Contents

1 Data Processing

Subject about data wrangling, controlling data. Iterative, not linear.

Data processing is the operations you do on the data. Data science is a set of techniques to manipulate large amounts of data to learn something. Data processing converts raw data to usable data, data science converts data to knowledge. Data processing is used in Data science. Data scientists spend the most time doing data processing and collecting data sets.

Challenges in data science are: variations in data in scope and quality, different methods to analyze data, quality of documentation in third party libraries.

Data processing is important because raw data has poor quality and structure. The main data processing pipeline is

- Data input
- Data cleaning and Enhancement
- Modeling and interpretation
- Data visualization
- Ethics

1.1 Data Types

Data is information we record and use. We aim to work with data by: capture and represent incoming data, turn it into information, and compute results with it.

Numeric values are stored in binary form (there're also other number systems). Textual data are also encoded in binary using text encoding (ascii and unicode).

Non-numeric data can be: symbols, strings. They are always discrete. These qualitative data includes ordinal data: ordering is meaningful but no meaningful distance. Nominal qualitative data are discrete information with no ordering. A special type of nominal data is categorical data of groups.

Numeric data are integers or reals. Discrete numeric data are countable, which can be categorical/nominal if there are no order, or ordinal if they are ordered. Continuous numeric data are sensor/real-world readings or function outputs. For these quantitative data, there are no fixed rule to differentiate discrete and continuous data. Numerical discrete can be modeled as continuous or categorical, and this treatment of one data type as another can be approximately correct. Numerical continuous allows distance measures and mathematical operations. They are however represented as discrete number of digits.

In-between data, or constantly changing data with large magnitudes can be treated as continuous despite being discrete.

1.2 Data Formats

Categories of data formats are: unstructured, semi-structured, structured. The more unstructured, the more human readable; the more structured, the more machine readable. Unstructured data include: text, audio, video. Semi-structured include: csv, html, xml, json. Structured data include: databases, spreadsheet.

We mainly focus on semi-structured data.

1.3 CSV

Comma separated values represents a spreadsheet (table) with columns separated by commas and rows separated by newlines. The first row contains the header, and the first column may be the index.

CSVs are easy to load and write using pandas (`pd.read_csv('file.csv')` and `df.to_csv('file.csv')`).

Used for numbers, stored in plain text, human readable with no formatting.

Tab separated values is csv with tabs.

1.4 HTML

Markup signals important aspects of text. The markup format used for websites is HTML.

Contains elements, which are defined by a start and end tag in angled brackets. Tags are keywords which are case insensitive. Elements can have attributes defined in its start tag. The order of attributes does not matter. Some elements only have one tag.

Important tags: html, head, title, body, table, tr, th, td.

1.5 XML

A generalization of HTML files and is also a markup language. It is used to model custom data.

XML has no predefined tags and elements, flexibility to define what the data looks like. It contains hierarchical data (objects and children) instead of relational data (entities and relationship).

XML must start with the declaration `<?xml version="1.0"?>`. Every XML file must have a single root element. Every tag must be properly closed, empty tags are allowed,

tag/element names are case-sensitive. Tags must be closed in the correct order. Elements can have attributes enclosed in double quotes.

Allows comments.

Both `<` and `&` are illegal inside an element, we need to convert them with XML escape codes (to `&` and `<`). CDATA (character data) allows large block of text inside an element, in which no escaping is needed. CDATA format is `<![CDATA[ text ]]>`.

1.6 JSON

Similar to XML, originally used to send data between client and servers. Represents data as objects and nested objects.

JSON is lightweight, standard, streamlined form of data exchange. Native javascript code. Used for any semi-structured data, but lacks schema.

It contains: JSON object, JSON array, JSON values.

- JSON objects is a key value pair. Key is in double quotes, Object in curly braces.
- JSON array is a list of values. Array in square brackets.
- JSON value can be: numbers, string, boolean, array, object, null.

JSON schemas are metadata that enforces the structure and types of other JSON data. The schema itself is written in JSON and is stored as a JSON file; its structure is also similar to the structure it defines.

We can load and write data in JSON in Python. And we can convert between XML and JSON.

1.7 Comparisons

HTML is not extendable, and is mainly focused on formatting and not semantics.

XML allows complex schemas, is extensible, elements representing semantics.

JSON is more lightweight and compact, speed and efficiency, noSQL.

1.8 Metadata

Data has syntactic types and semantic types.

Every data stands for some real life object, so in addition to data types, data also has a semantic type: the meaning of the data, the computations sensible on it. Data's denotation is its mapping to a world object.

Large programs/databases use metadata schema: structure of all the types with their definitions. This is because there are a lot of duplicated syntactic types with different meanings.

A taxonomy is a metadata schema. It is unbounded in depth, bounded in width. An ontology is a specific populated metadata schema with lots of information.

Knowledge is concepts and relationships between them. A knowledge graph is a structure representing data about the world. We can use JSON to define both the concepts, relations, and actual data.

Knowledge graphs represent data about the world; Ontology represents your understanding of the world.

2 Data Capture

2.1 Structured Data

Structured data typically contains numbers or symbols: time series, records of instances of the same type.

Spreadsheets contain tabular data with validation capability: excel, openoffice calc.

To ingest/parse structured data

1. Determine input data layout
2. Determine semantic types of each unit (column headings)
3. Define internal data structure to store the records
4. Write code to ingest line by line

To reduce the number of types/columns for a spreadsheet, and to better structure your data, use a relational database that splits the data into smaller interconnected tables. The steps are to: setup the database, saving the data into the database, then querying using SQL.

A relational database is easier to wrangle than spreadsheets, but spreadsheets are easier to create.

2.2 Data preprocessing

Raw data comes in various formats, with missing entries. Preprocessing aims to make the data more consistent.

Main steps of data preprocessing is

1. Determine the forms of raw data
2. Define data types
3. Write data ingestion code
4. Ingest data into types and records to make it structured
5. Perform necessary preprocessing: data cleaning, data integration, data reduction/sampling, data transformations
6. Store it

Data cleaning use existing tools to do: data scrubbing, discrepancy detection, auditing, extract transform load (ETL) using a graphical interface to transform. Domain knowledge is important.

Data quality is determined by

- Accuracy, is it correct
- Completeness, are there missing data
- Consistency, discrepancies in representation
- Timeliness, is it updated
- Believability, trust
- Interpretability, easy to understand

Data inconsistencies can be

- Different naming representations or formats
- Equivalent info
- Wrong info
- Outliers

Data rescaling is for when you want different sets of data with different ranges to be in the same range to compare. Normalization is to rescale the values to be within $[0, 1]$ or $[-1, 1]$, and is used when the numbers have different units

$$(x - \min)/(\max - \min)$$

Standardization is to rescale the values to zero mean and one standard deviation. This preserves the internal ratios and is used when two sets of numbers in the same unit are in very different ranges

$$(x - \mu)/\sigma$$

Importantly, use normalization if the data are on different scales with bounds, and standardization if they are on similar scales without bounds. Normalization is helpful when the feature distribution is unclear, while standardization is good when it is consistent (similar distributions).

We can also use other scaling transformations for feature engineering. This may obscure relevant details, however.

Data entity resolution is about the clarity of the data. We need language processing to understand the context.

2.3 Missing data and sampling

Missing completely at random (MCAR) is when the probability of missing data is unrelated to any other measurements and unrelated to the data itself. It is unpredictable.

Missing at random (MAR) is when the probability of missing data is related to other measured variables but not with itself. Make a rule that the data is missing based on the values of another data. This is kinda predictable. (Can predict missing data better than guessing)

Missing not at random (MNAR) is when the missing data are related to the value itself. Can make a rule that accurately predicts what's missing. (Can accurately predict missing data)

Disguised missing data are placeholder data that is not denoted as empty. Handle by looking for unusual values in the dataset. Requires domain knowledge.

Imputation is to guess missing data. Statistical measurement imputation uses statistical strategies to fill out numeric records: mean, median, mode. This is easy to compute, but biases other statistical measurements like variances. Time series data may use average of endpoints or a curved line. Sklearn SimpleImputator allows mean, median, and mode imputations.

Sampling is the technique of selecting a representative sample from all data. We sample when the dataset is too large to efficiently analyze in full. If we sample too little there will be sampling error and the results are not trustworthy. The sample size depends on the type of data.

Random sampling is to randomly select the data points. With/without replacement both works.

Stratified sampling is to split the population into stratas on the relevant features (gender, race), then random sample within each group to net a representative sample. We usually do this to get a representative sample for under-represented groups.

Samples must be representative (same properties as population) and balanced (not excluding). They must also be large enough to be trustworthy, but avoid biases and dishonesty.

2.4 Web API

Basic methods of gathering data from the web are

- API, using the website owner's instructions to get structured data from their website. Has limitations on data extraction
- Crawling and scraping, write a spider bot to crawl the web, following links and collecting content. Then extracting info from that content. Is uncontrolled.

Representative state transfer (REST) API is a set of protocols that format the exchange of info between server and client. It includes metadata header, and JSON, XML, or CSV data. Uses the HTTP protocol.

A request has a method type and a uniform resource identifier (URI). The format is `protocol://servicename/resourcetype/resourceid`. The HTTP methods include: GET, POST, PATCH, DELETE. The URL points to the API service, and URI points the specific resource. The HTTP request headers are

- Accept, the format that the response data is in
- Authorization, token showing permission
- Content-type, format of the request
- User-Agent, client info

The HTTP response contains

- Header, what to expect in the response content
- Content, data
- Status code

2.5 Web crawlers

Web crawlers are spiders or robots. They attempt to visit every page of interest and retrieve data for indexing. They disregard synonym urls and more frequently visit dynamic pages. Problems with web crawling are

- No central index of wanted URL
- Old url have same content as new url
- No success status

- Websites not intended to be crawled
- Content provided on the fly, or content with short lifespan

URLs are crawl targets, they are `protocol://hostname.domain.tld/path/file`.

We can treat the web as a graph where pages of interest have links as edges. The web crawling algorithm is

1. Start with a list of seed URLs
2. Visit their pages to collect all outgoing links
3. Crawl and Use new pages as new starting points
4. Repeat

The algorithm retains a list of URLs to visit and a list of visited URLs. We crawl by selecting the most needed URL from the list and fetch the page, parsing and extracting the URLs that is new into the crawling list, then adding the URL to the visited list. Occasionally moving expired URLs back into the visiting list. This can be BFS, DFS, or best first search.

Considerations of web crawling are

- Malicious pages like: spam pages, spider traps like a calendar that is dynamically generated and can be infinitely followed
- Latency bandwidth, depth of crawl, site mirrors

The robot exclusion standard is a protocol that crawlers are recommended to follow. Website managers can restrict access to crawlers while allowing browsing. This is robots.txt that is developed by Google. The fields are

- user-agent to specify which crawler the rules apply to
- allow is a url that maybe crawled, disallow not to crawl
- sitemap for a complete url list of the website to make a crawler easier to all links on the site

Things that must be followed when crawling

- Be polite, only crawl allowed pages, respect robots.txt
- Be robust, be immune to spider traps and malicious server behavior
- Be legal, respect copyright and ownership

Things that you should do

- Be distributed and scalable, run on multiple machines
- Be efficient, fetch higher quality pages first
- Be up to date, refetching old pages
- Be extensible, adapting to new formats

2.6 Document scraping

The scraping pipeline is

1. Crawl the web for documents
2. Scrape info out of documents
3. Clean and save info

Raw text needs NLP, but html and xml are semi-structured so use libraries to scrape.

