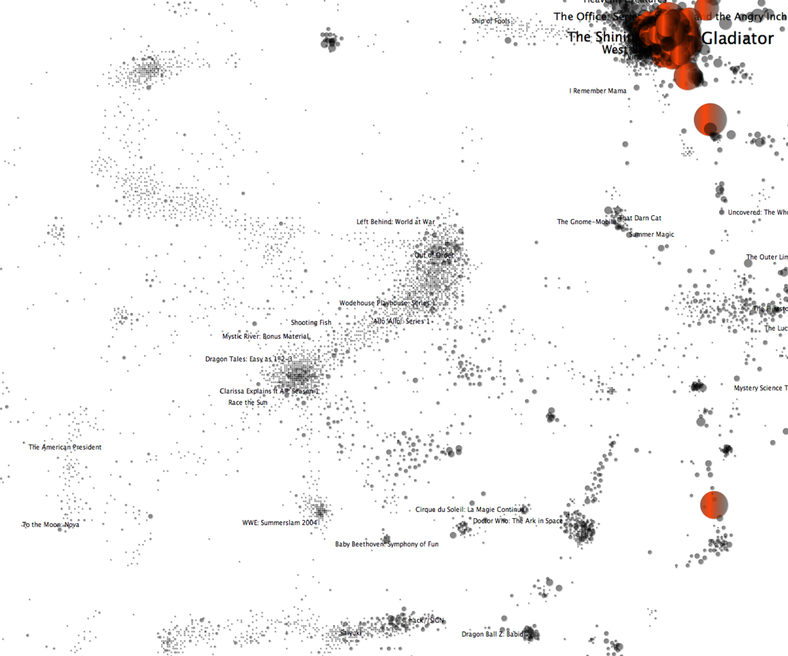
8/12

NEAREST NEIGHBOURS

DR. DONAGH HORGAN

DEPARTMENT OF COMPUTER SCIENCE
CORK INSTITUTE OF TECHNOLOGY

2018.03.20



Credit: Todd Holloway / Netflix

Overview

1. Measuring classifier error:
    - Confusion matrices.
    - Accuracy.
    - Precision.
    - Recall.
2. Decision trees:
    - Tree structures.
    - Classification trees.
    - Regression trees.
3. The CART algorithm:
    - How to build a decision tree.
    - Impurity measures.
    - Stopping criteria.
4. Advanced techniques:
    - Pruning.
    - Random forests.

1. Recommender systems:
    - The long tail phenomenon.
    - Content-based recommenders.
    - Collaborative filters.
    - Advantages and disadvantages.
    - The Netflix Prize.

2. Nearest neighbours:
    - k nearest neighbours.
    - Regression.
    - Classification
    - Hyperparameters.
    - Advantages and disadvantages.

Recommender systems

- Until relatively recent times, physical systems have dominated our lives, *e.g.*
  - We bought our goods in physical stores.
  - We got our news by reading physical newspapers.
- However, in the past twenty years, things have changed significantly:
  - We buy goods online.
  - We get news, audio and video from online media sources.

- Much of this change has been driven by the limitations of physical systems, *e.g.*
  - Shelf space is limited in a physical store.
  - Page space is limited in a physical newspaper.
- Virtual systems usually don't suffer from these limitations, *e.g.*
  - Products can be stored in different physical locations, but presented to us in a single location.
  - Webpages can be dynamically scaled to fit any content size.

- As a result, book shops only stock the most popular books, video stores only stock the most popular titles, *etc.*, while online retailers stock an enormous variety of goods.
- This is known as the long tail phenomenon, as illustrated to the right.
- However, the long tail phenomenon creates a whole new problem: with such huge variety, how can users find items that are relevant to them?

- One solution is the use of *recommender systems*, *i.e.* systems that can make recommendations to users about new items based on information about other users and items.
- By exploiting items' properties and/or hidden relationships in users' behaviour, we can determine which items are the most relevant and present these to the user.

- A *recommender system* is a predictive tool that makes recommendations about items based on the preferences of similar users and/or the properties of similar items.
- There are two main varieties:
    1. Content-based.
    2. Collaborative filtering.
- Content-based recommenders determine similarity by comparing the properties of an item to those of other items.
- Collaborative filtering determines similarity of items by examining the similarity of their ratings from users.

- Content-based systems recommend items based on the similarity of their *properties*, *e.g.*
    - They might recommend Star Wars if you have watched Indiana Jones because they were both written by George Lucas.
    - They might recommend Johnny Cash if you have listened to Hank Williams because they are both classified as country music.
    - They might recommend Nike runners if you have bought Adidas runners because they have similar properties.

