COMP20008 – ELEMENTS OF DATA PROCESSING

Complete Subject Notes

Semester 1 2016

Contents

Table of Contents

Data Format and Storage

Relational Databases

- Classic method of storing structured data
- Data is stored in rows
    - Each row describes an attribute of the data
- Data can be stored across multiple tables – as many as needed
    - Tables can be linked together via columns that share the same data being in each of the tables
- Database can be queried using languages like **SQL**
- Once data is in a database, it becomes easier to wrangle
    - May have been hard to load it there in the first place

SQL

- Language for manipulated relations in relational databases
- Example: creating a table in SQL

```
create table branch (
    branch_name char(15) not null,
    branch_city char(30),
    assets integer,
    primary key (branch_name)
    )
```

NoSQL

- **Not only SQL** database
- They can be elastic, distrusted over many servers
- Can store many different data types
    - Documents, graphs, sequences, objects
- Does sacrifice some of the properties of relational databases
    - i.e. Query language (not as nice), consistency, fewer integrity guarantees
- Examples: Google BigTable, Amazon SimpleDB. Apache CouchDB

Spreadsheets

- Huge amounts of data are stored in spreadsheets, widely used and very popular
- Used in:
    - Business
    - Medical/Hospitals
    - Sciences
- Examples of spreadsheet software:
    - Microsoft Excel
    - OpenOffice Calc
    - Google Docs Sheets
- **Comma separated values (CSV)**
    - Another spreadsheet format
    - Simpler
    - Human readable text file
    - Lack formatting options of software based spreadsheets

- Python libraries can be used to parse spreadsheets
    - csv
    - xlrd, openpyxl

Text Data

Regular expressions

- Used to specify patterns in text
    - These patterns can be used for computing statistics, checking integrity, filtering, substitutions
- A number of symbols are used to match particular patterns or character within text:

```
- '.' matches any character
- '^' matches start of string
- '$' matches end of string
- '*' zero or more repetitions
- '+' one or more repetitions
- '|' the "or" operator
- '[]' a set of characters, e.g. [abcd] or [a-zA-Z]
```

- To match:
    - Every occurrence of letter 'o': use regular expression **`'o'`**
    - Every occurrence of 'of': use regular expression **`'of'`**
    - Every occurrence of string 'll': use regular expression **`'ll'`**
    - Every occurrence where 'e' appears one or more times: use regular expression **`'e+'`**
    - Every occurrence where 'ee' one or more times: use regular expression **`'(ee)+'`**
    - Every occurrence of 'e' or 'a': use regular expression **'(e|a)'**
    - Match an email address: use regular expression **`[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+`**

Table Data

- Table data is very abundant across the web
    - Encoded in HTML
- Can use Google table search to find tables
- People can create tables using Google fusion tables

HTML & XML Data

- Made up of elements, delineated by start and end tags
    - Not all elements need start and end tags
- Elements correspond to logical units
- Data is stored between tags
- Some elements can have **attributes**
- **HTML – hypertext markup language**
    - Browser uses HTML to determine how to display information
    - Specific set of tags which correspond to specific styling of the data
        - i.e. `<ul>, <ol>, <li>` tags for lists
        - i.e. `<h1>, <h2>, <h3>` tags for headings
        - i.e. `<p>` tags for paragraphs
- **XML – eXtensible markup language**

- Allows user defined elements
- Applications can process or generate XML
- Enables data exchange between platforms
- Facilitates better encoding of semantics

```
<?xml version="1.0"?>
<customers>
    <customer isAlive="True" >
        <name>John Citizen</name>
        <age>23</age>
        <height units="cm">163.1</height>
        <address>123 First Street Sydney, NSW 2000</address>
        <phone>(02) 9250 7111</phone>
        <email>johnc@gmail.com</email>
    </customer>
</customers>
```

- XML has **namespaces:**
  - Used to qualify names with **URIs (universal resource identifiers)**
    - URIs uniquely identify a resource on the web
    - URIs don't have to refer to a real web resource, they just have to be unique
    - Namespaces can apply to elements and/or attributes
    - Consist of `namespace:local-name`
  - Namespaces are declared, and used by appending an associated prefix to the beginning of a tag name, followed by a colon

```
<... xmlns:tabular-info="http://www.tabularinfo.com">
<tabular-info:table>
<tr>
<td>Dogs</td> <td>Cats</td>
</tr>
</tabular-info:table>
```

  - Namespace scope:
    - The element that contains the namespace declaration
    - All its (nested) descendants
    - The declaration can be overwritten by further nested namespace declarations
  - Elements/attributes without a namespace prefix are assigns a default namespace
- XML has a **schema:**
  - Used to ensure data integrity
  - Defines the expected content of an XML document

JSON Data

- Simpler more compact method of storing data compared to XML

```
{
  "firstName": "John",
  "lastName": "Citizen",
  "isAlive": true,
  "age": 23,
  "height_cm": 163.1,
  "address": {
    "streetAddress": "123 First Street",
    "city": "Sydney",
    "state": "NSW",
```

```
    "postalCode": "2000"
  },
```

Other Data Types

Sequences

- Long strings of data, may be repeated
    - i.e. Biological data – protein sequences, DNA sequences

Graphs (Linked Data)

- Used to connect data together, forming links
    - Part of the semantic web
    - Important for the Internet of Things (IoT)
- When one "thing" is looked up, information about that "thing" will be returned as well as relevant information about other related "things"
    - This process use graph data
    - i.e. Friends on Facebook, Related Videos on YouTube
- Examples:
    - Google Knowledge Graph (Google use to improve search)
    - DBPedia
    - Freebase
        - Large database that connects entities together as a graph
            - Facts
            - People
            - Organisations
            - Places
        - Basis of Google Knowledge graph
    - FOAP (friend of a friend)
- Standards of linked data:
    - JSON-LD (JSON Linked Data)
        - Provides mechanisms for specifying unambiguous meaning JSON data
        - Provides extra key using "@" sign – specifies URIs to give meaning to data
            - "@context"
            - "@type"
            - "@id"

```
{"@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "homepage": {
        "@id": "http://xmlns.com/foaf/0.1/workplaceHomepage",
        "@type": "@id"
    },
    "Person": "http://xmlns.com/foaf/0.1/Person"
    },
    "@id": "http://me.example.com",
    "@type": "Person",
    "name": "John Smith",
    "homepage": "http://www.example.com/"
}
```

    - RDF (Resource Description Framework)

- URIs used to give meaning to graph nodes
- RDF can be added to an XML document
- RDF can be added as **triple store:** subject, predicate, object

```
<http://www.w3.org/People/EM/contact#me> <http://www.w3.org/
2000/10/swap/pim/contact#fullName> "Eric Miller" .
<http://www.w3.org/People/EM/contact#me> <http://www.w3.org/
2000/10/swap/pim/contact#mailbox> <mailto:em@w3.org> .
<http://www.w3.org/People/EM/contact#me> <http://www.w3.org/
2000/10/swap/pim/contact#personalTitle> "Dr." .
```

- Graph data can be represented as **Matrix Data**
  - Zeros and ones in a matrix
  - Ones (1) are added if there is an edge between two nodes

Data Pre-Processing and Cleaning

Pre-Processing

- Pre-processing is needed to standardise all different representation of data in a table so they are all the same
    - i.e. Different number representations: written (ten), digits (10), floats (10.0), strings ("10")

Measuring Data Quality

- Accuracy
    - Correct or wrong, accurate or not
- Completeness
    - Not recorded, unavailable
- Consistency
    - Discrepancies in representations
- Timeliness
    - Updated in a timely way
- Believability
    - Can the data be trusted to be correct?
- Interpretability
    - How easily can the data be understood

Terminology

- Columns are referred to as **attributes or features**
- Data rows are referred to as **instances or objects**

Data Pre-Processing Activities

Data Integration

- Bringing data from multiple sources together
    - Resolve conflicts
    - Detect duplicates

Data Reduction

- Decrease number of feature and/or instances in a dataset
- This can be done by:
    - Modifying sampling strategies
    - Removing irrelevant features and reducing noise
- Makes data easier to visualise and faster to analyse

Data Cleaning

- Processes for fixing:
    - Incomplete or missing data
    - Noisy data
    - Inconsistent data
    - Intentionally disguised data

Data Cleaning – The Process

Data Cleaning Tools

- Tools that can be used:
  - Google Refine
  - Kettle
  - Talend
- To clean data, these tools employ techniques such as:
  - Data scrubbing
  - Data discrepancy detection
  - Data auditing
  - ELT (Extraction Transform Load) tools