Scraping starts with a document, and extracts data out of it. It is targeted for you specify the input and output format.

Common to clean text markup. Process is: remove file preamble, compress whitespace, strip out markup. To handle readable markups, we can use a markup processer or a software package.

To scrap HTML or XML, use beautiful soup to extract text. First we apply a parser that parses HTML, then run beautiful soup to extract the relevant pieces. It contains both HTML and XML parsers by default.

To parse a textual document is to create a class instance with the text. The parser contains an AST of the html document. We can them perform DOM lookups using the instance.

The scraping pipeline using beautiful soup is

1. Get HTML document
2. Decide the info to capture, define the fields in the output spreadsheet
3. Parse the document
4. Write extractor patterns to find the info needed and save in the database

3 Text Preprocessing

Focusing on unstructured data, specifically text.

The unstructured text pipeline is

1. Left with strings of text from crawling and scraping
2. Preprocess the text
3. Then process the text

The preprocessing sequence (usually for NLP) is

1. Sentence splitting
2. Tokenization and normalization
3. Word regularization
4. Ngram processing and string matching

3.1 Sentence splitting

Aims to split text into sentences.

Logically, we can split the text when two words are separated a specific symbol, and the second string starts with a capital letter. But there are exceptions for numbers and abbreviations where we shouldn't split. But there are exceptions to exceptions when the abbreviation ends the sentence.

3.2 Tokenization and normalization

Tokenization is to split a sentence into word tokens. The delimiter is usually a space.

The problem with splitting by spaces is that we will group punctuation into words. So we will strip off punctuation as separate words. The algorithm will split the sentence into word tokens separated by spaces, grouping punctuation into its own word. There are exceptions like "don't".

We can also tokenize using regexp.

Normalization is to convert all text into a single case (all lowercase). It helps to reduce sparsity in the token space by converting words with the same lowercase into a single token. It is simple and effective, and supports searching and matching on the result.

Normalization is also the transformation of text into canonical forms. For noisy text sources like comments, emails, text messages, it converts different representations (abbreviations, misspellings, out-of-vocab) of words into the same token.

The use of normalization will lose data. So we conduct them only if it is helpful to our future processing.

3.3 Word regularization

English has different forms of the same word due to: number, tense, aspect. Words are often composed of a root or stem (base part) plus more parts (morphemes, extra parts). We can reduce the size of our lexicon by removing the morphemes and treating words with the same root the same.

Stemming is the simple approach. Lemmatizing is a better approach.

- Use a stemmer to chop certain parts off words (removing the morphemes ideally). This is not a perfect solution
- Use a lemmatizer to convert each word to their root form (lemma). This is more than just a splice for some words must be completely changed

In general, writing code to handle all the edge cases of a language is difficult; all language have exceptions.

Stemming replaces word suffixes in trying to create a common root form. The consequence is that the results do not look like proper words. We will use Porter Stemmer that use replacement rules to stem for us.

Lemmatizing removes inflections (grammatical parts of a word) and maps a word to its proper root form. This is also called demorphing. There are manual demorphers with builtin rules, and ML demorphers.

Content words (nouns, verbs, adjectives, adverbs) give content information, they are called open-class words. Function words (determiners, pronouns, preposition, conjunction) give grammar, they are called closed-class words. It is likely that we just want the content words to cluster topics.

Stop words are close-class words that we wish to remove. A stopword list contains a list of stop words. The point is to reduce the number of features/words to allow more accurate clustering. We often use stopword lists before searching, classifying, topic modelling, topic extraction.

Stopword lists can be custom-made for a specific context or domain.

We don't always need to remove stopwords in NLP, for instance, they may convey sentimental meaning for sentimental analysis.

3.4 Ngram processing

A useful concept for text processing.

Word ngram (or just ngrams), is a series of length n words (n consecutive words). Letter ngram is a substring of length n letters in a word.

We use word ngrams to compare between sentences, and use letter ngrams to compare between words.

Let $G_n(w)$ be a function that returns all letter ngrams of a word w. Note that this is a set of consecutive n letters including the beginning and ending paddings as a hash.

Define the ngram distance between x and y as

$$d(x, y) = |G_n(x)| + |G_n(y)| - 2|G_n(x) \cap G_n(y)|$$

Ngram distances are useful for approximate prefix and suffixes (usually in abbreviations). They are problematic in

- less sensitive (score variance) to relative ordering of strings
- more sensitive to long substrings, some words with zero score (due to more matching), and other words with high score, not a smooth transition
- quite useless for long strings or small alphabets for there are lots of matches by accident

We can use letter ngrams to compute similarity/distance between words. Define similarity as one minus distance. The metrics to measure similarities are ngram distances or edit distances.

3.4.1 Ngram similarities

Jaccard similarity is

$$sim_{jac}(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

Sorensen-Dice similarity is

$$sim_{dice}(S_1, S_2) = \frac{2|S_1 \cap S_2|}{|S_1| + |S_2|}$$

Cosine similarity converts ngrams to vectors, where an entry is k if the ngram appears k times in G_n. The metric is

$$\frac{X \cdot Y}{||X|| \times ||Y||}$$

notice both the dot product and magnitude are computed on the combined entry index vector (0 for an index indicates the ngram doesn't appear). An equivalent procedure is to divide the size of the intersect of ngrams by the square roots of each word's ngram set.

3.4.2 Minimum edit distance

Transform the source to target using insert, delete, replace, in the least number of step. Define the edit distance as the least number of steps taken to transform the source word to the target word, sometimes with a different weight on each of the operation.

Levenshtein distance is a dp method to compute the minimum edit distance.

Similarity in the edit distance metric is

$$sim(S_1, S_2) = 1 - \frac{d(S_1, S_2)}{\max(|S_1|, |S_2|)}$$

where $d(S_1, S_2)$ is edit distance and the fraction is the normalized distance.

3.5 String matching

To approximate matches for a pattern in a text, we select the words with the lowest distance.

String matching is used for spellchecking, neologism (new words), word equivalence. We use a dictionary that is a list of correct words, then compare each word against the dictionary. If the word is not found, the closest words are the likely fixes to the typo.

Spellchecking uses a dictionary and check each word.

Neologism are new words, which are just newly created words or word blending (merging two existing words). We cannot use a pre-existing dictionary, and would need to dynamically insert new words when they are introduced.

Word equivalences are equivalent words with different spelling. They can be US/UK spellings, or abbreviation. We can use a dictionary to solve this.

3.6 Regex

Aim to find a pattern in a large unstructured text. A regexp is a pattern that can match all kinds of substrings depending on the specified rules. It is used to find desired items, calculate statistics, or for filtering.

You know regex you dont need this part. Here are some odd features

- Metacharacter is a character that does not represent itself
- Character sets are in square brackets, pre-defined sets are the d, w, s ones
- Grouped patterns are captures and numbered (index 1), we can specify back references to capture groups by escaping the number (this is match the exact capture group match)

- Lookaheads are placed after the pattern and specify what happens after a match. Positive lookahead ensures that an expression after must also match if the preceding pattern can match. Negative lookahead ensures that an expression after must not match if the preceding pattern can match

4 Document processing and Visualization

Assume we have access to the pre-processed spreadsheets and cleaned text. Our goal is to perform analysis on the data: gaining information from it and visualizing it.

Text, while being unstructured, actually has structure. There are units (words) in groups (conseq words). Each unit carry information by themselves, with groups also carrying some information. We have to determine the groupings and roles ourselves.

We aim to get a representation for the whole document that is structured, easy to produce, and easy to manipulate.

4.1 Bag of words

The bag of words model is the simplest representation of a document.

The bag of words model treats a paragraph/document as a probability distribution of words in a lexicon space, and generate a feature vector where each word in the lexicon is a dimension and the value for a dimension is the number of its occurrence in the document.

To get an accurate counting, we need to clean up the text by tokenization, normalization, lemmatizing it to the root form, stopword removal, punctuation removal. Each will reduce the size of the lexicon space.

The benefits of BoW

- Simple representation of just integers
- A long vector
- Easy to produce using preprocessing tools
- Used to compare documents
- Works for all languages and emoticons

The problems of BoW

- Paragraphs with the same counts may not convey the same context (ordering of words are ignored)

- Word context are missed due to case folding and others (the white house vs The White House)

To represent BoW, we can

- Use a fixed vector of around length 100000. Every cell will be clearly defined, we can also add extra information to each position. But it can't handle new and unknown words.
- Use a list of pairs (word, count). Need more storage than the counts, but will be shorter for we only store the words we see. Can handle new words

When representing a list of documents with BoW, we get a sparse matrix. We can store a sparse matrix with many zeros by the compressed sparse row format, where we store the values of the non-zero entries in one list, their rows in another list, and their columns in another list.

4.2 Term Frequency and Inverse Document Frequency

TF.IDF is a score that can be used in the BoW model instead of raw term frequencies. It better captures the rareness and frequency of the word.

Each document is a random sample of the set of the English lexicon. We are interested in the more frequent words in the same document and rare words across document.

Raw counts are not good enough because

- Raw frequency varies on the document length (longer documents have more counts)
- Counts don't capture important (rare) words that is unusual which can identify the type of document

Rare words are words that are specific to a given type of document. They are usually less common across documents. If a common word occurs in many documents, we can either delete them as stopwords, or keep them but downweight them.

Under raw frequency, two documents with similar BoW vectors are probably talking about the same topic. But some words are more helpful than other words in similarity and document clustering — the rare words.

Document clustering aims to group documents by their topics or themes. Clustering algorithms take a set of BoW vectors as inputs, and outputs a tree showing the clustering. Usually, they use simple term frequency counts (raw frequencies).

When making a document cluster, rare words are more useful in making the clusters (we make clusters based on the shared rare words). And we can identify the cluster a new document is in by its rare words.

To reward rare words, we compute

$$df_w = \frac{\text{number of clusters with term w}}{\text{number of clusters}}$$

so that the rarer the word, the lower the document frequency. This is the document frequency. However, we are interested in the inverse document frequency which swaps the numerator and denominator. IDF is higher as the document frequency decreases.

Because a word is stronger if it is more frequent in the document, and more helpful if it is more localized to a few clusters, we multiply the term frequency and inverse document frequency to get its score/weight

$$w = \text{tf} \times \text{idf}$$

But idf is too sensitive, so we take the log of the idf

$$w = \text{tf} \times \log \text{idf}$$

The formula for tf-idf is

$$v(w) = \text{count}(w) \log(\frac{\text{total number of clusters}}{\text{number of clusters with } w})$$

but this is problematic if: document sizes are not equal, one document has more clusters than another. The term frequency and cluster frequencies will be higher for longer documents with more clusters, even though the relative frequencies of the words are equal to a shorter document.