- For a given item, content-based recommendation works as follows:
    1. The properties of the item are profiled and added to a catalogue or database of item properties.
    2. The properties of the item are then compared to the properties of every other item in the database and a similarity measure is computed.
    3. The most similar matches for the item are identified and stored.
    4. When the item is selected by a user, the matches are retrieved and shown as recommendations.

Advantages	Disadvantages
• They can recommend completely new items, *e.g.* new films/products. • As properties of items are generally fixed, the recomputation of similarity is only required when new items are added to the system.	• The properties of each item must be profiled, which can be time consuming. • If an item has a small number of properties, this may limit the identification of similar items. • User preferences are not taken into account, and so items are recommended based *only* on the similarity of their properties.

- Collaborative filtering systems recommend items based on user preferences.
- They come in two varieties:
    1. User-based filters: recommend items based on the similarity of your preferences to those of other users.
    2. Item-based filters: recommend items based on the similarity of their ratings by other users.
- A user-based filter might recommend Johnny Cash because you like the same variety of music as users who also like Johnny Cash.
- An item-based filter might recommend Star Wars because you liked Indiana Jones and users who have seen both, like both.
- Both kinds can be implemented via the k nearest neighbours algorithm.

- User-based filtering works as follows:
    1. Select a target user and a target item (*e.g.* an unrated film) to make a prediction for.
    2. Compute a measure of the *similarity* between the target user's item ratings and other users' ratings of those same items.
    3. Compute a prediction for the target user's rating for the target item as the weighted average/mode of the k most similar users' ratings for that item, using the similarity measures computed in Step 2 as weights.

- Item-based filtering works as follows:
    1. Select a target user and a target item (*e.g.* an unrated film) to make a prediction for.
    2. Compute a measure of the *similarity* between the target item's ratings and the ratings of every other item.
    3. Compute a prediction for the target user's rating for the target item as the weighted average/mode of the target user's ratings for the k most similar items, using the similarity measures computed in Step 2 as weights.

1.12 / USER-BASED VS ITEM-BASED FILTERING

User-based	Item-based
• Can suffer from small sample sizes, as most pairs of users have few items in common. • Regular retraining is usually required, as the similarity between pairs of users can change significantly over time. • Once user similarities have been measured, an arbitrary number of predictions can be made.	• Not as susceptible to small sample size issues, as similar items tend to have many users in common. • The similarity between items tends to change less frequently than the similarity between users, and so retraining is required less regularly. • Item-based filtering must be repeated for each prediction.

- Unlike content-based recommenders, collaborative filtering *requires* user ratings.
- Generally, we can get these in two ways:
  1. Ask users to rate items, *e.g*. IMDB, Amazon, Airbnb.
  2. Observe user behaviour and infer preferences, *e.g*. Google, Facebook, Spotify.
- Asking users to rate items can be effective if many users rate items, but some users can be unwilling to participate, which may lead to sampling bias.
- Inferring preferences doesn't require users to rate items — instead, user actions are interpreted as a "like" (*e.g*. listening to a song, watching a video).
- However, there is usually no way to distinguish inaction from "dislikes".

ADVANTAGES	DISADVANTAGES
• Like content-based systems, they can recommend completely new items. • As the recommendations are based on user preferences, they generally correlate with the perceived quality of items.	• Items that haven't been rated cannot be recommended. • Niche items don't get recommended as often as mainstream items (*i.e.* there is a popularity bias). • Recommendations can be poor when there is not a *reasonable* amount of *reliable* data available.

- In 2006, Netflix offered a million dollar prize for the first algorithm to beat its own movie recommender by more than 10%.
- Netflix provided 100 million ratings for nearly 500,000 users.
- Netflix's own algorithm was beaten in less than a week, but it took nearly three years to beat it by 10%.
- In the end, Netflix didn't implement the winning algorithm because of the effort required!
- Sometimes, increases in accuracy are outweighed by the cost of improvement.

Nearest neighbours

- *k* nearest neighbours is supervised machine learning algorithm that can be used to build *both* classification and regression models.
- Predictions are made based on the most similar records in the data, *e.g.*
  - Application memory usage at 15:00 might be estimated to be the average of the usage between 14:00 and 16:00 (two nearest neighbours).
  - A new credit approval application might be accepted or rejected based on the outcomes of the ten most similar applications seen previously.
  - How much a user might like the film Titanic can be estimated as the average of what the hundred users with the closest taste in films thought of that film.
- *k* nearest neighbours is commonly used to build collaborative filters, though has applications in many other domains too.