Missing Data

- Lacking feature values
  - i.e. Age = Null
  - i.e. Name = “”
- Types of missing data:
  - Missing at random: data is missing independently of observed and unobserved data
  - Missing not completely at random: Data missing because people don’t fell comfortable answering the question
- Reasons for missing data:
  - Equipment malfunction
  - Not recorded due to misunderstanding
  - May not be considered important at time of entry
  - Deliberate
- Often difficult to determine why data is missing

Noisy Data

- Types of noisy data:
  - Truncated fields
    - i.e. Extends 80-character limit
  - Text incorrectly split across cells
  - Invalid values
    - i.e. Age = “-5”
- Some causes of noisy data:
  - Imprecise instruments
  - Data entry issues
  - Data transmission issues

Inconsistent Data

- Types of inconsistent data:
  - Different naming representations
    - i.e. “Three” or “3”
  - Different data formats
  - Different birthday representations
    - i.e. Age = 20 or Birthday = 1/1/1996
  - Outliers

Disguised Data

- Types of disguised data:
    - Deliberate misinformation supplied
        - i.e. Everyone's birthday is January 1st?
        - i.e. Made up email address, xx@xx.com
- How to handle:
    - Look for unusual or suspicious values in the dataset, using knowledge about the context of the data

Dealing with Missing Data

- Consequences of not cleaning data:
    - Make break programs that are not expecting missing data
    - Less power for later analysis
    - Could bias later data analysis
- Strategies to deal with missing data:
    - **Case deletion:** Delete all data that has missing data
        - Pros: gives clean new dataset with nothing missing
        - Cons: new dataset may be a lot smaller, more biased
    - **Manually correct:** A human fills in the missing values using their expert knowledge
        - Pros: good for small datasets only
        - Cons: time-consuming
    - **Imputation:** Impute a value (using an algorithm) to replace the missing value with
        - Algorithm 1: Fill all missing values with zero
            - Pros: simple, easy to do, won't break application programs
            - Cons: limited utility for application programs, inaccurate
        - Algorithm 2: Fill all missing values with the mean (or median or mode) value
            - Pros: can be good for supervised classification, not changing data too much
            - Cons: for extremes, the mean will not be accurate, reduces variance and distribution of feature
        - Algorithm 3: Fill all missing values with the mean of only certain values of the same category/cluster, i.e. mean value of only females to find the missing age for a female
            - Pros: more information used, could mean the missing value imputed is more accurate
            - Cons: for extremes, the mean will not be accurate regardless of category, may still reduce variance and distribution of feature, can be difficult to choose clusters of features to compute the mean
        - Algorithm 4: Fill all missing values with the previous value that was entered
            - Used for missing data over time (Day 1, Day 2, etc.)

Recommender Systems and Missing Values

What Are Recommender Systems?

- Users rate certain items on a scale
    - (i.e. movies out of 5)
- Generally, users only rate a few items
    - This leaves missing values
- Major companies want to predict the missing ratings to better recommend items in the future

Examples of Recommender Systems

- Netflix: movies/TV shows to watch
- Amazon: books to read
- IMDb: related movies
- Online Dating: matches people together
- Twitter: who to follow
- Spotify: which song to listen to
- YouTube: which video to watch next
- Facebook: who to add as a friend, read this news
- Tourist attraction apps: tourist attraction
- Steam: related games

How It Works

- User has a profile
- Users rate items
    - Explicitly: by giving a score
    - Implicitly: using web page data
        - Time spent in viewing the item,
        - Navigation path to the item

Collaborative Filtering

- "Making predictions about a user's missing data according to the behaviour of many other users"
    - Looks at the user's **collective** behaviour
    - Looks at the active user **history**
    - Combines this data to form a prediction for missing values

User-User Similarity

Intuition

- How to measure similarity?
- How to select neighbours?
- How to combine the data?

Process

- How to measure similarity?
  - **Method 1:**
    - Compute mean value for User1's missing values
    - Compute mean value for User2's missing values
    - Compute squared Euclidean distance between resulting vectors
  - **Method 2:**
    - Compute squared Euclidean distance between resulting vectors, summing only pairs without missing values
    - Scale the result using total number of value pairs divided by total number of value pairs minus number of pairs with missing values: $\frac{n_{no.of\ total\ pairs}}{n_{no.\ total\ pairs} - n_{no.\ of\ missing\ pairs}}$
  - **Other Methods:**
    - Correlation
    - Cosine similarity: angle between user profile vectors
- How to select neighbours?
  - Need to select users to compare to
  - At runtime choose top *k* most similar users
  - Combining: prediction of rating is the weighted average of the value from the top *k* similar users
  - *k* parameter is determined based on the type of data and the context of the data
  - Pre-computing can be done to speed this process up
    - This is done by finding the nearest cluster and using the centre of the cluster as the rating prediction
    - Faster, but less accurate

Summary

- Achieve good quality predictions in practice
- User based method are a bit slow, as user bases are constantly changing, similarity needs to be re-computed
- Issues:
  - User preference is dynamic
    - This means there will be a high frequency of offline computation need to calculate information
  - No recommendation for new users
    - Not enough is known about new users

Item-Based Methods

Intuition

- Searches for similarity among items
- All computations can be done offline
- More stable than user-user similarity due to no need for frequent updates

Process

- Same as user-user similarity, but on item vectors
  - Find similar items to the one whose rating is missing
  - i.e. For the item i_i compute its similarity to every other item i_j
- **Offline phase:** for each item
  - Determine its *k* most similar items
  - Can use the same type of similarity as used for user-based

- **Online phase:**
  - Predict rating r_{aj} for a given user-item pair as the weighted sum over *k* most similar items that are rated
  - $r_{aj} = \frac{\sum_{i \in k \; similar \; items} sim(i,j) \times r_{aj}}{\sum_{i \in k \; similar \; items} sim(i,j)}$

Matrix Based

- Treat the user-item rating table as a matrix, *R*
- User matrix factorisation to factorise this rating table

Factorisation

- Can factorise whole numbers
  - 15 = 3 × 5
  - 99 = 3 × 33
- Can also factorise approximately
  - 17 ≈ 6 × 2.8
  - 167 ≈ 17 × 9.8

Matrix Factorisation

- Given matrix *R*, the matrices *U* and *V* can be found such the *U* and *V* can be multiplied to give *R*
  - $R \approx UV$
- Matrices must be all the right dimensions
  - *R* is *m* × *n*
  - *U* is *m* × *k*
  - *V* is *k* × *n*
  - Where *k* is the 'number of latent factors'

How to Factorise a Matrix

- Intuitively, an algorithm searches over lots of choices for *U* and *V,* with the aim of making the error as low as possible
- If there are missing values in *R*, these values will be ignored when computer the error

Factorisation and Missing Values

- Once the algorithm computes the factors of *R,* it will give two matrices *U* and *V*
  - When *U* and *V* are multiplied, a matrix very similar to *R* will be produced*, R'*
  - *R'* will contain the missing values from matrix *R*
- **The matrix *R'* can be used to predict the missing values is it itself will have no missing values and the all values will be very similar to the original *R* matrix**

Commercial Recommender Systems

- Commercial recommender's systems (i.e. Netflix, Amazon) use variation of matrix factorization

Outlier Detection

Outlier Analysis

- **Outlier:** a data object that **deviates significantly** from the normal objects as if it were **generated by a different mechanism**
    - i.e. Unusual credit card purchase, unusual sports statistics
- Statistically,
    - Normal objects are generated using some statistical process
    - Outliers will deviate from this generating process
- Outliers can be different to **noise data**
    - Noise is most commonly a random error or variance measured in a variable
    - Noise should be removed before outlier detection
- Outliers can be interesting
    - Understanding why or how an outlier occurred can help to determine why that data point was generated differently to the rest

Applications

- Credit card fraud detection
- Telecom fraud detection
- Medical analysis: unusual test results
- Sports: identifying exceptional talent

Why is Outlier Detection Important?

- Outliers can skew results if the average (mean) needs to be calculated for the data set

Types of Outliers

Global Outlier

- Also called *point anomaly*
- An object is a global outlier (O_g) if it significantly deviates from the rest of the data set
- Issues:
    - There needs to be an appropriate measurement of deviation

Contextual Outlier

- Also called *conditional outliers*
- An object is a conditional outlier (O_c) if it significantly deviates based on a selected context
- Example:
    - Is the weather 5°C in Melbourne?
        - Depends on if the season is Winter or Summer
- Attributes of data should be divided into two groups
    - **Contextual attributes:** defines context, i.e. time, location
    - **Behavioural attributes:** characteristics of the object, used in outlier evaluation
- Issues:
    - How to define or formulate meaningful context?