We fairly compare the tf-idf, we normalize the TF.IDF by dividing the raw score by the sqrt of the sum of squared of all the weights in the document

$$\text{tf.idf}(w) = \frac{v(w)}{\sqrt{\sum_t v^2(t)}}$$

the normalized score is between $[0, 1]$.

There are many formulas for term frequency, inverse document frequency, and normalization. The TF formula we use is a simple frequency, the IDF formula we use is $\ln(\frac{1+N}{1+df(w)})$ where N is the number of documents and $df(w)$ returns the number of documents with the term. The normalization we use is a simple Euclidean distance (sqrt of sum of squares). The entire process is

- Compute term frequencies
- Compute IDF
- Compute raw scores

- Compute normalization terms on scores, and the normalized scores

We can use sklearn's TfidfTransformer to do this automatically. The inputs are the raw BoW vectors, and the outputs are the normalized tf.idf scores with the same dimension.

By replacing the term frequencies with TF.IDF scores, we generated features for the text document. We can use these scores for EDA, ML, and analysis. Importantly, the features for structured data are the column headers, and the TF.IDF features for unstructured text data are the words for each vector dimension.

4.3 Document Similarity and Vector Spaces

A document is a long string of words, are there document similarities? The benefits of similarity metrics between documents are that

- We can sort them into clusters
- Use one document to find similar ones
- Can put all documents in a collection into a big space and arrange them by similarity in that space
- Given some search words, find the document we want

Because a document as a bag of words is a long vector of scores, we can place this vector in a vector space, with a document corresponding to a point in that space. If we use TF.IDF scores for the BoW, then the vector is Euclidean normalized (length 1) with all values between 0 and 1.

We need a distance metric in the vector space to define document distance. A distance metric takes two vectors in the space and returns the distance number.

The manhatten distance is the L1 distance. It does not use a straight line and measure the distance if walking along the grid. It is the sum of the absolute differences between each dimension

$$d(v, w) = \sum_i |v_i - w_i|$$

The euclidean distance is a straight line distance called the L2 distance. It is

$$d(v, w) = \sqrt{\sum_i (v_i - w_i)^2}$$

which generalizes to all dimensions.

The general metric for all dimensions and spaces is the Minkowski distance

$$d(v, w) = \left(\sum_{i}^{n} |v_i - w_i|^p \right)^{1/p}$$

where p is a number of dimensions inherit of the space (not the vector dimension). If $p = 2$, this is the eucldiean; if $p = 1$ this is L1.

All the three metrics above are normal metrics with the properties: non-negative, identity, symmetry, and triangle inequality.

If the TF.IDF count is not normalized, there can be documents with similar words/topic but with different lengths, with an (incorrectly) large distance. To overcome this non-normalization problem, we use the cosine similarity. The cosine similarity uses the angle between the two vectors of the document. The closer the two vectors are in direction, the smaller the angle is. The cosine similarity between two documents A and B is computed by

$$sim(A, B) = \frac{A \cdot B}{|A||B|}$$

The dice similarity is

$$sim(A, B) = \frac{2|A||B|}{|A| + |B|}$$

The jaccard similarity is

$$sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where the intersection is computed by the number of shared non-zero cells, and the union is the total number of unique non-zero cells.

There are many metrics and no best metric.

Once again, we can use distance for

- Retrieving documents
- Identifying near duplicates
- Clustering documents
- Topic modeling by finding themes in data
- Trend analysis, seeing if documents change overtime

Word embedding is another approach to perform feature extraction on documents. It uses machine learning techniques (neural networks) to convert documents into dense feature

vectors of the same length, where word order semantics are encapsulated (meaning of consecutive words are captured). Then two similar documents will have small vector distance between them.

4.4 Visualization

Vector spaces can be for both structured and unstructured data. The features are the columns or words, and the values are the feature values and tf.idf counts for both structured and unstructured data.

Data driven hypothesis generation is bottom-up, you form hypothesis from the data. Hypothesis testing is top-down, you have a hypothesis, then use the data to prove or disprove it.

EDA is a visual method to find structures, using graphing and plotting, because human eyes are more powerful to detect visual structures. Visualizations are good for

- Analyse data and reveal patterns
- Communicate data to others

we aim to use effective tools to encode quantitative and categorical information that is easy for others to visually decode.

You know what a distribution is. You know what the measures of locations are. You know what the measures of dispersion are.

Pandas has a describe method on the dataframe, it can produce all descriptive statistics we need.

For univariate data, we can use a histogram, binning, and boxplots.

Histogram has the x-axis divided into consecutive non-overlapping equal width intervals, and the y-axis containing the frequency of values within each bin. We can also use categorical histograms; we should use the same scale for all subplots.

Histograms of the same datasets may look different for different bin sizes. It may be hard to find an appropriate bin size. If it is too small, normal objects in rare bins, false positives; If it is too large, outliers in frequent bins, false negatives.

Outliers using the inner and outer fences.

Tukey's box plots contains: median and quartiles, whiskers within the inner fence, the inner fence and suspected outliers, the outer fence and outliers.

For bivariate and trivariate data, we can use scatter plots.

Scatterplots plots between two paired numeric variables that shows some relationship. Scatterplot shows the correlation direction and strength.

When there are too may data points, the dots may overlap. We can fix this by: reducing dot size or jitting (applying a small random noise). We can also plot a sample of the data points.

When there are three variables, we can use a contour plot where values along a contour have the same z-value. We can also graph a density plot by setting the z-value as the density of the 2D data points.

When differentiating between different data categories, we can use: colors, shape, size, movement. We can use scatterplot matrix, heatmaps, and clustering to find pattern via visualization.

A scatter plot matrix plots a scatterplot for each pairwise feature. It can show relationships between pairs of data points.

A heatmap graphs 3d data where the z-values are represented by colors. We should standardize every number plotted on the heatmap before plotting. This is useful when the objects are grouped/sorted to see clear patterns.

The elements of good visualizations are

- Scales should be the same when comparing data
- Suitability of the dataset, and context of the data
- Clear and by easily interpreted on its own: captions for context, dataset, and brief description, title
- Has no redundant information

Visualization can summarize the data. It can make it easy to identify important features, and is a visual tool for analysis. It is excellent for communication.

5 Clustering and Principal component analysis

Clustering is to take a large dataset and find naturally grouped together data.

A challenge when dealing with large datasets is that it is difficult to visualize data with a lot of features, so it is really hard to manually cluster groups together. We are hoping to automatically determine the significant groups in the dataset, so we can better understand it and can apply separate interventions to each group.

Applications of clustering include

- Market segmentation
- Image analysis
- Search engine result presentation
- Outlier detection
- Personality types

We would need to find the useful clusters (clustering), and not be overwhelmed by too much detail (PCA).

5.1 Clustering desiderata

The desiderata (desired features) for clusters are

- Each object belongs to exactly one cluster
- Similar objects are in the same cluster, and different objects in different clusters
- Each cluster is summarized by its centroid, the average of its objects

There are many clustering algorithms, each trying to achieve the same desiderata. All of them will use some type of distance metric: intra-cluster distances are minimized, inter-cluster distances are maximized.

We represent each object as a vector, then use the Euclidean distance between vectors as the metric. We should normalize/scale each attribute into the zero to one range before computing distances, so that we don't have one very large feature dominating the distance.

5.2 K-means clustering

An unsupervised way of creating k clusters.

The algorithm is

1. Choose the number of clusters wanted k
2. Randomly picking k cluster centroids (just points here), sampling from the data points
3. Assign each data point to its closet centroid, so each centroid is associated with a set of data points
4. Within each cluster, move its centroid to the updated centroid using the associated data points (taking the mean of the data points to generate a new centroid)

5. Repeat until the centroids are fixed (or the data point centroid association are unchanging)

This is guaranteed to converge to a fixed point. Each centroid and its associated data points forms a cluster.

Usually, distinctly grouped data sets will require fewer iterations of k-means clustering than more spread out and uniform data sets. But this heavily depends on the location of the initial speed centroids.

Because the initial seed points are assigned randomly, different runs of the algorithm will produce different clusters. The algorithm will also quickly converge to a local minimum with few iterations. The local minimum is on the cost that is the intra-cluster distance and inter-cluster distance.

An application of clustering is outlier detection. We can cluster a dataset, and the data points that are the furthest away in its cluster (using distance to its cluster centroid) can be considered as outliers. We can set a threshold for the distance and filter the potential outliers. This obviously depends on the number of clusters.

Always normalize the data before k-means clustering.

K-means clustering have problems with

- Different sizes of clusters (will eat into the other larger cluster)
- Different cluster densities (all the smaller high density groups are in the same cluster, while the low density group contains many clusters)
- Non-circular cluster shapes

We can fix this by using many clusters (increasing K), and merging close clusters together.

The elbow method is a way to choose the best K. We aim to find the K with the tightest clusters, where the tightness is measured by the average distance of points to its centroid. Then we can iterate over all K and select the one with the lowest distance. More specifically, the tightness measure is the "within cluster sum of squared distance" (WCSS)

$$WCSS(k) = \sum_{c=1}^{k} \sum_{x_i \in c} |x_i - \bar{x}_c|^2$$

where $\bar{x}_c$ is the centroid of cluster c. This is also called the SSE. The elbow method says to select the cluster number k when the distance curve stops decreasing (ie, at the elbow).

5.3 Visual assessment of clustering tendency

VAT is a visual approach to figure out the number of clusters the data set has naturally, as well as the rough size of them, and the rough data points belonging to each cluster.

A dis-similarity matrix using a vector distance function contains all pairwise distances between data points, where we assume each data record is of a vector form. It tells you how similar/dissimilar objects are.

The important properties of dissimilarity matrices

- The diagonal has all zeros
- It is symmetric about its diagonal, if the objects follow the same order in both row and column

The idea of VAT is to produce a heatmap that represents information using n by n image. We will graph a heatmap on the pair-wise similarities/dis-similarities of data points using the dissimilarity matrix. Essentially, the VAT is a heatmap plot and visualization of the dissimilarity matrix, with the cell color representing the cell value. Usually larger values (more distant pairs) are hotter/lighter, while closer pairs are darker.

The raw VAT is not enough to reveal the features. We need to order the data points by similarity and thus grouping more similar points closer in order on the matrix. The reordered matrix can reveal clusters in the form of dark black squares on the diagonal, because we will only see a dark diagonal square when a tight group with low within-cluster distances/dissimilarities and high outside-cluster distances exist in the data.

A good VAT image containing clear dark blocks will suggest both the number of clusters and members in those clusters. Lighter blocks indicate less tight/defined clusters that can be ambiguous. Blocks within blocks may indicate nested clusters.

When reordering the dissimilarity matrix, we aim to group similar points together. The algorithm is

- Start with the two most distance point pairs, select one point in the pair to grow
- Grow the group around the point by adding the closest unseen point to the group, repeat until all data points are seen

This is like prim's algorithm for the MST, where the node visiting order is the ordering of the points.

Formally, the VAT algorithm uses sets and set operations, and produces a re-ordered dissimilarity matrix D^* where closer points are grouped next to each other.

The VAT algorithm is not effective for all situations. For complex shaped data sets, with significant overlaps or irregular geometries, the quality of the VAT can be bad (blurry).