- The k nearest neighbours algorithm works as follows:
  1. Select a single unlabelled data record for which a target value is to be predicted, *i.e.* a record consisting of explanatory/feature variables only.
  2. Measure the *similarity* of the selected record to every other known record.
  3. Determine the k most similar records to the candidate record.
  4. Compute a prediction by taking a weighted measure of the target variable values of the k most similar records, using the similarity values as weights.
- The measure used in Step 4 depends on the data type of the target variable:
  - Continuous-valued / regression: weighted average.
  - Categorical / classification: weighted mode.

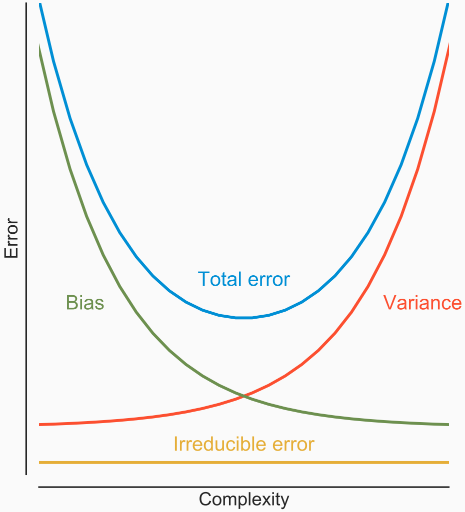
- The *weighted average* is a variation on the average:
  - Multiplicative weights are applied to each data point in the sample.
  - By choosing different weight values, we can increase or decrease the effect of individual points on the final result.
- The weighted average of the sample X is defined as

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}, \tag{8.1}$$

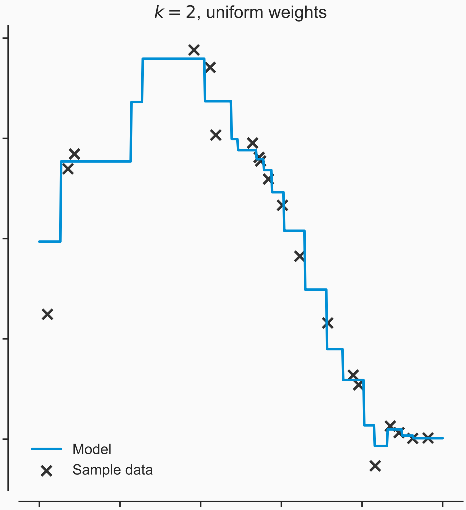
  where w_i denotes the i^{th} weight.
- When all the weights are equal to one (*i.e.* $w_i = 1, \forall\, i \in \{1, 2, \ldots, n\}$), the weighted average is equivalent to the average.

- The *weighted mode* is a variation on the mode:
  - Weights are assigned to each data point in the sample.
  - By choosing different weight values, we can increase or decrease the effect of individual points on the final result.
- The weighted mode of the sample X is defined as the value in the sample with the highest sum of weights.
- When all the weights are equal to one (*i.e.* $w_i = 1, \forall\, i \in \{1, 2, \ldots, n\}$), the weighted mode is equivalent to the mode.

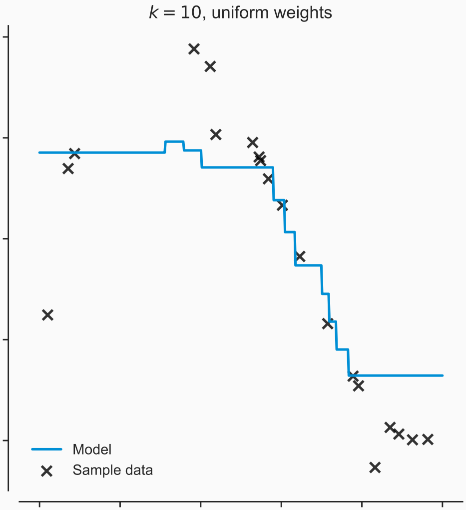
- When applying *k* nearest neighbours, we must make a number of choices:
    1. The number of neighbours to take into account, *k*.
    2. How similarity between pairs of records is determined.
    3. How the similarity measures are converted to weights.
- A grid search over different parameter choices can be used to select the best set, in cases where the correct choice is ambiguous.



- Choosing a smaller value of k makes the model more sensitive to data that is nearby:
  - Valuable neighbours can make a larger contribution.
  - However, neighbours with extreme values have a larger effect on predictions.
- In general, smaller values of k tend to create models with higher variance error.