Collective Outliers

- A subset of data objects that **collectively** deviate significantly from the whole dataset, even if the individual data objects may not be outliers
- Applications, i.e. intrusion detection
    - When a number of computers keep sending denial of service packages to each other
- Detection:
    - Consider not only behaviour if individual objects, but also that of groups of objects

Challenges of Outlier Detection

- Issues with types
    - A dataset may have multiple types of outliers
    - One object could be characterised as more than one type of outlier
- Modelling normal objects and outlier properly
    - It is hard to enumerate all possible behaviours in an application
    - The border between normal and outlier object is a grey area, it is hard to determine
- Application specific outlier detection
    - Choice of distance measure among objects and the model of relationship among objects are often application dependent
- Handling noise
    - Noise may distort the normal objects and blur the distinction between normal objects and outliers
    - This can hide outlier and reduce effectiveness of outlier detection
- Understandability
    - Understanding why there are outliers to justify the detection
    - Specifying the degree of an outlier

How to Detect Outliers

- For 1D data:
    - Boxplots
    - Statistical tests
- For 2D Data:
    - Scatter plots
    - Eyeballing
- For 3D data:
    - 3D Scatterplots
    - Eyeballing
- For data greater than 3D
    - Statistical methods
    - Algorithmic methods

Box and Whisker Plots

- From the sample compute:
    - Minimum and maximum values
    - Median
    - First and third quartile (Q1 and Q3)
    - Interquartile range, (Q3 – Q1)
- Draw:
    - Vertical number line

    - Plot min and max as whiskers on the number line
    - Draw a box that begins at Q1 and finishes at Q3 on the number line. Attach the whiskers
    - Plot the median as a horizontal line across the appropriate spot in the box

Outliers and Tukey Boxplots

- Whiskers range becomes:
    - Lowest point still within 1.5IQR of lower quartile
    - Highest point still within 1.5IQR of upper quartile
- Outliers determined are values: (plotted as shaded dots):
    - >3IQR above the third quartile
    - >3IQR below the first quartile
- Suspected outliers are values: (plotted as open dots):
    - >1.5IQR above the third quartile
    - >1.5IQR below the first quartile

Statistical Methods

- Assume that normal data follows some statistical model
    - That data not following that model will be outliers
- Effectiveness:
    - Depends on whether the assumption of statistical model holds in the real data

Univariate Outlier Detection

- Detect one outlier at a time and repeat
- Compute:
    - $\frac{max_{i=1,\dots,N}|x_i-\mu|}{\sigma}$
    - Where:
        - x_i is a data instance
        - μ is the sample mean
        - σ is the sample standard deviation
- Then the assumption can be made that the population is normally distributed and a statistical hypothesis test can be done
- The farthest point is an outlier if it is unlikely to have occurred under the normal distribution assumption
- Throw away this outlier if test indicates that the instance is 'too far' from the mean

Histograms

- Draw a histogram by first **binning** data
    - Data that is in a bin that is very small, or doesn't even exist may be an outlier
- Issue: hard to choose appropriate bin size for data
    - Too big bin, normal object will be in empty/rare bin, false positive
    - Too small bin size, outliers will be in some frequent bins, false negative

Outlier Detection for Data in Higher Dimensions

- Datasets with more than 4 dimensions
    - Hard to visualise
    - Hard to estimate parameter of the data using statistical methods
    - Data may not follow a normal distribution

Distance Functions

- Measure of distance can be used to detect outliers for data over 4 dimensions

- Most well known as used is **Euclidean distance**
  - $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 \dots (x_n - y_n)^2}$
- It is common to normalise each attribute into range [0,1] via a pre-processing step before computing distances

Basic Method: Centre

- Computer distance of object from the centre of the data
  - Must determine a "centre"
  - Assumes there is only one centre
- The further away an object is from the centre, the more likely it is to be an outlier

Cluster-Based Outlier Detection

- Outliers will be the furthest away points from any **groups of normal objects**
  - Called **clusters**
- Considers that fact that there can be multiple localities and multiple "centres" for that data
  - Outlier score is relative to a data objects locality
- Each instance is associated with exactly on cluster and its outlier score is equal to the distance from its cluster centre
- Challenges:
  - How to determine the number of clusters that give a meaningful answer
  - How to generate a cluster: algorithmic
- Good Clustering:
  - Objects are grouped into clusters, and that cluster is very tight, compact: **intra-cluster distance is minimised**
  - Cluster are far away from each other: **inter-cluster distances are maximised**

K-Means

- Given a parameter *k*:
  - Select *k* seed points as initial cluster centres at random
  - Assign each object to the nearest seed point
  - Compute new seed points as the centroids of the clusters of the current partitioning
  - Continue this process until assignment does not change
- Different runs of the algorithm will produce different results
  - Run many times and choose result that is the best
- Closeness is measured by Euclidean distance
- Algorithm can be shown to converge to a local optimum
  - Typically it does require many iterations to achieve this

Hierarchical Clustering

- Produces a set of nested clusters organised as a hierarchical tree
- Can be visualised as a **dendrogram**
  - A tree-like diagram that records the sequences of merges or splits
- Two main types **agglomerative** and **divisive**
- **Agglomerative:** more popular technique
  - Basic algorithm:
    - Compute a proximity matrix
    - Let each point be a cluster
    - Repeat
      - Merge two closest clusters

            - Update proximity matrix
        - Terminate when only a single cluster remains
    - Key operation: computation of the proximity of two clusters
        - Can be done a number of ways
    - Defining Inter-Cluster Similarity
        - **MIN** (single linkage): similarity of two clusters is based on the two most similar (closest) points in the different clusters
            - Can handle non-elliptical shapes
        - **MAX** (complete linkage): similarity of two clusters is based on the two least similar (most distant) points in the different clusters
            - Less susceptible to noise and outliers
            - Tends to break large clusters
            - Biased towards globular clusters
        - **Group Average** (average linkage): proximity of two clusters is the average of pairwise proximity between points in the two clusters
            - Less susceptible to noise and outliers
            - Biased towards globular clusters
        - Distance between centroids
- Strengths:
    - Does not have to assume any particular number of clusters
        - Any number of cluster can be obtained by "cutting the dendrogram at the appropriate level
    - The clusters may correspond to meaningful taxonomies
        - i.e. biological sciences
- Limitations:
    - Once a decision has been made to combine two clusters, it cannot be undone
    - No objective function is directly minimised
    - Depending on the methods used, each has problems:
        - Sensitivity to noise and outliers
        - Difficulty handling different sized clusters and shapes
        - Breaking large clusters

Limitations of Distance Based Clustering

- Strengths:
    - Works for many types of data
    - Clusters can be regarded as summaries of the data
    - Once the clusters are obtained, only need to compare any object against the cluster to determine whether it is an outlier – fast
- Weaknesses:
    - Many clustering methods, therefore many clustering-based outlier detection methods
    - Clustering is computationally expensive: clustering methods for outlier detect can be costly and does not scale up well for large datasets

Proximity Based Outlier Detection

Distance-Based Outlier

- An object is an outlier if its *D*-neighbourhood (object less than distance *D*) contains only a few (less than 1 – *p* percent) objects
- Need to choose:
    - A distance radius, *D*

    - Occupancy measure; percentage parameter, *p* (higher *p* enforces fewer objects inside the circle)
- Choosing these parameters is hard:
    - Could use domain knowledge
    - Doesn't provide ranking for outliers

K-NN

- Involves only 1 parameter, *k*, to compute the *k*th nearest neighbour of an object
- Outlier score of an object is the distance to its *k*th nearest neighbour
    - Sorts the object in order of score (highest to lowest)
    - Select the *n* objects with the highest outlier score
- Algorithm:
    - Nested loop
        - For each object: compute *kth* nearest neighbour with sequential scan

Visualisation and Clustering

Motivation for Visualisation

- Converting data in visual format
    - Reveals characteristics of the data
    - Shows relationships between objects and/or features
    - Simplifies the data
- Humans are good at analysing information in a visual formatted
    - Spotting trends, patterns, outliers

Basic Visualisations

- Boxplots
    - Shows median, quartiles and outliers
- Bubble plots
    - Size of circle at a point indicates its intensity

Scatter plots

- Plotting points in 2D or 3D space
- Take two columns from a dataset and plot them
- Can show some clustering
- Could be losing some information due to 2D nature