5.4 Hierarchical clustering

Produces a set of nested clusters organized as a hierarchical tree (where each cluster can contain many smaller nested cluster).

A hierarchical cluster can be visualized as a dendrogram, a diagram that records the clusters and the sequence of merges of the clusters. The x-axis contains the points, the y-axis has the dissimilarities where the clusters are merged at (the same as the cluster distance between the two nested cluster).

We can cut the cluster at a specific distance to find all the clusters with a distance smaller than the specified. This can be done by a horizontal line on the dendrogram and selecting all clusters under the line.

The main algorithms for hierarchical clustering are

- Agglomerative, start points as individual clusters, at each step, merge the closest pair of clusters until there are k clusters left
- Divisive, start with one inclusive cluster, at each step, split a cluster until there are k left

we would need to define cluster distance (and also point distances, which we already have) to implement these algorithms.

We can define cluster distance as either: the distance between average points of clusters, the distance between min points between clusters, the distance between max points between clusters. The different cluster distance metrics will give different resulting cluster shapes.

To implement agglomerative hierarchical clustering under a specific cluster dissimilarity and point dissimilarity, different cluster distance metrics defines different algorithms

- Start with a proximity matrix with pairwise cluster distances
- Initial clusters are just points
- At each step, merge the two clusters with the smallest cluster distance, update the proximity matrix by merging the two cluster labels (and distances)
- Repeat until we have the desired number of clusters left

The linkage defines the cluster distance

- Single linkage implies the use of minimum point distance between the two clusters
- Complete linkage implies the use of maximum point distance between the two clusters

Reflection on different linkage

- Single linkage can handle non-similar shapes of clusters because it merges closely grouped clusters, but it cannot handle intermingled clusters with noise and outliers (points in one cluster that leaks into another cluster).
- Complete linkage is less susceptible to noise and outliers because outliers don't contribute to the distance much, but it cannot handle clusters with different sizes, as it tends to create spherical clusters with consistent diameters and can break up a large cluster.

these two approaches have opposite strength and weaknesses.

Note that when two clusters are combined in the algorithm, it cannot be undone. So if an object is in the wrong cluster due to an early step, it is never fixed. Also, the algorithm is not minimizing an objective function, so there are no optimal solutions or "better" clusters.

5.5 PCA, Dimensionality reduction

The true dimensionality is usually less than the observed high-dimensional data, because there can be lots of irrelevant features.

The curse of dimensionality is that data analysis techniques that work well at lower dimensions often perform poorly when the dimesionality of the collected data increase. Additionally, if the number of features are high, the space of data becomes more sparse, and distances tends to be equal between points.

To ignore irrelevant features, we perform feature selection

- Select the most useful features from the original based on correlation
- Sampling on features
- May require domain knowledge
- The output features does not need to be a subset of the input features, but can be new features that are functions of the original features

Without domain knowledge, we have mathematical techniques that automatically identifies the most useful independent features of the data — each feature is a semantic primitive (independent meaning) in how the object is understood. This is a transformation of the data points to a new space, and it should preserve the properties of

- Close pairs should be preserved under transformation
- Far pairs should be preserved under transformation
- The set of the nearest neighbours (closest points) for points are also preserved

This automatic principal component analysis has the name: Singular value decomposition (SVD) in eng, and latent semantic analysis (LSA) in psy. We use this for text categorization, finding the most reliable topic signatures, etc. In detail

- PCA finds the new set of features that better captures the variability of the data
- Each dim is a linear weighted combination of features. They are orthogonal and independent, and has an importance value
- We order the resulting dimensions by strength, the first will capture as much variability as possible, the second also does that but is orthogonal, etc.

In general, we want the strongest feature to have the largest variance, and the less strong features to have lower variance.

The PCA calculation process is not examine-able.

- Standardize the feature values
- Calculate covariance matrix of dataset
- Calculate eigenvectors and eigenvalues of the covariance matrix, each eigenvector represents a principal component, and their eigenvalue indicates strength of the component (this is just diagonalizing covariance matrix)
- Select number of PC by ordering by the largest eigenvalues and their eigenvectors
- Multiply the data matrix with the new eigenvector matrix to get the data points in the PC space

This is a bit different to SVD, but the idea is close enough.

PCA are helpful because

- It can eliminate irrelevant features and reduce noise: fewer features means smaller models, so faster and more accurate answers
- Avoid curse of dimensionality
- Easier visualization of data
- Independent PC features

Their limitations are

- Unclear on how many dimensions to consider
- Non-intuitive PC
- Loss of information

- Only assumes linear combinations of features
- New features may not help with classification problems
- Sensitive to scale and outliers, so standardize them

Other techniques for dimensionality reduction is: CCA, LDA, not examinable.

6 Correlation

Correlation is a measure that is used to detect pairs of related variables. It may involve the direction and strength of the relationship.

We can identify simple correlation by graphing the features on a scatterplot and visually check for relationships.

Correlations strength measures the degree at which the two features are correlated. Strong correlation has data points generally in a tighter trend, weak correlation has data points more spread out around the trend.

Linear correlations is a linear relationship. Non-linear correlation/relationships can also happen.

The benefits of correlated variables

- Gives a greater understanding of data
- Hint at potential casual relationships
- Business decisions can be based on correlation/relationships, use one feature to predict the other
- Feature ranking to build better predicative models, we rank features based on their correlation with response/outcome we want to predict, higher correlated features are better features

Correlation does not imply causation, this is due to confounding variables.

A rank correlation is a correlation on ranking features (the features are rankings). We can also do correlation on time series data. We can cluster strongly correlated fields into a network, to understand behaviors of groups of fields.

Correlation can be affected by outliers, both in strength and trend. We may need to perform separate analysis with and without outliers.

6.1 Pearson's Correlation Coefficient

There are various types of correlation: positive, negative, null, linear, exponential, u-shaped. They can also be strong or weak.

Euclidean distance between features doesn't work as a measurement of correlation because different features have different measure scales, the output is in the wrong unit and does not grant an intuitive/clear answer to the degree of correlation. Namely

- It ignores shifts and scales, cannot discover variables with similar behaviors at different scales
- It ignores direction of correlation, cannot discover negatively correlated variables

Pearson's correlation can detect positive/negative linear correlations between numeric features. It defines a correlation measure $r_{x,y}$ from a sample that assess how close their scatter plot is to a straight line.

The properties of $r_{x,y}$ is

- Range of $[-1, 1]$
- 1 for perfect positive, -1 for perfect negative, 0 for none
- $|r|$ for strength of correlation
- Sensitive to outliers
- Can only detect linear relationships, and not non-linear relationships
- Scale invariant, multiplying one feature's value by a constant doesn't change correlation
- Location invariant, adding a constant to one feature's value doesn't change correlation

The formula is

$$r = \frac{Cov(X, Y)}{\sqrt{V[X]V[Y]}}$$

Interpretation of the strength of pearson correlation. In general, this depends on the domain of the data, but usually: 0.5 is large, 0.3-0.5 is moderate, 0.1-0.3 is small, under 0.1 is trivial.

6.2 Mutual Information

Pearson's correlation doesn't work well with non-linear correlations. Instead, we use the metric of entropy, conditional entropy, and mutual information to capture these non-linear correlations.

6.3 Entropy

Entropy is a measure that quantifies the uncertainty of a random variable. The more uncertain of the outcomes (more uniform the probabilities are), the less predictable the rv, the higher the entropy and the higher the information we receive upon observing a realization.

We will use the binary logarithm for all entropy formulas.

Given a feature rv X, assuming that X has k categorical bins (so k possible values), then the entropy of the feature is

$$H(X) = -\sum_{i=1}^{k} p_i \log p_i$$

where p_i is the probability of the category i. In the case of a sample of features, let p_i be the sample proportions of the category i.

Entropy properties

- $H(X) \geq 0$, it is always non-negative
- Entropy is maximized for a uniform distribution (because maximum uncertainty)
- Entropy measures the uncertainty of outcome in bits. This is also the amount of information gained with knowing an outcome.

If the features are continuous, we must discretize the into bins before computing the entropy. Some techniques for binning are

- Equal-width bins, divide the range of the feature into equal width intervals
- Equal-frequency bins, divide range of the feature into equal frequency intervals where each bin has roughly the same number of objects. We can also use quantiles to determine the cutoffs
- Domain knowledge, assigning the thresholds manually

This is similar to the binning of histograms. We then assign a categorical group for each bin, and compute the entropy using those bins.

Conditional entropy of Y given X is $H(Y|X)$, and it represents the entropy of Y given that we know X. It is defined by

$$H(Y|X) = \sum_{x \in X} p(x) H(Y|X = x) = \sum_{x \in X} p(x) \left(-\sum_{y \in Y} p(y|x) \log p(y|x) \right)$$

where $H(Y|X = x)$ is the conditional entropy given $X = x$, which is the entropy of Y using only feature values where $X = x$. The conditional entropy represents how knowing another

feature can impact the uncertainty of a feature, and is a probability weighted average of the conditional entropy given each feature of the other feature.

Define mutual information, another measure of correlation, as

$$MI(X,Y) = H(Y) - H(Y|X) = H(X) - H(X|Y)$$

which represents the amount of shared information between X and Y. Its properties are

- If MI is large, the two features are more correlated and more dependent, and vice versa.
- $0 \leq MI(X,Y) < \infty$, where zero MI indicates unrelated features
- $0 \leq MI(X,Y) \leq \min(H(X), H(Y))$
- It is symmetrical, also describes the information on X given that we know Y
- $0 \leq H(X|Y) \leq H(X)$ because knowing Y cannot increase entropy of X
- Zero MI implies the features are independent

This definition is intuitive for $H(Y)$ is the information needed to describe Y, while $H(Y|X)$ is the information needed to describe Y given X, hence their difference must be the information on Y that observing X gives, ie, their shared information.

To normalize mutual information to an interval $[0, 1]$, we can define the normalized mutual information by

$$NMI(X,Y) = MI(X,Y)/\min(H(X), H(Y))$$

we can also divide by max or mean between $H(X)$ and $H(Y)$. If NMI is large and close to 1, the features are highly correlated and more dependent; if NMI is small and close to 0, the features are less correlated and less dependent.

We cannot use MI to compare two different correlations because it depends on the raw entropys of the features. We can use NMI to compare the degree of correlations between two different cases because they are normalized and thus comparable.

Class features are features that we are aiming to predict. We can identify good features (predictors) by ones that are highly correlated with class features.

Pros and cons of MI

- Can detect both linear and non-linear relationships unlike Pearson
- Applicable and effective for use with discrete features, unlike Pearson
- If continuous features, it must be discretized and a choice of the type and number of bins must be made

- Size of bins are not obvious, different bin choices will lead to different estimations of MI and correlation

High Pearsons correlation but low NMI usually indicates that the bins are chosen correctly (Pearsons is correct, but MI is not), or there are extreme outliers (Pearsons is incorrect, MI is correct).

7 Classification

Classification is a data analysis technique. It is the prediction of future data based on historical data, and has names like predictive modeling and predictive classification.