- Choosing a larger value of k makes the algorithm more sensitive to data that is further away:
    - Neighbours with extreme values have a smaller effect on predictions.
    - Need to use more neighbours to make a prediction though — can dilute the effect of valuable nearby neighbours.
- In general, larger values of k tend to create models with higher bias error.

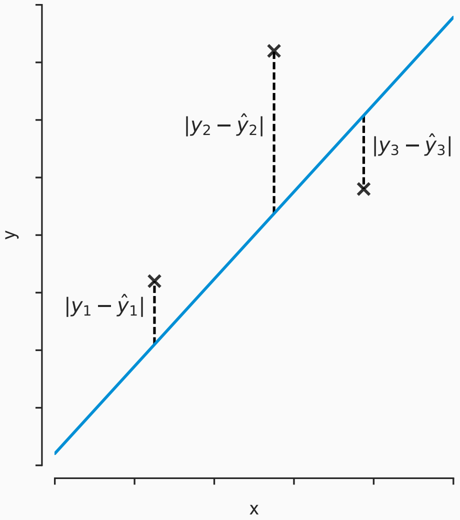


- A *distance measure* is a function that computes how far away one sample is from another in space.
- Distance can be used as a proxy for similarity:
  - The more similar two samples are, the smaller the distance between them.
  - The more dissimilar two samples are, the larger the distance between them.
- Distance can be measured in many ways — typically, it depends on the type of the data you are working with, *e.g*.
  - Continuous-valued: Manhattan distance, Euclidean distance, *etc*.
  - Categorical: cosine distance, Jaccard distance, *etc*.
- Correlation can also be used as a continuous distance measure, but must be scaled so that it is non-negative.

- One way to measure the difference between data samples using their Manhattan distance, $M(X, Y)$, *i.e.*

$$M(X,Y) = \sum_{i=1}^{n} |x_i - y_i| \,. \tag{8.2}$$

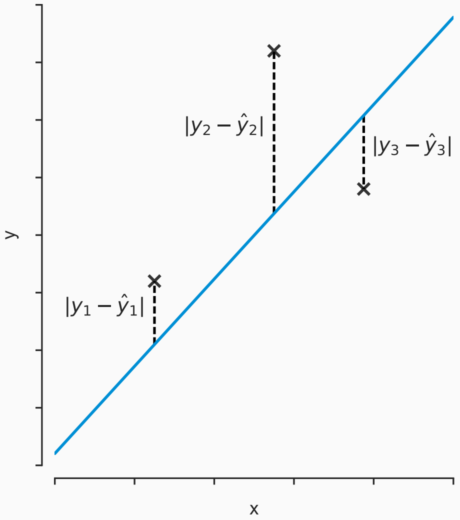
- Manhattan distance is very similar to the idea of mean absolute error (MAE), as illustrated in the chart to the right.



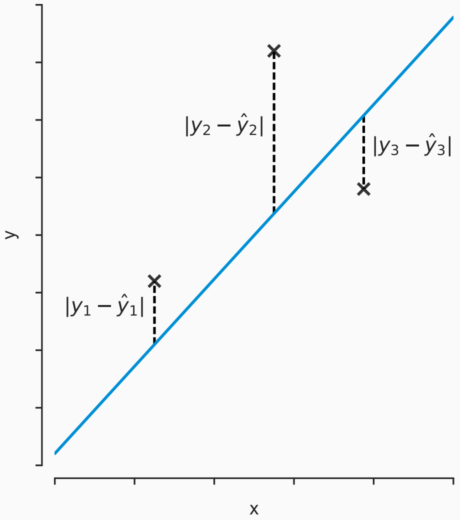
- We can also measure the difference between data samples using their Euclidean distance, $E(X, Y)$, *i.e.*

$$E(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \qquad (8.3)$$

- Euclidean distance is very similar to the idea of root mean square error (RMSE).

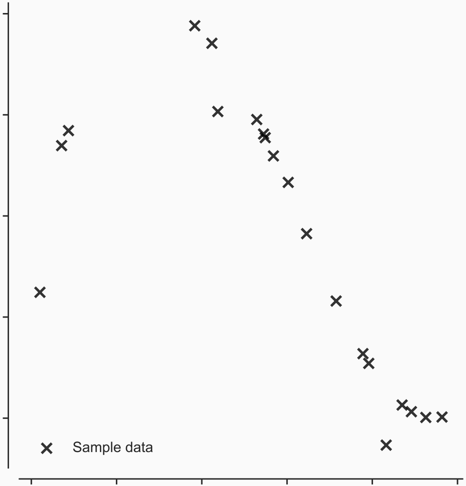


- Manhattan distance relies on the absolute value of the differences between samples, whereas Euclidean relies on the squares of the differences.
- As a result, Euclidean distance emphasises large differences more than Manhattan distance.