Histogram

- Shows the distribution of values of a single variable
- Divides the values into bins and show a bar plot of the number of objects in each bin
- Height of bar indicates the number of objects
- Shape of histogram depends on the number of bins

Advanced Visualisations

Heatmap

- Plot the data matrix
- Use different colours to highlight different intensities of the data
- Useful when objects are sorted according to class
- Features are typically normalised to prevent one attribute from dominating the plot
- Heatmaps can be used for visual assessment of clustering tendency

Visual Assessment of Clustering Tendency

- Difficult to determine how many clusters and how bog clusters are for high dimensional data sets
- Cluster structure can be visually determined by inspecting a heatmap
    - Represents datasets in an n*n image format
    - Applicable for many different types of data
- **Dissimilarity Matrix**
    - Compute all pairwise distances between objects
        - i.e. the distance between object 1 and object 1, the distance between object 1 and 2, etc.
    - The matrix will be symmetric about its leading diagonal

        - The value 0 will run down the diagonal length
    - Can visualise the dissimilarity matrix using different colours for different values
    - The bigger the dissimilarity matrix is, the more confusing it becomes to determine any information from
    - Dissimilarity matrix can be ordered/arranged in a better way
        - Nearby objects in the ordering are similar to each other
        - This produced large, dark blocks
        - Each block will represent a cluster
    - VAT Algorithm for ordering
        - Choose pair of objects that are furthest apart
        - Choose next object that is closest to the first
        - Repeat by selecting next closest object
    - VAT algorithm won't be effective in all situations
        - For complex shaped datasets, quality of a VAT image may slightly degrade

Parallel Coordinates

- Used to plot feature values of high-dimensional data
- Instead of perpendicular axes, use a set of parallel axes
- The feature values of each object are plotted as a point on each corresponding coordinate axis and the points are connected by a line
    - Each object is represented by a line
- Often, lines representing a distinct class of object group together
- Ordering of attributes is important in seeing such groupings
- Key Issues
    - Scaling axes
        - Affects the visualisation
        - Could choose to scale all features into the range [0,1] via pre-processing
    - Ordering of axes
        - Influences the relationships that can be seen
        - Correlations between pairs of features may only be visible in certain orderings

Dimension Reduction

- **Curse of dimensionality:** data analysis techniques that work well at lower dimensions often performs poorly as the dimensionality of the data increases
- As dimensionality increases, data becomes increasingly sparse and all the distances between the pairs of points begin to look the same
    - This effects any algorithm based on distance
- Purpose of dimension reduction
    - To avoid curse of dimensionality
    - Reduce amount of time and memory required by processing algorithms
    - Allow data to be more easily visualised
    - Helps to eliminate irrelevant features or reduce noise
- Method
    - Input: dataset with N features and K objects
    - Output: a transformed dataset with n<<N and K objects
        - n is often set to 2 or 3 so that the transformed data can be visualised easily
- Transforming
    - Must preserve characteristics of the data
        - Preserve distances between the points

- If a pair of objects is close/far before the transformation, they should be close/far after transformation
- In general, to reduce dimensionality, use a subset of the original features
- However, when transforming N to n<<N
  - The output features do not need to be a subset of the input N features
  - They can be new features created by applying some function to the input N features

- **Principle Component Analysis**
  - Find a new set of features that better captures the variability of the data
  - First dimension to capture as much of the variability as possible
  - The second dimension is orthogonal to the first, captures as much of the remaining veracity as possible
  - The third dimension is orthogonal to the first, captures as much of the remaining veracity as possible

NoSQL Database Management Systems

Distributed DBMSes

Rise of "Big Data"

- The Internet has created massive amounts of data that can be stored in databases
  - Computer logs
  - Commercial transactions
  - Social media posts
  - Many other data sources across the web
- DBMSes that manage this data generally uses **NoSQL**
  - Does not follow the traditional relational model of databases
- NoSQL DBMSes include:
  - Amazon DynamoDB
  - Google Spanner
  - Apache Cassandra

What is "Big Data"

- **Volume**
  - The amount of data (giga, tera, peta, exa)
- **Velocity**
  - Frequency of new data being added to the system and analysis being performed
- **Variety**
  - Variability and complexity of the data schema
  - More complex the schema, the high the probability of the scheme changing along the way, adding more complexity
- **Veracity**
  - Level of trust in the data accuracy
  - The more diverse and unstructured the sources are the less veracity

Tackling "Big Data"

- Computing load is split among different computers – **distributed databases**
- Set of computers that cooperate to manage the database is called a **cluster**
  - Each computer in the cluster is called a **node**
- **Horizontal scaling**: making a computing system more powerful, by adding more computers
- **Vertical scaling**: making a computer system more powerful by adding more resources to a node
- DBMSes have to propagate changes across the nodes while keeping consistency of the whole database

Database Availability and Consistency

- Two major database qualities are availability and consistency
  - **Availability:** the database is always able to answer queries from clients
  - **Consistency:** the database gives the same response to queries happening at the same time
- Availability and consistency do not scale
  - This is because when the volume of data calls for a cluster of databases servers, ensuring that every change is propagated at every node is a slow process and can be blocked by a partition of the cluster
- **Two-Phase Commit**
  - Usual algorithm used in relational DBMSes to enforce consistency during transactions

        - **Transactions** are a set of changes in a database that are treated as one single change
    - Algorithm logic:
        - Locking data that are within the transaction scope
        - Performing transactions on a temporary database
        - Completing transactions **(commit)** only when all nodes in the cluster have performed the transaction
        - Aborts transaction **(rollback)** when a partition is deleted
    - This algorithm means that there is:
        - **Reduced availability:** due to the data lock, stop in case of partition
        - **Enforced consistency:** every database is in a consistent state, and all are left in the same state

Multi-Version Concurrency Control (MVCC)

- MVCC is a method to ensure availability and some sort of recovery for a partition
    - This is done using **revisions,** i.e. data is not replaced, just given a new revision number
- Concurrent updates are possible without distributed locks
    - This is because updates will have different revisions numbers
    - The transaction that completes last will have a higher revision number, hence it will be considered the "current" value
- Relies on increasing revisions numbers and the preservation of old objects to avoid read locks and ensure availability

Sharding

- **Sharding:** the partitioning of a database "horizontally"
    - Rows are partitioned into subsets that are stored on different servers
- Advantages
    - Improves the performance through the distribution of computing loads across nodes
- Sharding strategies
    - Hash sharding: to distribute rows evenly across the cluster
    - Range sharding: similar rows are stored in the same node

Replication

- **Replication:** the action of storing the same row on different nodes to make a database fault tolerant
- Replication and sharing can be combined with the objective of maximising availability while maintaining a minimum level of data safety

Non-Relational Data Models

Why DBMSes for Distributed Environments?

- Relational DBMS are good for ensuring consistency and availability, there is nothing preventing them implementing partition-tolerate algorithms
- Document-orientated database can be used
    - Consists of only one document type
    - All features/columns of the data are nested as arrays in the same document
- Document-orientated databases need less synchronisation and are easily shardable

Non-Relational Data Model Types

- Key-value stores (Redis, PostgreSQL Hstore)
- BigTable DBMSes (Google BigTable, Apache Accumulo)

- Graph DBMSes (Neo4J, OrientDB)
- Document-oriented DBMSes (CouchDB, MongoDB)

MapReduce Algorithms

- A framework used to process huge amounts of data in parallel on large clusters of hardware
- Has two steps, map and reduce
- Algorithm logic
    - Map: distributes data across machines
    - Reduce: hierarchical summation of data until the result is obtained
- Advantages
    - Parallelism
    - Move the process to where the data is, greatly reducing network traffic
    - Horizontally scalable

CouchDB

Basics

Installation

- Installs easily on any system
- Version 2.0 adds clustering and new user interface called Fauxton
- Resource-conscious:
    - uses less than 100MB of memory
    - uses less than 10MB of disk space

Main Features

- Document-oriented DBMS
    - Documents expressed in JSON
- HTTP ReST API
- Web-based admin interface
- Web ready
    - Talks in HTTP
    - Produces JSON (also HTML or XML)
- Supports MapReduce algorithms
- JavaScript is used as the default manipulation language
- Schema-less data model
- Supports replication and sharding

Fauxton User Interface

- GUI (graphical user interface) for CouchDB
- Accessed by going to http://localhost:5984/_utils in a browser
- Can do most operations easier that using command line and cURL
    - Operations include:
        - Edit documents
        - Design documents
        - Test MapReduce
        - Modify configuration of the database
        - Manage users
        - Set up replications