Examples of classification problems are

- Predicting diseases from microarray data
- Classifying/labeling animals
- Classifying defaulting borrowers

7.1 Classification Definition

When creating a classification model, we usually split the dataset into a training set and a testing set. We use the training data to create the model, while using the testing data to cross validate the model.

Formally, assuming a collection of records from the training set, with each record containing a set of attributes including one class label. Classification is to find a predictive model for the class label as a function of the other attributes

$$y = f(x_1, x_2, \dots)$$

where y is a discrete/categorical value that is the response/class attribute, and x_i are predicator attributes. The function f is a predictive model that can be a tree, a rule, or a formula. Our goal is for the model to assign previously unseen records to the most accurate class.

The entire classification framework/pipeline

- Use training set and a learning algorithm to learn a model via induction
- Apply the model to the testing set via deduction

7.2 Regression Definition

A regression model is very similar to a classification model with the same datasets (train test split) and same predictor attributes predicting a response attribute. The difference is that the class label (response, or target variable) is continuous and has a real value.

7.3 Decision Tree

The decision tree is a model for a classification problem. The tree has splitting attributes as nodes and class labels as leafs in the form of a flow chart, and with each new record, its prediction is the leaf node reached when following the flow chart. There maybe many different trees that fit the same training data.

Formally, a decision tree

- Is a flow chart like tree structure
- Internal nodes denotes a test/condition on an attribute (splitting attribute)
- Branch represents the outcome of the test
- Leaf nodes represents the class labels

7.3.1 Application of Decision Trees

To apply the tree model to a test record

- Start at the root
- Choose the path given the condition on the splitting attribute of the test record
- Repeat until reached leaf
- Prediction is the leaf label

7.3.2 Decision Tree Learning Model

There are many algorithms to learn/build a decision tree given a training set. We will focus on the common Hunt's algorithm.

Hunt's algorithm is a recursive algorithm that produces a decision tree for a subset of records. Let D_t be the set of records that reaches node t, the recursion to build a decision tree is

- If D_t contains records all in the same class, then node t is a leaf node of that class label
- If D_t is empty, then node t is a leaf node labeled by a placeholder class

- Otherwise, creating an attribute test that splits the records into smaller subsets, then recursively apply this procedure on the smaller subsets

The first call supplies the entire training set to the procedure.

We will stop the splitting when the node contains records with only a single class label; this is the base case of the recursion.

7.3.3 Decision Tree Splitting Algorithm

The main issue is to determine how to split the records. This is separated into specifying the test condition and ranking the splits.

The possible test conditions (or splitting conditions) depend on the attribute type and number of splits wanted.

- For nominal attributes, we can either do a multi-way split that uses as many partitions as the number of distinct values, or can do a binary split by partitioning values into two subsets
- For ordinal/discrete attributes, we can also use a multi-way split on distinct values. The binary split must divide values into ordered subsets (where each subset must contain a full sub-range of values with no gaps, this groups similar values together)
- For continuous attributes, we can do a multi-way split by discretizing the values into ordinal categorical values. Or we can do a binary split through a binary decision on v through the groups $(A \geq v)$ and $(A < v)$.

Note for discretization on continuous attributes, we can perform them statically by discretizing once at the beginning, or dynamically by finding equal width intervals, equal frequency intervals, or clusters for each unique node. In general for all cases, we iterate through all possible splits and finds the best cut.

The heuristic is that a good split will lead to homogeneous partitions or high purity nodes (containing records with mostly the same class label). Therefore, we perform greedy splits at each level for homogeneous and high purity partitions.

To measure node impurity in a split, we can use entropy. Entropy is treated as an impurity measure where high entropy is more impure, and low entropy is more pure. This is intuitive for high entropy corresponds to higher uncertainty and a more uniform/heterogeneous distribution, hence higher impurities, and vice versa for low entropy.

The entropy calculation on a node concerns only the records that flows to that node. The entropy measure will be maximized at $\log n$, where n is the number of classes, when the records are uniformaly distributed along all classes; it is minimized at 0 when all records belong to one class.

To measure the goodness of a split, we compare the impurity/entropy of the parent node against the weighted average impurities/entropy of all the child nodes after the split. The gain in information is

$$\begin{aligned} gain &= H(parent) - H(parent|child) \\ &= H(v_i) - \sum_{j=1}^{k} p_j H(v_j) \end{aligned}$$

where v_i denotes the node i, k is the number of children, and p_j is the proportion of records in the child j partition. The larger the gain for the split, the better the split.

Notice that the gain in information (or split utility) is equivalent to the mutual information between the class feature and the feature groups being split on. So, splitting using minimized entropy and maximized information gain is to choose the feature with the highest mutual information against the class variable. This happens greedily at every recursive call until the node consists of only one class label.

7.4 KNN

Nearest neighbor classifier requires

- A set of stored record (training set)
- Distance metric between records
- The hyperparameter K, the number of nearest neighors to retrieve

and classifies a new record by

- Compute distance to all other training records
- Identify the K nearest neighbors
- Use the class labels of the K nearest neighbors to determine the class label of the new record, by taking the weighted vote or the majority label

We define the K nearest neighbors for a record as the training records that are the K smallest distance to it.

Our distance metric can be

- Euclidean distance
- Pearson's coefficients assuming paired features
- Any other distance metric

To determine the class label from the nearest neighbor, we can

- Use the majority vote
- Weight each vote by the distance, usually the weight is $w = 1/d^2$, then take the label with the highest weighted vote

A Voronoi diagram defines the classification boundary for a KNN classification where $K = 1$, it specifies what labels the new point will be assigned to given its location in space.

When choosing K

- If K is too small, we are sensitive to noisy points (outliers from another group)
- If K is too large, the neighbors may include points from other classes

8 Data Science Research

The key steps in data science research is

- Question
- EDA
- Formal modeling
- Interpretation
- Communication

For each step, we perform the following in loops: set expectations, collect information, revise expectations.

The types of research question are

- Descriptive, summarize the characteristics of a set of data
- Exploratory, identify patterns, trends, relationship between variables
- Inferential, state hypothesis as a question and perform analysis on a different dataset
- Predictive, identify factors that can predict an outcome
- Causal, find factors where changing it results in changes in another factor
- Mechanistic, how does the change in one factor change another

Good research questions are

- Interesting, someone cares about the result
- Not already answered, check existing scientific literature

- Plausible, there exists a believable framework that could lead to the predicted outcome
- Answerable, data exists can could be ethically and feasibly collected
- Specific, measurable or investigatable
- Clear and simple

Before starting the analysis, think about the data to see if they can actually lead to a result. For instance, cannot use observational study to deduce causal relationship. Consider confounding factors, biases, and limitations of the data.

The EDA steps

- Formulate question
- Read data, Check packaging
- Look at the range of the data
- Validate with an external data source
- Plot and visualize
- Try the easy solution/analysis first
- Follow up with more advanced techniques

We should consider: preprocessing to correct data or combine data, EDA to quickly gain an understanding of the dataset, modelling in which techniques are applicable and which algorithms should be used.

Key concepts in communication is

- Audience, select the right audience for the feedback you are looking for
- Content, be focused and concise, while providing sufficient information
- Style, avoid jargon unless needed, use languages and figures that are easily understandable
- Attitude, be collaborative and not defensive

9 Linear Regression

Classification is predicting a categorical variable; regression is predicting a continuous variable.

Formally, a regression model finds a predictive model from a training set by fitting the equation

$$y = f(x_1, \ldots, x_n)$$

where y is a continuous target variable and x_i are predictor variables. Statistically, the target variable (response variable) is the conditional mean of $Y(x)$.

Key terms in regression analysis

- Dependent variable is the variable we wish to predict/explain
- Independent variables are variables used to predict/explain the dependent variable

The key idea of regression is to predict the values of dependent variables based on the values of independent variables, or to explain how changing independent variables is likely to impact the dependent variables.

This subject only focuses on a high level concept: what are the inputs, what is the predicted output, how to measure prediction performance?

9.1 Simple Linear Regression

A simple linear regression model has only one independent variable, and one dependent variable. The relationship between X and Y is a linear function, and changes in Y are assumed to be caused by changes in X (causation). Like correlation coefficient, SLR can only detect linear relationship and can deduce strength of the relationship.

The model equation is

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

where β_0 is the intercept, β_1 is the slope and ϵ_i is the error/noise term. Note that this equation is an estimate of the true population regression, and this particular form is for predicting the only sample Y_i without interpolation/extrapolation.

We deduce the slope and intercept by minimizing the sum of squared differences between Y_i and $\hat{Y}_i$ (observed value and predicted value).

$$\min \sum_i R_i^2 = \min \sum_i (Y_i - \hat{Y}_i)^2$$

The β_0 term is the estimated mean of Y when $x = 0$. The β_1 term is the estimated change in the mean Y value as a result of a unit change in X. We can also interpret the intercept β_0 as the portion of values of Y that is not explained by X (although we should use the shifted intercept in statistics instead).

We can predict the mean Y for a new record with feature X by using the fitted equation.

The guidelines for regression models on prediction is to only predict within the relevant range of data (namely the range of X of the sample). Predictions in the range are called interpolation, predictions outside the range are called extrapolation.

9.2 Multiple Regression

Multiple regression is an extension to SLR. It is the prediction of a variable based on two or more independent variables.

Multiple regression (2 independent variables) is based on fitting a plane as close as possible to the observed coordinates on a 3D graph. In general, the high dimensional shape is chosen so that the sum of the squared distances to the observed data points are minimized.

10 Supervised Machine Learning, Experimental Design

Experimental design refers to the various ways we can evaluate the model that is fitted (but in general, it refers to the design of the entire experiment where we are finding relationships in the dataset).

Supervised ML contains labeled records. It is used to predict labels on new samples. Unsupervised ML contains unlabeled records. It is used to cluster related instances. Supervised ML often has performance metrics, while unsupervised ML has no scores/metrics.

10.1 Evaluation Methods

The generalization challenge for regression is about whether the model generalizes for new, unseen data points. A model that can't generalize well is overfit, which is usually when it learnt too much from the training data. It will have low bias error on the training data (predictions within one dataset is good), but high variance error between fitted models of different training sets (predictions change significantly with different training sets).

10.1.1 Holdout Method

The first method of evaluating the model on unseen data is the train-test split (or the holdout method). We train the model on the training set, and evaluate the model on a different testing set. We must assume that both sample sets come from the same distribution, and that these samples are independently drawn from this distribution.

If there is only one set of data, split about 70% into the training set and the rest into the testing set. We must split by random sampling, for there might be patterns in consecutive records.

The aim of this split is that the accuracy on the training set is irrelevant, because the model knows the dataset. The true metric accuracy is the performance of the model on a different testing set.

The entire process (depending on the problem) is usually

1. Pick an evaluation/performance metric comparing prediction against label
2. Randomly create an independent, labelled testing set, and the corresponding training set
3. Train model on training set
4. Average the evaluation metric on the predictions on the testing set

10.1.2 Cross Validation

It is important that the evaluation testing set is large. If the available dataset is small, use cross validation.

The steps of cross validation are

1. Split the dataset into N equal sized partitions P_i (randomly sampled no replacement)
2. For every partition, use that as the testing set while the rest as the training set
3. Train the model on the training set and average the performance metric on the testing set
4. The performance of the model is the average of average performance on each train-test split

We generally use $N = 10$ partitions, or $N = 5$ if the dataset is really small.