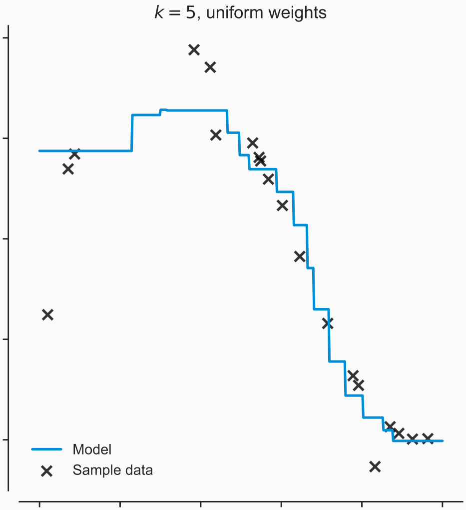


- Distance measurements like Manhattan distance and Euclidean distance are sensitive to scale.
- If the magnitudes of the sample values are very different, then their distance tends to be very large, even though the samples themselves may behave similarly, *e.g.*
    - X is measured in kilobytes while Y is measured in gigabytes.
    - X is measured in metres while Y is measured in kilometres.
- One way to compensate for this is to standardise the samples, so that each sample has zero mean and unit standard deviation.
- Alternatively, we could use correlation — though this also requires scaling (so that it is non-negative).

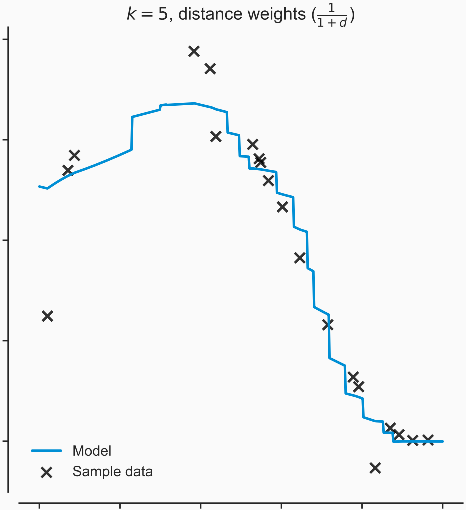
- The choice of weighting scheme affects how the most similar samples are weighted in the prediction.
- Common schemes include:
    1. Uniform weighting: samples are all weighted equally, regardless of distance.
    2. Distance-based weighting: samples are weighted according to their distance.



- In uniform weighting, all samples count equally when computing the prediction.
- All weights are equal, *i.e.* $w_i = \frac{1}{n}$.
- Effectively, this reduces the weighted average/mode to just a simple average/mode calculation.



- Distance-based weighting places more emphasis on samples that are closer and less on samples than are far away.
- Common variants include:
    - Inverse weighting: $w_i = \frac{1}{1+d_i}$.
    - Exponential weighting: $w_i = e^{-d_i}$.



Advantages	Disadvantages
• Simple and intuitive algorithm. • Doesn't make assumptions (*e.g.* linearity) about the structure of the data. • Can be very effective when there is lots of data.	• Prediction quality can be low when k is small (tends to overfit). • Resource usage can be high when the data set is large, as all the data must be searched through when making new predictions. • k nearest neighbours models cannot be represented using an equation or diagram.

Summary

- k nearest neighbours:
    - Used for regression and classification.
    - Three hyperparameters: the number of neighbours, the distance measure, the weighting scheme.
    - Useful when we can't make assumptions about linearity.
    - Cost of prediction becomes high at scale, can be unstable if k is small.
    - Can be used to build recommender systems.
- Lab work:
    - Build a k nearest neighbours classification model for SMS spam data.
    - Build a k nearest neighbours regression model for server load data.
    - Build a user-based recommender system for films.
- Next week: unsupervised learning with clustering algorithms!

1. Hastie et al. *The Elements of Statistical Learning: Data mining, Inference and Prediction.* 2nd edition, February 2009. (stanford.io/2i1T6fN)
2. Ullman et al. *Mining of Massive Data Sets.* Cambridge University Press, 2014. (stanford.io/1qtgAYh)
3. Segaran, Toby. *Programming Collective Intelligence: Building Smart Web 2.0 Applications.* O'Reilly, 2007. (oreil.ly/1nzWODy)
4. Masnick, Mike. *Why Netflix Never Implemented The Algorithm That Won The Netflix $1 Million Challenge.* Techdirt, 13th April 2012. (bit.ly/1BdyZbW)