Databases

- A CouchDB instance can have many databases
    - Each database can have its own set of functions (**design documents**)
- Every CouchDB instance has system databases (indicated by _ as the first character)
    - i.e. _replicator and _users

HTTP Methods

- For communication to CouchDB, HTTP requests are used
    - GET: download existing content from a server
    - POST: create new content on the server

- PUT: update/create content on the server when its ID is unknown
- DELETE: removal of content on the server

Manipulating Databases

- Databases can be manipulated using cURL in the command line
  - Adding and deleting a database

```
curl -X PUT "localhost:5984/samll?n=2&q=3"
```

    - *n* is the number of replicas
    - *q* is the number or shards

```
curl -X DELETE "localhost:5984/small"
```

  - Listing all databases

```
Curl -X GET "localhost:5984/_all_dbs"
```

    Responses are JSON objects:

```
["_replicator", "_users", "_exampledb"]
```

  - Inserting a document

```
curl -X POST "localhost:5984/exampledb" --header
"ContentType:application/json" --data '{"type": "account", "holder":
"Alice", "initialbalance": 1000}'
```

    Response:

```
{"ok":true,"id":"c43bcff2cdbb577d8ab2933cdc0011f8","rev":"1-
b8a039a8143b474b3601b389081a9eec"}
```

    - Each document added is given a unique ID, which can be used to retrieve the document later
    - Each document is also given a unique revision number, indicating which version of the document is the newest
  - Retrieving a document

```
curl -X GET "localhost:5984/exampledb/
c43bcff2cdbb577d8ab2933cdc0011f8"
```

    - To retrieve a document, its ID is required

System Properties of Documents

- **_id:** the ID of a single document
  - By default, an ID is generated by CouchDB and is guaranteed to be unique
  - ID can be set by the user during the document load
- **_rev:** revision number of document
  - Guaranteed to be increasing per-document
  - Every database instances will pick up the same revision as the “live” version of the document
  -
- To request to set a custom ID for a document, PUT must be used

```
curl -X PUT "localhost:5984/exampledb/charlie" --header
"ContentType:application/json"
--data '{"type": "account", "holder": "Charlie",
"initialbalance": 100}'
```

- The ID will be “charlie”

Response:

```
{"ok":true,"id":"charlie","rev":"1-faff5448bf3051ac4fb8f1cc2b04bc51"}
```

- Generating **UUIDs** (universally-unique Identifier)
  - Guaranteed to be unique
  - To get a UUID:

```
curl -X GET "http://localhost:5984/_uuids" ::Gives 1 UUID
curl -X GET "http://localhost:5984/_uuids?count=3" ::Gives 3 UUIDs
```

- Documents can have attachments
  - One or more of whatever MIME-type is needed

```
curl -X PUT "localhost:5984/exampledb/text/original?
rev=1-26074febbe9a4a0e818f7d5587d7411a" --header "ContentType:image/png"
--data @./scannedtext.png
```

  - Attachments listed in the _attachments attribute of a document

Conflicts

- Conflicts and errors can occur when manipulating documents in a database
  - HTTP status code 409

```
curl -X PUT "localhost:5984/exampledb/charlie" --header
"ContentType:application/json"
--data '{"type": "account", "holder": "Charlie",
"initialbalance": 200}'
```

Response:

```
{"error":"conflict","reason":"Document update conflict."}
```

  - To avoid errors and conflicts, the revision number of the document being updated must be stated

```
curl -X PUT "localhost:5984/exampledb/charlie?rev=1-
faff5448bf3051ac4fb8f1cc2b04bc51" --header "Content-Type:application/
json" --data '{"type": "account", "holder": "Charlie", "initialbalance":
200}'
```

Response:

```
{"ok":true,"id":"charlie","rev":”2c0716f36b7cb2b2d31102fe807697573"}
```

    - A new revision number will be issued in the response
- Conflicts can occur on a cluster of CouchDB nodes
  - If a cluster gets partitioned and two nodes revive different updates of the same document, two different revisions are added
  - Only one of these revisions are returned as the current version
  - This “winning version is computed deterministically
    - Guaranteed to be unique on any node of the cluster
  - CouchDB can return all conflicting revisions in a database

```
GET /exampledb/_all_docs?include_docs=true&conflicts=true
```

Deletion of Documents

- Documents are not deleted fully until they are **purged**
    - Documents may be able to be retrieved with a bit of effort

Bulk Managing Documents

- Documents can be bulk loaded, deleted, and updated by the CouchDB bulk docs API

```
curl -v -X POST "localhost:5984/exampledb/_bulk_docs" --header
"Content-Type:application/json" --data '{"docs":[{"name":"joe"},
{"name":"bob"}]}'
```

Response:

```
[{"ok":true,"id":"c43bcff2cdbb577d8ab2933cdc18f402","rev":"1-
c8cae35f4287628c696193172096c988"},
{"ok":true,"id":"c43bcff2cdbb577d8ab2933cdc18f796","rev":"1-
c9f1576d06e12af51ece1bac75b26bad"}]
```

- The same POST request can be used to update multiple documents
    - Revision numbers and IDs need to be provided

Attachments

- Documents can have attachments
- Attachments can be whatever MIME-type is needed
    - Binary MIME-types are also allowed
- Attachment list in eh _attachments attribute of the document
    - Must include content type, hash code and length

Record Linkage

What is Record Linkage

- Combine related records across sources
- Common problem for organisations like
    - Census Bureau
    - Credit risk assessment
    - E-commerce
    - Health systems
        - Match patient data from a hospital with population data from a city
        - Match hospital data and death register data
        - Match hospital data and job occupation data
    - Security
        - Match data about people flight schedule with information across different databases (previous flights, crime) to determine high-risk passengers
        - Identity matching for bank loan or similar
    - Business
        - Two business collaborate with each for mutual gain
        - Geospatial data: utility providers
    - Bibliographic databases
        - Matching authors with publications from different sources
    - Matching a database against itself
        - i.e. business wants to carry out advertising campaign
    - Facebook: linking likes, interests, places
- Also called **data linkage**

Issues with Data Linkage

- Need an attribute with unique value per record
    - This is a **key**
    - If this key exists, then a *database join* can be used to merge multiple datasets
- Datasets not normally have common unique keys
    - Concatenate multiple records to make a unique key
- Values are noisy
    - Need to clean data
    - Numbers - written and digits, remove diacritics
- They may be missing attributes in some of the datasets

Record Linkage Process

Preparation

- Clean records
- General amortisation

Blocking

- All the records will need to be scored
- Blocking phase helps to reduce the number of strings that need to be compared later

- To block:
  - Choose an attribute to block against (i.e. title)
  - Find some simple values from within that attribute (i.e. a single word)
    - These words are the "blocks"
  - Sort all records into their appropriate blocks if they contain the word that the block represents
  - Only compare records that are in the same block
    - i.e. Only compare movies that share common word/s in their title

Score

- Need to score each record pair for similarity
  - **Cosine Similarity Scoring:**
    - (s_1, s_2): $\frac{|s_1 \cap s_2|}{\sqrt{|s_1||s_2|}}$
    - Treating s_i as a set of words
    - $|s_i|$ means count of number of words in the set
    - Denominator normalises the score based on how long each word is
  - **Edit Distance Similarity**
    - Minimum number of character insertions, deletion and substitutions to go from s_1 to s_2
    - Expensive to compute
- Then need to combine these scores to produce a single score for each record
  - Idea 1: sum up feature scores
  - Idea 2: add similarities, minus dissimilarities
  - Idea 3: weighted sum
  - Idea 4: label examples of non/matching record pairs, train classifier *f* using **machine learning**

Matching

- Need to match sufficiently similar records based on the score each recorded received
- Can be done using a **threshold** value
  - i.e. match when score is > 0.5
- Can be matched using **bipartite graph matching**

Merging

- Combine the matched records
- Resolve any conflicting attributes that arise

Similarity

- **Approximate Similarity**
  - Convert each string to 2-grams. Convert to lowercase and list all substrings of length 2
    - "Advise" -> ["ad","dv","vi", "is", "se"]
    - "Advice" -> ["ad","dv","vi", "ic", "ce"]
  - Similarity is then calculated using: $sim(X, Y) = \frac{2h}{x+y}$ (Dice Coefficient)
    - *h* is the number of common 2-grams
    - *x* is the number of 2-grams in string X
    - *y* is the number of 2-ghrams in string Y
  - 3-grams and 4-grams can be used
    - Not 1-grams as ordering of letters in a string would be ignored
    - Not 10-grams as many words are not long enough