In general, a model trained on a larger dataset will generalize better to unseen data than a model trained on a smaller dataset. However, a model evaluated on a larger testing set will grant a higher confidence estimate of the accuracy of the model across all data.

When using cross validation, report the model accuracy using cross validation but use the model trained on all the available samples.

10.1.3 Stratified Cross Validation

Stratified cross validation aims to stratify the groups in the partitioning process. Each random partition is therefore not made completely randomly, but rather in a way that maintains the class label distribution (each partition has the same class label distribution as the whole dataset). This ensures that for each train-test split, the distribution of class labels are the same between the training set and the testing set.

10.1.4 Leave-one-out sampling

An extreme version of cross validation, where there is a single record per partition.

The advantages are better evaluations of the model for a high variety of training/testing sets used. The disadvantages are the long time needed to compute this metric (computational cost).

10.1.5 Repeated Cross Validation

Repeated cross validation is repeating cross validation r times, each time with a different random partitions. This smooths the effect of random partition selections.

The procedure is

- Repeat r times
- Partition the data set and perform cross validation
- Average the cross validation scores over r iterations

10.1.6 Hyperparameter selection

We should not look at the test dataset when choosing hyperparameters, otherwise there is a data leakage (where testing data leaks into the training set). Instead, create a subset of the training set as a validation set, and use the validation set as a pseudo test set to select the hyperparameter.

The hyperparameter optimization using the validation set procedure is

- For each hyperparameter option, compute the score trained on the training set and tested on the validation test
- Select the highest performing hyperparameter, train on entire training set
- Report score when tested on the testing set

10.1.7 Bootstrap Sampling

Bootstrap sampling is the use of many samples of near-independent datasets (datasets as train-test splits) via sampling with replacement to measure the score of the model.

The method is (for a training set with n records)

1. Generate b datasets, each size n sampled with replacement from the n training samples. The selected records from the sample form the bootstrap sample, the out-of-bag records (non-selected records) form the testing set

2. Train the model on the bootstrap sample, test the model on the testing set
3. Report average score over b datasets

It can handle imbalanced datasets by using stratified bootstrap sampling with a biased sample to maintain class ratios.

In bootstrap sampling, the bootstrap sample contains an average of 63% of data, while the out-of-bag sample contains an average of 37% of data.

The advantages of bootstrap sampling are

- Nice statistical properties

The disadvantages of bootstrap sampling are

- Only using limited proportion of training data
- Computationally expensive

10.2 Performance Metrics

The performance metrics are various numbers that summarize how accurate a model is.

10.2.1 Classification metrics

Confusion matrix summarizes the outcomes of a classification model on a testing set. It is a contingency matrix with columns of the predicted labels and rows the actual labels. Each cell contains the number of testing records in that category.

We define true positive, false positive, true negative, false negative with respect to a particular label. This implies that we treat the target class label as positive, while all other class labels are negative.

Define the label accuracy as the proportion of correctly classified observations

$$\frac{TP + TN}{n}$$

It is the sum of the diagonal divided by the number of samples. Accuracy is however misleading if the class label distribution is imbalanced, because accuracy is skewed towards the most frequency label and ignores the less frequent label (we can net a high accuracy by always selecting the most likely label).

Define recall for a label as the effectiveness of a classifier to identify correct class labels. It is

$$\frac{TP}{TP + FN}$$

where we fix the actual class label. We maximize recall when we don't want FN and want to detect as many potential problems as possible. Recall is also known as the sensitivity.

Define precision as the proportion of true labels compared to those predicted by the classifier. It is

$$\frac{TP}{TP + FP}$$

where we fix the predicted label. We maximize precision when we don't want FP and want to detect as few false positives as possible.

Define F1 for a class label as the harmonic mean between precision and recall

$$F1 = 2 \times \frac{Precsision \times Recall}{Precision + Recall}$$

Where the higher the value the better.

For multi-class tables and classifications, where there are k labels and n total samples. Define the subset accuracy as

$$\frac{\sum_i TP_i}{n}$$

which is also just the accuracy. Define the average accuracy as

$$\frac{\sum_i TP_i + TN_i}{kn}$$

The subset accuracy is stricter than the average accuracy, in that it only focuses on true positives.

Note that when calculating these label metrics for multi-class classifiers, we use a weighted average of the individual label metric scores, weighted by the proportion of that label class in the testing set.

When interpreting confusion matrices, discuss all class label metrics and compare them with each other, focus on class labels where recall/precision are low. We can also talk about overall accuracy. To evaluate a model, compare its accuracy (subset accuracy) with the accuracy of the baseline model. The baseline model always predicts the majority class label.

10.2.2 Regression metrics

For a linear regression model, define the error/residual as the difference between the predicted values and the observed values $\hat{Y}_i$ and Y_i.

Define the SSE as

$$SSE = \sum_i (Y_i - \hat{Y}_i)^2$$

which is the sum of squaree residuals. Define the MSE as

$$MSE = \frac{\sum_i (Y_i - \hat{Y}_i)^2}{n}$$

which represents the average squared residual of the model. And define the root mean squared error as

$$RMSE = \sqrt{MSE}$$

All three metrics are that lower values are better.

Define the MAE as

$$MAE = \frac{\sum_i |Y_i - \hat{Y}_i|}{n}$$

which is the average absolute residual of the model. This may help if there are lots of outliers for squaring makes outliers more significant.

Also, all of MAE, SSE, and $RMSE$ are on the same scale as the residuals, so they are less sensitive to outliers than MSE.

10.3 Feature Selection

For feature selection, we want to evaluate the goodness of each predictor feature. We can consider each feature separately, which will only take linear time on the number of features. This is called univariate feature selection.

Define a single feature as good if it is well correlated with the class label, so it is likely to improve the performance of a prediction model.

10.3.1 Ranking-based feature selection

This ranks each feature by their predictiveness to the class label. For discrete valued features, we use the MI. For continuous valued features, we use the correlation coefficient.

The final set of features can be determined by a threshold or taking the top N features from the ranking.

The potential issue of univariate feature selections is predicting XOR. It cannot control for inter-dependence of features — where the entire set of features can predict a class, while each specific features cannot. Hence, it might remove some important features. To fix this in practice, we can use feature extraction where we construct new features out of existing ones to preserve the inter-dependence.

When we are evaluating the model using cross validation, we should perform feature selection for each training set and treat the selected features as a hyperparamter. For N-fold cross validation, we should feature select on each combination of the partitions; for bootstrap sampling, we should feature select on each bootstrap sample.

11 LLM

Evolution of AI

- Old AI, Rule based systems built by hand
- Machine Learning, rules learnt from examples
- Deep learning, neural networks learn rules and operations

The methodology for machine learning is

- World too complex to model by hand
- ML aims to copy human expert's actions
- Many families of ML algos, the winner is ANN

The typical ML method is

- Train a system using input output pairs
- Test system on unseen tests, evaluates the output and either continues training, adjust the algo, or stop training
- Apply system knowledge to real world inputs

11.1 Neural Networks

Human brain contains neurons connected by axons in a network, each neuron performs small computation on its inputs to produce outputs activating other neurons. We can change the strength of its activation by training.

Define a neuron as a small computational function that combines its input values (vector of real numbers) to create an output value (also a vector of real numbers). The defining features are: the activation function, the connections and network structure, and how to train the weightings.

The neural network architecture and procedures are

1. Encode data as numeric vectors
2. Define a neural network by: number of nodes, number of layers, connections between nodes, choose activation functions and numerical combination functions, interpretation of output nodes
3. Train nn using input output examples to adjust node function parameters
4. Inference

11.1.1 Activation Function

Activation functions maps input signals (on x-axis) to output signals (on y-axis). We can change its shape to change the strength of the output. They can be

- Linear
- Step
- Exponential
- Sigmoid

11.1.2 Network structure

The simplest nn structure is a perceptron. It is one-direction only, a feedforward score combination.

Single layer perceptrons have no hidden layers. Multi-layer perceptrons have some hidden layers. Single layer perceptrons do a majority of simple tasks but is too weak for most tasks.

Other nn structures are

- Feedforward, one direction, no loops
- Recursive, signal can loop back and reinforce itself
- Recurrent, signals can see repeating input
- Convolutional, treat small local input groups together

11.1.3 Training

The initial link weights are random, so it is wrong. Intuitively, we can train the weights by

1. Calculate the performance
2. Calculate the changes that brings the result closer to the true answer

We ignore the backpropagation and gradient descent procedure

- Start from last layer and working backwards
- If node score trending in the right way, strengthen links to it. If they are not, weaken them
- When adjusted all link weights, repeat with the next input/output pair
- Repeat until stability

11.2 Encodings and Embeddings

The nn inputs are real vectors, and the outputs are also real vectors. We can easily use a classifier to select the final label from the output using a list of potential options. But we need to encode input data as numbers.

To encode characters or images

- Define grid of numbers
- Insert the fraction of each cell filled by ink
- Use the flattened vector

The NN will then recognize and remember the average cell weightings for the character, which may be encoded as its weights. Then to check if a new input is the character, it can compare the remembered weights against the input encodings using a dot product.

To encode words

- Represent words as a bag of words, using other words that tends to appear close to it. This encodes orderings and semantic meanings of words
- Can also use tf.idf scores for the words instead of raw counts
- Can also use NN to compute word embeddings

To calculate word embeddings

- Create training set using surrounding words predicting a target word, the design matrix contains surrounding text, the response is whether the target middle word is the actual target word
- Train NN using a network, backpropagation will adjust the weights
- Use the resulting weights as the word embedding vector for that target middle word
- Result is that words with related meanings will have similar word embeddings/weights

Essentially, word embeddings are real vector representations of words or ngrams computed by NN. We can cluster them to create groups of words with similar meaning.

Word embedding tools generate word embeddings.

11.3 Language Models

LMs are first developed in the 1970s for automated speech recognition, where it is used to guess the next ngrams from ambiguous speech conversions based on their probabilities of co-occurrence.

Define a language model as a list of likely word ngrams with their relative probabilities.

LMs are later used in machine translations in the 1990s. The simplest model from IBM follows the process of

- Take source language sentence
- Convert each word to equivalent word in target language
- Apply bag of words for each ngram
- Use language model of word combinations for target language, pick most likely in-between words using bag-of-words of direct translation.

To construct a LM

1. Scrape text from web
2. Data cleaning on the text, ideally have strings with punctuation but lemmatized words
3. Move a word window of length n across text
4. Record frequency of ngrams on every window shift
5. Normalize counts of ngrams into probability, also use log if too small to get log likelihoods

In practice, the ngram LMs are large. We may need to apply smoothing/estimation methods to approximate a similarity score if we don't have the full ngrams.

11.4 Large Language Models

Large language models are LM scaled up. They typically contain a lot of world knowledge and can also be used to support MT, ASR, information retrieval, QA.

LLM can be useful and powerful in

- Exams and tests
- Help writing, reduce writing time, improve quality, reduce writing inequality between school grades

LLM experiments can be misleading. People treat them as smarter than they are.