Privacy

- Matching with a single organisation is usually not a concern from a privacy perspective
    - Assume individuals doing the matching are authorised and aware of privacy policies
- If matched data is being passed to another organisation or being made public there will be privacy issues

Hashing

- Hash function maps a data item of an arbitrary size to a data item of a fixed size
- Non-invertible hash function: Given output *H(X)* it is extremely hard to reconstruct X
    - MD5: produces a 32-digit hex number
    - SHA-3-512: produces a 64-digit hex number

Bloom Filters

- An array of *l* bits, all initially set to 0
- Used to store strings by using hash functions H_1 ... H_k to turn on certain bits in the bloom filter
    - Each hash function maps an input string to a value in range [0, *l-1*]
    - To store a string *X* in the bloom filter, set array $H_i(X)$ in the bloom filter to value "1", for each $H_i(X)$
- Example (supposing *k=2* and *l=10*):
    - Bloom filter: [0,0,0,0,0,0,0,0,0,0]
    - Store string: "yes"
        - $H_1(yes) = 2$, $H_2(yes) = 5$
        - Therefore, set bits 2 and 5 to have value "1" in the bloom filter
    - Bloom filter: [0,0,1,0,0,1,0,0,0,0]
    - Store string: "no"
        - $H_1(no) = 7$, $H_2(no) = 9$
        - Therefore, set bits 7 and 9 to have value "1" in the bloom filter
    - Bloom filter: [0,0,1,0,0,1,0,1,0,1]
- If a bit is already set to "1", no change need be made
- Checking if a string is present in the bloom filter:
    - For a string *X*, to check if *X* is in the bloom filter, *X* can be hashed using the *k* has functions
        - If at least one of the bits having the value $H_i(X)$ is set to zero, *X* is **definitely not** in the bloom filter
        - If all of the bits having the value $H_i(X)$ are set to 1, then *X* **appears** to be a member of the bloom filter. It may also be a false positive, where probability of an FP is $(1 - e^{-kn/l})^k$
            - Can't say for certain that *X* is in the bloom filter, as another combination of strings may also turn on the same bits as *X*
- Comparing two strings for similarity
    - Given two strings
    - 2-grams of first string in bloom filter *B1* and the 2-grams of the second string stored in bloom filter *B2*.
        - Both bloom filter must be the same length and use the same hashing functions
    - If the two strings have a lot of 2-grams in common, then the bloom filters will have a large number of identical bits set to 1
    - Similarity can be calculated using: $sim(B1, B2) = \frac{2h}{b1+b2}$
        - *h* is the number of bits set to 1 in both bloom filters
        - *b1* is the number of bits set to 1 in bloom filter *B1*
        - *b2* is the number of bits set to 1 in bloom filter *B2*
    - A threshold value needs to be set for determining if *X* and *Y* are similar

Party Protocol

- **2 party protocol:**
  - Each organisation sharing data will apply a hash function to the attribute used to join the databases
  - Then the hash values can be shared with the other organisations
  - Issues:
    - One character changes (i.e. due to noisy data) will produce a completely different hash, and won't get matched
    - Dictionary attack: the hash value could be inverted using brute force against a pre-computed dictionary of all hash values for English words
- **3rd Party protocol**
  - The organisations that want to share data (A and B) can involve a trusted 3rd party (C)
  - Organisation's A and B send their hashed data to Organisation C
  - Organisation C checks that data for matches
  - Organisation C may be malicious:
    - They could mount a dictionary attack and guess the hashed values
    - A and B must perform "dictionary attack resistant" hashing
- **3rd Party Protocol Using a Salt:**
  - A salt is a secret key known only be Organisation A and B
  - The salt is concatenated to every name field by Organisation A and B before hashing
  - Organisation C never knows the salt
    - Therefore, they cannot perform a dictionary attack to reverse the hash
- **3rd Party Protocol Using Bloom Filters:**
  - Organisation A and B send their records encoding in bloom filters.
  - Organisation A has m records and Organisation B has n records
  - Organisation C will have to do $m \times n$ comparisons of bloom filters
  - Organisation C will send back IDs of any similar recorded pairs it found
  - A and B must agree on bit array length, l and k hash functions
    - A salt can also be used and must be agreed upon
    - A and B must also add dummy records (and their bloom filters)
  - There may be issues using bloom filters to represent short strings
    - A and B can add random 2-grams or random bits to the bloom filter

Frequency Attack

- The 3rd party protocol prevent dictionary attacks, but not frequency attacks
- The 3rd party can compare the distribute of hashed values to a known distribution
  - i.e. distribution of surname frequencies can be compared against the number of times a particular hash value appears in the datasets provide for matching
    - If the frequency of hash values matches the frequency of surnames, the original values can be assumed
- This can be used to guess some of the hashed values, even if a salt is used
- This can prevent by A and B, by adding random records in their data to obscure common frequencies

Correlation

Discovering Correlation

Importance of Correlation

- It helps to discover relationships that are hard to see in raw data files and data tables
- One step towards determining causality: **A causes B**
- Used for:
    - Finding relationships
    - Fining causality
    - Feature ranking: selecting the best features for building better machine learning models

Measures of Correlation

- Euclidean distance
- Pearson coefficient
- Mutual information

What Is Correlation

- Correlation is used to detect pairs of variables that might have some relationship
- Correlation is usually determining by plotting the variable on a scatter plot
    - Linear relation: indicates a strong relationship between the two variables
    - Non-linear relation: variable show less correlation
- Correlation **does not** necessarily imply causality

Measures of Correlation

Euclidean Distance

- When *(x,y)* is a data point:
    - $d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$
    - This calculates the length of the lien connecting *x* and *y*
- Square Euclidean Distance
    - $d^2(x,y) = (x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2$
    - Often used as squaring the distance makes errors more obvious
    - Squaring also places progressively greater weight on objects that are farther apart
- Limitations:
    - Objects can be represented with a different scale in the dataset
    - Will not give a clear intuition about how well variables are correlated
    - Cannot discover variables that have similar behaviours, but at a different scale
    - Cannot discover variables that have similar behaviours, but in the opposite direction

Pearson's Correlation Coefficient

- Produces a result, *r,* within the range *[-1,1]*
    - 1: perfect positive linear correlation
    - -1: perfect negative linear correlation
    - 0: no correlation
    - $|r|$: indicates the strength of the linear correlation

- Properties:
  - Scale invariant: $r(x, y) = r(x, Ky), K\ is\ a\ real\ positive\ constant$
  - Location invariant: $r(x, y) = r(x\ C + y), C\ is\ a\ real\ constant$
  - Can only detect linear relationships
    - Therefore, cannot detect non-linear relationships

Mutual Information

- Can detect non-linear relationships
- Works with discrete variables
- Discretisation into bins is needed for continuous variables
- **Variable discretisation**
  - Domain knowledge: assign bins thresholds manually
  - Equal-width bin: bins have the same length
  - Equal frequency bin: bins have the same number of points
    - Sort values in increasing order
    - Take cut point as percentiles
      - i.e. 2 bins, cut pint is the median
      - i.e. 3 bins, cut point at 33.3% and 66.7% percentile
- **Entropy of a random variable**
  - Entropy is a measure of information content
  - $H(X) = -\sum_{i=1}^{\#bins} p_i \log p_i$
    - Where p_i is the proportion of points in the *i*-th bin
  - Properties:
    - $H(X) \geq 0$
    - Maximised for uniform distribution
  - Joint Entropy Example
    - Two variables, *X, Y*
    - X = {high, low, low, high, med}
    - Y = {low, med, high, med, low}
    - Can construct a contingency table

X/Y	Low	Med	High
Low	0.1	0.2	0.2
Med	0.2	0.1	0.05
High	0.03	0.03	0.08

    - Can then calculate entropy
    - $H(X, Y) = -\sum_{i=1}^{\#binsX} \sum_{j=1}^{\#binsY} p_{ij} \log p_{ij}$
    - Where p_{ij} is the proportion of points in the *{i, j}-the* cell
- $MI(X, Y) = H(X) + H(Y) - H(X, Y)$
  $= \sum_{i=1}^{\#binsX} \sum_{j=1}^{\#binsY} p_{ij} \log \frac{p_{ij}}{p_i \times p_j}$
  - $0 \leq MI(X,Y) \leq min(H(X), H(Y))$
- Normalised Mutual information:
  - $NMI(X, Y) = \frac{MI(X,Y)}{\min(H(X),H(Y))}$
  - Range produced is within [0, 1]
- Properties of mutual information
  - Measures the amount of information shared between two variables *X* and *Y*
  - If *MI(X, Y)* is large: *X* and *Y* are highly correlated

Classification and Regression

What Is Classification?

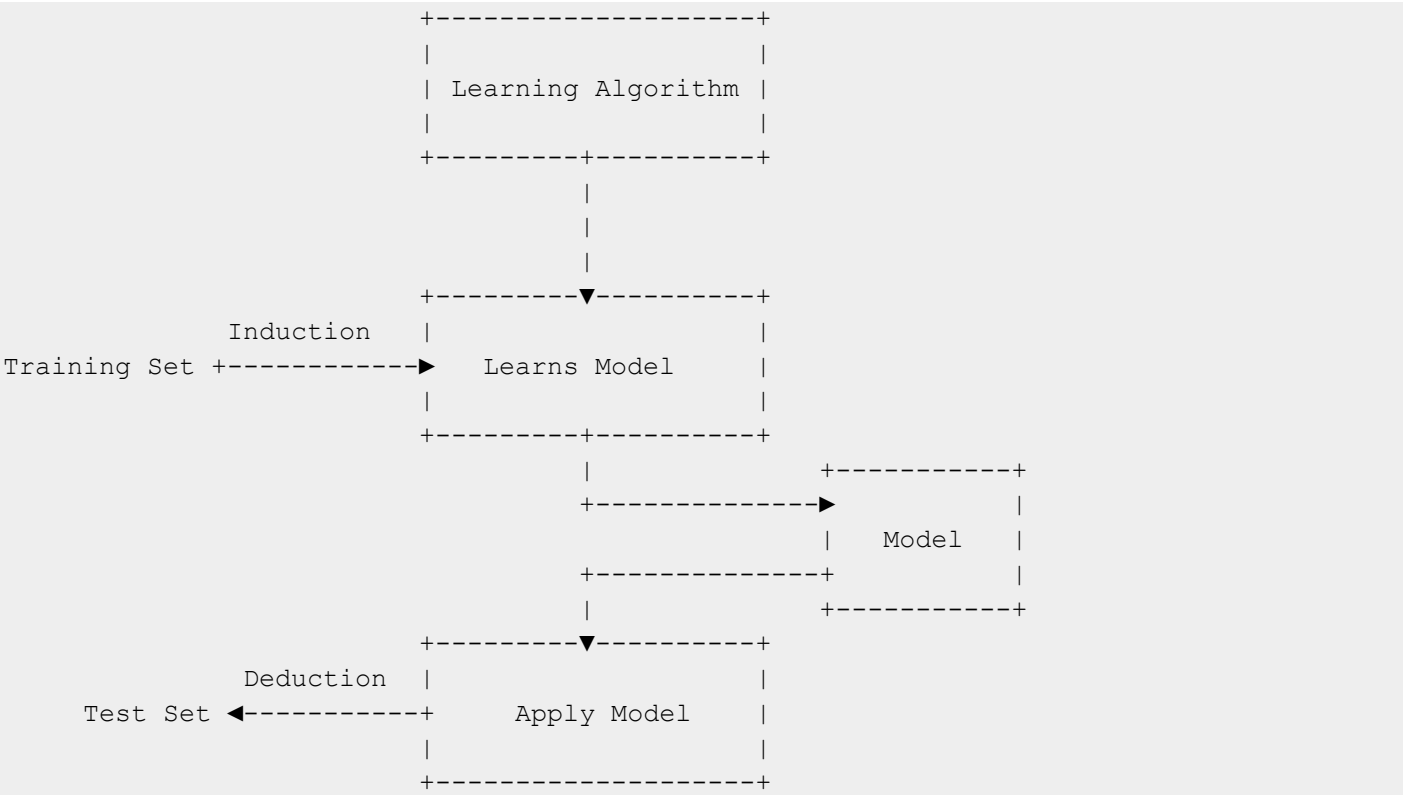
Technical Definition

- Given a collection of records **(training set)**
  - Each record contains a set of **attributes,** one of the attributes is the **class**
- Find a model for class attribute as a function of the values of other attributes, $y = f(x_1, x_2 \dots x_n)$
  - *y:* **discrete value,** the target variable
  - $x_1 \dots x_n$*:* attributes, predictors
- **End goal:** previously unseen records should be assigned a class as accurately as possible
- **Test sets:** test sets are used to determine the accuracy of the model.
  - Usually, the given data set is split into a training and test sets
    - Training set is used to build the model
    - Test set used to validate the model

Examples

- Animal classification
- Banking, classifying borrowers
- Detecting tax fraud
- Medical, predicting tumour cells and benign or malignant

Classification Framework



What Is Regression?

- Similar to classification, but for continuous variables

Technical Definition

- Given a collection of records **(training set)**
  - Each record contains a set of **attributes,** one of the attributes is the **target variable**
- Learn a protective model from the data, $y = f(x_1, x_2 \dots x_n)$
  - *y:* **continuous real value,** the target variable
  - $x_1 \dots x_n$*:* attributes, predictors

Regression Examples

- Stock market regression
- Predicting activity level of target gene

Classification Algorithms - Decision Tree

- A flow-chart like a tree structure is used for classifying
  - **Internal node:** denotes a test on an attribute
  - **Branch:** represent an outcome of the test
  - **Leaf node:** represents class labels or class distributions
- Need a tree induction algorithm to generate the tree based on training data
- Once the tree is generated, it becomes the model
- This model is used to classify other data

Hunt's Algorithm

- One of the earliest tree induction algorithms
- **General procedure:** letting D_t be the set of training records that reach a node *t*
  - If D_t contains records that belong the same class y_t, then *t* is a leaf node labelled as y_t
  - If D_t is an empty set, then *t* is a leaf node labelled by the default class, y_d
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset
- **Stopping condition,** at leaf nodes
  - If D_t contains records that belong the same class y_t, then *t* is a leaf node labelled as y_t
  - If D_t is an empty set, then *t* is a leaf node labelled by the default class, y_d

Determining How to Split the Records

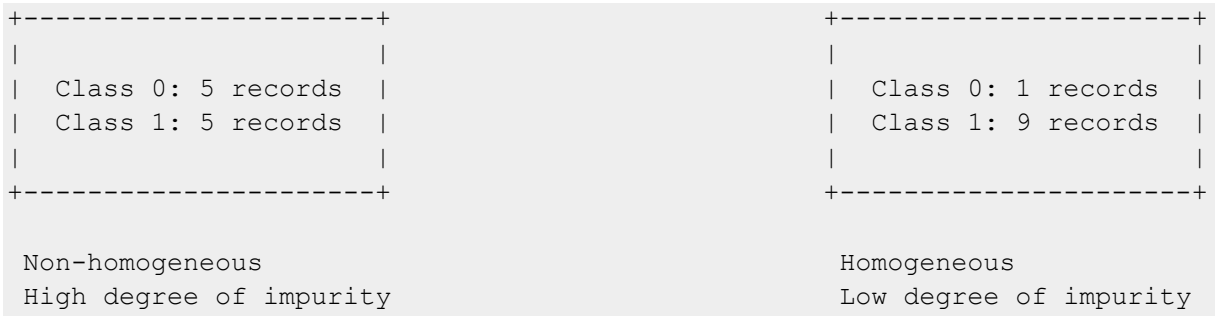
- Depends on attribute type
  - Nominal
  - Ordinal
  - Continuous
- Depends on the number of ways to split
  - 2-way split
  - Multi-way split
- Splitting options:

	Nominal & Ordinal	**Continuous**
2-way split	▪ Divide values into two subsets ▪ Need to find optimal partition	▪ Split into following: *(A < v) or (A ≥ v)* ▪ Consider all possible splits and find the best cut ▪ Can be more compute expensive
Multi-way split	▪ Use as many partitions as distinct values	▪ Discretisation, to form an ordinal categorical attribute ▪ Static: discretise once at the beginning

		▪ Dynamic: ranges can be found by equal interval bucketing, equal frequency bucketing or clustering

How to Determine the Best Split?