- It succeeds in predicting hypernyms (similar meaning words). But it did not show general and concept understandings
- It can interpret numbers, but is not perfect especially on out of range samples

- It can do basic arithmetic, but cannot scale for larger numbers.
- For a question answer problem (use passage to answer question), it shows good results. But when tested on just the question or just the passage, and the missing part filled with random words, they use pre-trained facts to answer the question showing no understanding of the passage

This is likely because it memorized the training set so it can only correctly predict results for in the sample inputs.

We can test/show this by altering the problems it has been trained on, so that it is forced to extrapolate and get the wrong answer.

In summary

- LLMs memorize sequences like ngrams
- LLMs generalizes very little, unless we train them on the particular topic
- LLMs looks for shortcuts, using associations from the training data instead of understanding the problem

LLMs cannot do inference on most problems.

11.5 Generative LLM

LLMs are passive, you have to provide special format input for them to extract output.

OpenAI developed a method for generative LLM, with simple English input and simple English output. This makes LLMs behave like chatbots.

Overview of a GenLLM

- Build a transformer like an LLM
- Add prompt as input format (use microfeatures to feed into transformer)
- Add generation capacity for output (chaining generator loop)

The basic transformer is

- A neural network where inputs are past words and outputs are probabilities of predicted next words
- A simple feedforward NN (transformer), where each neuron has a combination function with weights and activation function
- Use past data to train these weights through backpropagation
- Activation function not linear, needed to strengthen some features and weaken others

The reason that this transformer works is that the NN will extract microfeatures from the surrounding text. Hence, we can imagine that the LLM will maintain a dictionary of words, each associated with microfeatures of words surround that word. Note that microfeatures are not real features

- They are ngram patterns on ngrams patterns
- They are encoded in the NN layers
- Impossible to extract these microfeatures by looking at the neuron

All it can do is: given words, return matching words.

What ChatGPT did is

- Trained on large English text
- Took time and computing resource
- Added chat loop
- Hired humans for reinforcement learning

To generate text from GenLLM, use an output generator loop on an LLM

1. Input initial prompt
2. Use prompt microfeatures to select possible response ngram using LLM
3. Pick a matching ngram by probability, the temperature setting controls how likely it will choose the most probable ngram (lower temperature implies more likely to pick most probable one, vice versa)
4. Output it
5. Add output to prompt and repeat
6. Continue until a threshold

This generates ngrams left to right.

Some points about GenLLM

- Pretty hard to convert from ngrams LLM to valid grammar
- Variation is hard, we need to add randomness by temperature
- Deciding whether to stop; how to structure, format; and other factors

Perils, problems of left-to-right generation

- Near-distance falseness: hallucination

- Long-distance incoherence
- Socially unacceptable output

Near-distance falseness refers to the fact that the model has no understanding of truth, so it is happy to report something it has read that might be false. It can also connect two unrelated ngrams to form a sentence that is separately true but not true together. This lack of understanding also means that you can steer the ngram generated by changing the prompt.

Long-distance incoherence refers to the fact that it has small context window, so if you let it continuously generate data, it will eventually get off point and generate random, incoherence data due to low context.

Unacceptable output refers to that the GenLLM has no understanding of social rules and norms. So if the system is trained on ugly content, it may also output those bad outputs. Because we can't understand what microfeatures the LLM nodes encode, we can't simply find the bad nodes and remove it. We would either have to deweight the bad content and re-train the system (or to filter the output generated). To deweight, we can use reinforcement learning where humans read the generated output and vote on the goodness.

GenLLM is both popular and attractive. Research on new capabilities, frameworks, specialized hardware for GenLLM. Universities and companies are experimenting on how to use it. Governments are trying to understand the threats and how to regulate it.

11.6 Recent GenLLM developments

GenLLM is fast moving, with new capabilities being developed

- Smaller and free LLM
- Private LLM, dont share private data
- RAG (retrieve augmentative generation), adding own content without re-training
- Agency, adding functionality not native to LLM (interface through other apps)
- Explanation, explain why the LLM did what it did
- Chips and specialized hardware

RAG adds our own material into the LLM by maintaining a private database for custom knowledge. It then uses a local search engine to find the relevant information and inject that as a prompt before the user prompt to simulate custom content without needing to re-train the weights.

Plugins allows LLM do things that is not just text generation. It matches specific keywords that interface with plugins, and the LLM will invoke that plugin which can invisibly run

bookings/searching. Agency extends this to allow LLM act in the real world. It is an interface that connects LLM output to real world system software, then take the service response output and feed that back into the LLM. However, this has high potential for error, requires access control and user validation.

To get an explanation out of the LLM, we can use hotspot analysis or LLM over LLM.

11.7 Image and Video generation

Similar to text LLMs, there are Generative Vision for images and videos. This is because text and image processing use similar NN and embedding representation.

Image generation process

1. Train NN on images and not text, using image microfeatures associated with input words
2. Start with a random field of colored dots
3. Find matching microfeatures from input text
4. Match them into random dot field, unify them to create stronger microfeature regions on the image
5. Repeat, growing the local information of the image by merging in microfeatures for an increasingly specific image

But NN microfeatures are not the same features that people see, so it can make mistakes when classifying images. We can fix this by doing adversarial ML, where we generate image from NN that looks similar to NN but not to a human. This allows us to reward microfeatures that humans see.

Software generation LLM trains on software programs. The benefits are that code input data has: smaller vocab, more restrict combinations/syntax, easier to generate code than a story. It has the same problems of hallucination and incoherence.

11.8 General Shortcomings of LLMs

LLMs are just a big database of text fragments (ngrams).

It has no understanding of truth. May report incorrect things, may assemble two separately true statements into a single untrue statement.

It cannot reason, cannot infer, be consistency, or be logical. The logic it mimics comes from the long context window.

It has no goals or wishes.

Vulnerabilities of LLMs

- Training data poisoning, including malicious information into the training set
- Insecure plugins
- Prompt injection, users injecting prompts that overwrites LLM prompt
- Overreliance, use of output without verification
- Sensitive information disclosure, releasing ip protected information, including model details
- Excessive agency
- Model DOS
- Supply chain vulnerability

The 4 major problems for LLMs

- Privacy (security problem), use of public LLMs may leak private information
- Accuracy (truth problem), hallucinations
- Understandability (explanation problem), a black box NN
- Bias (ethics problems), no understanding of social norms

11.9 Commercial Uses

We use can LLMs for

- Writing code to data preprocess
- Produce training data
- Detect anomalies in datasets
- Identify patterns

The main challenge is writing the prompt. The prompt should describe the overall goal, the specific task, example outputs and examples of what not to output.

The two main uses of them are to improve products or to automate processing. We see that within each industry, only 1-2 use cases have high penetration rate, with a sharp falloff after, indicate a focused area of investment.

11.10 Other uses

Fake news in images and text. Image fakes may be more dangerous for they are more believable.

Job loss due to automation. These long term threats may be social, and not technical.

What we can do are to learn how to use them, and discover what they are good at and not good at.

12 Recommender Systems

Recommender system provides a list of personalized items to the end-user. We need recommender systems because of

- Information is becoming abundant instead of scarce
- Internet changing shopping behaviors
- Some online systems like social media is largely dependent on recommender systems
- People often don't know what they want until you show them

Interestingly, the distribution of items sorted by popularity is right skewed.

Recommender systems are used by most online platforms.

Facts about recommender systems

- A majority of what people watch is from recommendations
- Ideally we show provide personalized stores for each user
- Recommender systems should provide the best relevant items that reduces search times and frustration

Mechanisms of recommender systems

- An online system where users interact with items
- Each user has their unique profile
- User rate items: explicitly through a score, or implicitly through web usages
- System use information to generate recommendations

Issues and challenges in recommendation systems

- Interpretability of recommendations
- Fairness to rare items

- Avoid recommending popular items
- Handle new users and items

To measure the performance of a system, we can use the holdup method and calculate RMSE between predicted and actual ratings using the predictions from the training set on the testing set.

12.1 Popularity Based Recommendations

Simply show the popular items. Needs to define popularity metric. This is very simple by is not personalized.

12.2 Collaborative Filtering

Collaborative filtering is the idea of making predictions about a user's missing data (missing ratings) using the collective behaviors of many other users.

It looks at the users' collective behaviors towards items, and looks at the current user's behavior history, then combine the two to make predictions.

There are two methods: item-based collaborative filtering, and user-based collaborative filtering.

The framework around collaborative filtering assumes a set of m users U and a set of n items I. It then has a pre-existing (not-complete) $m \times n$ interaction/rating matrix R between users and item with the ratings. The purpose of collaborative filtering is then to find the ratings of missing cells in the matrix.

12.2.1 Item-based

Main idea is that people like items similar to other items they like. To do this, we need to

- Search for similarity between items. Assume that items are similar if users collectively like/dislike both similarly: define similarity as collective similarity in ratings between many users.
- Recommend items similar to those that are rated (and likely) by the user

To measure item similarity, we can use a metric between the item ratings. A simple version is using Euclidean distance with mean imputations of missing values. Define the Euclidean similarity score as

$$\text{sim}(a,b) = \frac{1}{1+d(a,b)} \quad d(a,b) = \sqrt{\sum_k (r_{ak} - r_{bk})^2}$$

To find similar items for a specific item, we first compute the similarities between it and every other item. Then given the user ratings on other items, choose a number k, and find the k most similar items in the set of items that the user rated using the similarities metric. Finally, computed a similarity weighted average of the user's ratings for the k items, and use that as the predicted item rating for the specific item.

For similarities s_i and ratings r_i for the set I of the k most similar items, the prediction is

$$\frac{\sum_i s_i r_i}{\sum_i s_i}$$

In summary, the algo is

1. Batched and offline: compute similarities between pairwise items using a metric
2. Online: predict rating of a specific item for a user based on the k most similar items rated by them, prediction using weighted average ratings

Note that because similarities can be pre-computed offline, the algorithm is efficient at runtime (when suggesting recommendations). Suitable for applications where the number of users are much greater than the number of items (for it is quadratic only on the number of items).

The cold-start problem is that new items/users won't have any ratings and thus no similarities. A solution is to show these new items to a portion of users to get some data.

12.2.2 User-based

Main idea is that people like things liked by people with similar taste.

We need to find similar users based on the similarities of their ratings to items, then recommend items like by similar users. This is mathematically similar to item-based methods.

To measure similarity between users, use the Euclidean distance with mean imputation between the user's ratings to items. This is done offline.

To predict the user rating for an item, choose k most similar users whose rated the item, and take the similarity weighted ratings of the items as the predicted ratings. This is done online.

Problems

- Item-based often performs better in practical cases
- User preferences are dynamic, so high update frequency of offline-calculated user similarities, compared to the relative static item-based item similarities

- Sparsity problem in the lack of rated items when computing similar users
- No recommendations for new users
- Hard to scale when the number of users increases, due to more expensive similarity calculations

To scale up the search for similar users, we can do an offline step of clustering users. Then we can find similar users by considering only users within the same cluster.

12.2.3 Similarity metrics

The three most popular similarity metrics are

- Euclidean distance and similarity
- Cosine similarity
- Pearson correlation

12.3 Content based recommender systems

An alternative to collaborative filtering systems. Content-based systems use pre-determined features of an item to recommend similar items. This can overcome the cold start problem.