- Greedy approach:
  - Nodes with **homogeneous** class distribution are preferred:



- Need a measure of node impurity: **Entropy**
  - At a given node: $entropy(t) = -\sum p(j|t) \log p(j|t)$
    - Where *(j|p)* is the relative frequency of class *j* at node *t*
  - Measures the homogeneity of a node
    - Maximum when records are equally distributed among all classes
    - Minimum when all records belong to node class
  - Example (assuming 6 records need to be classified):

```
+-------+
| C0: 0 |  P(C0) = 0/6 = 0     P(C1) = 6/6 = 1
| C1: 6 |  Entropy = -(0 log 0) -(1 log 1) = - 0 - 0 = 0
+-------+

+-------+
| C0: 1 |  P(C0) = 1/6     (C1) = 5/6
| C1: 5 |  Entropy = -(1/6) log (1/6) - (5/6) log (5/6) = 0.65
+-------+

+-------+
| C0: 2 |  P(C0) = 2/6     (C1) = 4/6
| C1: 4 |  Entropy = -(2/6) log (2/6) - (4/6) log (4/6) = 0.92
+-------+
```

- How good is a split?
  - To determine how good a split is, need to compare impurity of **parent node *before* splitting** with impurity of the **children nodes *after* splitting**
  - Called **Information Gain,** Δ
  - $\Delta = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$
    - $I(v_j)$: impurity measure of node v_j
    - j: children node index
    - $N(v_j)$: number of data points in child node v_j
    - N: number of data points in the parent node
  - The larger the gain, the better
  - Therefore, to determine the best split:

- Compute the gain of all splits
- Choose the one with the largest gain

Classification Algorithms – Others

Random Forest

- Community of experts
- Involves training multiple decision trees on random subsets of samples
- The decisions are made via majority voting

Deep Neural Networks

- Designed to mimic the biological brain
- State of the are classifiers
- Used for images are speech data

Classification Methodologies

Nearest-Neighbour Classifiers

- Have some **training records**
- To classify a **test record** using the classifier choose *k* of the nearest records by computing distance between the test record and the training record

Requirements

- A set of stored records
- A **Distance Metric**
    - Used to compute distance between records
    - Often used:
        - Euclidean distance, $d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$
        - Pearson coefficient
    - Determining the class from the nearest neighbour list
        - Take majority vote of class labels among *k*-nearest neighbours
        - Vote can be weighted according to the neighbour distance
            - Weight factors, $w = \frac{1}{d^2}$
- The value ***k*,** which is the number of nearest neighbours to retrieve
    - The nearest neighbour is the data points that have the *k* smallest distance to a data point of unknown class, *x*

K-Nearest Neighbour Classifier

- **Lazy learning:** no models is learnt in the *k*-NN classifier
    - Just need to store some data
- Choices:
    - Must choose distance function
    - Must choose *k* value
- Challenges:
    - When there is a large number of points, storage space need to be considered
    - Nearest neighbour search cost need to be considered

Choosing the *k* Value

- If *k* is too small:
    - Classifier will be sensitive to noise points
- If *k* is too large:
    - Classifier may include points from other classes

Parameter Tuning

- Parameters that could be tuned:
    - *k*: the number of nearest neighbours
    - The distance metric
- Method:
    - Divide training data into
        - Training subset
        - Validation subset

- Train models on the training subset
- Evaluate performance on the validation subset

Performance Evaluation

- To evaluate the classifier, need to focus on its predictive capability
  - Not necessarily need to considered how fast it takes or scalability
- A **confusion matrix** can be used

		Predicted Class	
Actual Class		Class = YES	Class = NO
	Class = YES	True Positive (TP)	False Negative (FN)
	Class = NO	False Positive (FP)	True Negative (TN)

  - $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- Issues with accuracy measurement
  - If a 2-class problem with a large value in one class (i.e. 9990) and a small number in the other class (i.e. 10):
    - The accuracy will be predicted to be 99.9% if the model predicts everything to belong to the large class, which is wrong and misleading
  - There are other metrics that can be used instead of accuracy to evaluate the classifier
- Bias in accuracy evaluation
  - When evaluating the performance of a learning model
    - **The testing data should not be seen during model construction**

K-Fold Cross Validation

- More complex method of accuracy evaluation
- Method:
  - Partition data into *k* disjoint subsets, approximately the same size
  - *k*-fold: **train** in *k-1* partitions, **test** on the remaining one
    - Repeat this *k* time
    - Overall accuracy is the average of the *k* accuracies obtained
  - *k* is typically set to 10
- Provide a more robust evaluation of accuracy
  - Less dependency on a single set of testing data

Learning Curve

- Model performance generally improve with sample size
  - i.e. Greater accuracy as more data points are used
- A learning curve can be created to explore how accuracy changes with varying sample size
- Estimating learning model accuracy
  - Randomly partition a dataset into two parts
    - Reserve 2/3 for training (learning the model, optimising the parameters)
    - Reserve 1/3 for testing (accuracy evaluation)

Public Data Release

The Issue

- Public is concerned that computer scientist can identify individuals from anonymised data with ease

Data Release Examples

- Massachusetts Mid 1990s
    - Massachusetts Group Insurance Commission released records about hospital visits of state employees
        - Contained zip code, birthday, sex, hospital visits
    - Governor of Massachusetts ensured all personal information had been removed from the data
    - A PhD student went searching for the Governor's records, using voter roll
        - Found only 6 people with same birthday as Governor
        - Only 3 of those were men
        - Of these, only one lived in his zip code
- Census data
    - A study of records from the 1990 US census concluded that:
        - 87% of Americans can be uniquely identified by zip code, birthday and sex only
        - 53% of Americans can be uniquely identified city, birthday and sex only
    - This lead to changes in privacy legislation in the USA
    - In Australia, the Privacy Act 1988 covers release and use of census data
- Netflix dataset
    - In 2006 Netflix realised 6 years of data about its customers viewing habits
    - The aim was to build a better collaborative filtering algorithm
    - An anonymised ID was created for each user rather than using their names
    - Two researchers were able to correlate Netflix data with Internet Movie Database (IMDb) public data
        - This allowed for anonymous Netflix data to be linked with public IMDb data, revealing people's private movie habits within the Netflix dataset

Measures of Anonymity for Individuals

- Removing explicit identifiers from a dataset is not enough
- Terminology:
    - **Sensitive attribute**
        - Information that people don't wish to reveal
    - **Non-sensitive attribute**
        - One that isn't sensitive
    - **Quasi-identifier**
        - A combination of non-sensitive attributes that can be linked with external data to identify and individual

K-Anonymity

- A data table satisfies *k*-anonymity if
    - Every record in the data table is indistinguishable from at least *k-1* other records with respect to quasi-identifier attributes
    - Such a table is called a *k-anonymous* table

- When *k*-anonymity is applied the data table can be divided up into groups of common attributes
    - The minimum size of any of the groups is *k*
    - Can't be distinguished from any fewer than *k* individuals in the table
- For every combination of values of the quasi-identifiers in the *k-anonymous* table, there are at least *k* records that share those values
- Achieving *k*-anonymity:
    - Manipulate the records to remove the quasi-identifiers completely
    - Manipulate the records to make the quasi-identifiers less specific
        - i.e. Bin range values like age
            - Age 16 becomes 12 - 17
            - Age 43 becomes 35 - 45
        - i.e. Redact some numbers in a postcode, telephone number, etc.
            - Postcode 3010 becomes 301*
            - Postcode 2006 becomes 200*
- Benefits
    - In the worst case, identity can only be narrowed down to a group of *k* individuals
- Issues
    - Groups can be created that leak information due to lack of diversity in the sensitive attribute
    - Does not protect against attacks based on background knowledge
- Solutions
    - Make the sensitive attributes diverse within each group, called **l-diversity**
        - Ensure that there are at least *l* different values of the sensitive attribute in each group

Location & Trajectory Privacy

Trajectory Data

- Location data being collected and stored throughout the day
    - GPS-enabled smartphones, cars and wearables
    - Wi-Fi access points
    - Cell towers
    - Geotagged tweets, Facebook posts, location checks, Instagram posts

What Is a Trajectory

- A function of time to geographical space
    - Includes latitude and longitude and a time stamp
- Can be plotted on 3D graph

Benefits of Location Data

For Individuals

- Precise, tailored location services
- Monitoring of daily activities for fitness purposes
- Finding nearby friends
- Tracking children or elderly
- Traffic monitoring and navigation purpose

For Governments/Companies

- Identify the most frequent paths between toe points
- Provide best point of interest information for particular groups of people
- Improve traffic management and urban planning
- Enable personal data analysis

Privacy Concerns

- Current mobile systems:
    - Able to continuously monitor, communicate and process information about a person location
    - Have a high degree of spatial and temporal precision and accuracy
    - Might be able to be linked with other data to reveal sensitive information
- Analysing and sharing location datasets has significant privacy implications
    - Personal safety
        - Stalking, assault
    - Location based profiling
        - Facebook does this
    - Intrusive interferences
        - Could reveal a person's political views, personal preferences or health condition