Realistically, a recommender system will use a combination of content-based and collaborative filtering methods.

The process is

1. Create feature vectors on an item's attributes that contain the rich features of the item
2. Create feature vectors of users using the weighted average of the feature vectors of rated items
3. Recommend items using similarlies on user/item feature vectors.

Problems of content-based systems

- Relies on properties of items, requiring manual selection of attributes
- Transparency in selected properties is hard
- Independent from other uses' ratings. Avoids the new item problem, but can recommend unpopular items
- Over specification, often recommends items too similar to those already seen by the users

- New-user problem still remains

12.4 Matrix based techniques (NOT EXAMINABLE)

Simultaneously fill in the missing ratings using both similar users and items.

The technique considers the rating table R as a matrix. It performs matrix factorization on the table and fills in the missing cells.

Formally, consider the factorization on the incomplete sparse matrix R

$$R \approx UV$$

where U is $m \times k$ and V is $k \times n$. The dimension k is the number of latent factors. The error of the factorization is the sum of squared residuals of the non-empty cells between R and UV.

Commercial recommender systems often used variations of the matrix factorization method.

12.5 Summary of recommender systems

Popularity based has no personalization.

Collaborative filtering has personalization. Item-item filtering is more efficient. Both have cold-start problems for new items and new users.

Content based recommendations uses feature vectors from item attributes. Has no new-item issue, but has new user issue.

13 Oral Presentation (NOT EXAMINABLE)

Key elements: structures, body language, voice, visuals, timing, audience.

Audience. The audience is interested, focus on the interested person. Signposting is explicitly saying what you are about to say. This can be used to introduce new sections or to shift a tone from descriptive to interpretive.

Voice. Word stresses, tones, pacing, etc. Avoid memorizing the talk to avoid a monotonic delivery or to get lost when problems occur. Aim for 140 words for minute ish.

Combining body language and functional language (speaking) for a better message delivery. If live, move during pauses, but stay still when talking.

Visual. Use 18-36 point, high contrast font. Remove distracting details. Use simple animations to introduce concepts step by step. For images, use simple graphics or highlight important sections.

Nerves. Pre-presentation, know the material, practice and listen to feedback. Positive thinking, and avoid stimulants. In presentation, focus on positive audience members, keep going for delivery mistakes.

14 Ethics, Privacy, and IP

Information has value, and it can be stolen or sold, which can be harmful to the individual.

Intellectual property protects your data from being used by others. If you've spent a lot of time or money to create information, it would be stealing if someone just took them or re-sold them. This is an important part of data processing.

Privacy is about keeping your data safe from bad actors or other people who you did not consent to. The problems of sharing your data are that

- Identity theft
- Detectability, in that we can deduce the sensitive information from someone
- The sensitive information can lead to discrimination, blackmailing, or advertizing.

Bias or ethics is about considering the potential effects of your conclusions or models. If the model predicts classes that are sensitive features of society, or that the data itself is imbalanced or biased, we can end up creating a result that can affect others negatively without proof.

14.1 IP

All of IP, privacy, and bias became a problem due to the internet.

People started sharing data in: text, images, structured data, analysis, and other personal data through social media. Data collectors then can gather bits of information, do analytics, and sell insights to advertiziers, companies, and others. But these small bits of information can be aggregated and analyzed to be valuable, so who owns the data?

Intellectual property is a category of property that includes intangible creations of the human intellect. They can be

- Patents, the right granted by the government to an inventor giving to right to exclude others from making, using, and selling the invention for a limited period of time. However, there must be public disclosure of the invention.
- Copyright, preventing an original work to be copied or used by all except the creator who has exclusive rights to it, for a limited time.

- Trademark, essentially a copyright for recognizable signs or logos. It is the ownership of a design, expression, signs, or logos that distinguishes a product or service from similar products or services by others.
- Trade secret, it is a formula, practice, process, design, instrument, pattern, or information which is not generally known or easily discoverable that the business can use to obtain an economical advantage over its competitors. They are protected from other businesses.
- Licenses are rules that allow others to use your data

14.1.1 Licenses

Our responsibility when data processing through crawling is to understand who owns the data and how others can use them. During all steps, we should make sure that all legal and ethical obligations are met. The rules and terms around using data are summarized in licenses.

The two types of licenses are: Open Data Commons License, Creative Commons License.

ODC contain licenses targeting data in databases.

The various ODC licenses are

- Public Domain Dedication and License (PDDL), dedicates the database and its contents to the public domain for everyone to use
- Attribution License (ODC-By), users are free to use the database and its content, provided that they give attribution to the source
- Open database license (ODC-ODbl), requires subsequent use of database to provide attribution, making an unrestricted version of the new product to be accessible, and any new products using the data must be distributed under the same terms. This is a copy-left license

CC licenses are relevant to data in general.

They are

- CC0, owners waive all copyrights on the data or database, no requirements, similar to ODC PDDL
- Attribution (cc-by), users must credit the sources
- Attribution-ShareAlike (cc-by-sa), users must give credit and use the same license
- Attribution-noncommerical (cc-by-nc), no commericalization

- Public domain mark (PDM), for works already in the public domain with no copyright restrictions. Flagging items as PDM shows that it is public domain and free

There are no right answers to which license we should use. CC licenses are more common for they are general purpose, while ODC only covers database rights. Can work with legal advisors on the license to use.

The most open license is ODC PCCL or CC0 and it's also the easiest. Anything else may cause problems later on in attribution stacking — every modification will add a line of attribution.

14.1.2 Copyright

Copyright holds automatically as soon as you create the work, no registrations are needed.

Under certain conditions, you can use someone's work if you give attribution (unless they've given written permission). The conditions are

- Using a single photograph or illustration
- No more than 5 images by a single artist
- Up to 10% and less than 15 images from a single publication
- A single figure from a book or magazine

If you repeat more than N words of someone's text, you must quote and cite them (unless they've given permission). The limits are

- up to 10% and less than 1000 words of a single written work
- up to 250 words of a poem
- no more than 3 poems from one poet, or in the single book
- up to 10% and less than 2500 cells from a database

14.1.3 Regulation

It is hard to define if a new work is new, or just copyright infringement.

- Generative AI trained on images can combine image fragments into new images, but does this violate copyright? Using direct images must require credit or pay, but what about the combination of pieces?
- Text Gen AI builders claim that their text is an original and novel combination of words based on common usage patterns, so they don't quote it. But if you reuse someone's text, you must quote it

The EU general data protection regulation (GDPR) is established to protect EU citizens from their data being used for data analytics for social, commercial, or political purposes. With exceptions of national security or legal proceedings.

The GDPR has some provisions

- Data subject must be informed
- Data subject must be given access to their data
- Data subject can ask for data erasure or removal
- Government organizations needs a data protection officer

The GDPR's impact are that: other countries adopted their laws, and caused lots of extra work for big tech data scrapers (because most are multinational corps that want to keep EU consumers, and it is cheaper to uniformly apply the changes).

Summary of ways that data can be restricted

- Copyrights, statements claiming to forbid re-use of data, no derivatives
- Patents, forbidding re-use of data for profit
- Payment, available for a charge
- Membership, only registered members have access
- Encryption, use of encryption that creates a barrier for access
- Access restrictions, robots.txt
- Time limited access, limited online but unlimited physical copies
- Access throttling, provides only single item downloads
- Aggregations, summary (or a bit) is available but the collection is not.

14.2 Privacy

Cookies collects user's data. Websites ask for cookies due to GDPR's requirement to ask for consent.

One database may preserve privacy, and another might also preserve privacy. But when combined, they may give enough information to violate privacy. Because you never know what some future databases will provide, you can never be sure that your privacy measures will be enough.

To preserve privacy, we have two options

- Releasing some data, with a guarantee of privacy in all cases
- Create a flexible degree of privacy tailored for each request

14.2.1 Public data release, individual anonymity

In most cases, removing explicit identifiers from a dataset is not enough.

Terminologies

- Explicit identifiers, unique features for an individual
- Quasi identifiers, a combination of non-sensitive attributes that can be linked with external data sources to identify an individual (gender plus age plus zip code)
- Sensitive attributes, are information that people don't wish to reveal and features that we are trying to protect

Measurements and approaches of privacy and anonymity are

- K-anonymity
- I-diversity

K-anonymity aims to release some of the data with scientific guarantee that the individuals who make up the data cannot be re-identified, but keeping the dataset useful. The higher the k, the more private the data, but the more processing is needed.

We define k-anonymity on a table if every record in the table is indistinguishable from at least $k-1$ other records with respect to all sets of quasi-identifiers. In other words, for every unique combination of quasi-identifiers, there are at least k records sharing those values.

When checking the anonymity level of a table, look for the minimum sized set of unique quasi-identifiers.

To achieve k-anonymity, we can use generalization or deletions.

- Generalization (collapsing) is to make the quasi-identifiers less specific. It works on the column level generalizing all values in that column
- Deletion (suppression) is to remove some the quasi-identifier records completely. This helps to moderate and fine tune the generalization method, and limits the number of outliers. Can work on both cell, columns, or row levels.

When generalizing, we should group the similar values in a column together for it is then harder to differentiate between records with the same generalized quasi-identifiers using other features of the dataset.

K anonymity guarantees that in the worst case when the data gets into the wrong hands, they can only narrow down a quasi identifier to a group of k individuals. As a data publisher, you need to determine the quasi identifiers and choose an acceptable parameter k.

Problems with k-anonymity are the homogeneous problem and the background information problem

- k anonymity may create groups that as a group, will leak information due to a lack of diversity in the sensitive attribute. If all the sensitive attributes are equal within a unique quasi-identifier, k-anonymity is useless in protecting the privacy of the users.
- Does not protect attacks based on background knowledge between the quasi-id and sensitive attribute (filtering unlikely sensitive attributes based on quasi-id)

L-diversity solves both problems. It ensures that the sensitive attribute is diverse within all groups. It is defined by: for each k-anonymous group, there are at least L different sensitive attribute values.

Similar to K-anon, we choose the group with the least L different sensitive attribute values and report that as the L-diversity value.

Limitations of L-diversity

- Difficult to do because the data is not suitable and imbalanced. Reducing imbalances within groups may also limit usefulness
- May be unnecessary

Summary for privacy when releasing data. To reduce risk of re-id,

- choose value of k
- manipulate data to make it k-anon by replacing or suppressing
- choose value of L
- further manipulate data to make it L-diverse by ensuring diversity

14.2.2 Differential Privacy

Aims to change the degree of privacy on a query by query basis.

Can either be done through local or global methods

- Local means that individual are responsible for adding noise to their data
- Global means that a trusted data manager analyzes the raw data, adds noise, and reports the noisy answer. This is more accurate and can result in less noise

Local privacy is biased. This is because that if you ask everyone to lie (negative data surveys), we can simply inverse the distribution to get the true answer. It also is challenging in designing the false questions, and relies on the honesty of the respondents. The privacy is worse with fewer categories.

Additionally, local privacy requires knowing who else is in the data or survey. If your sensitive attribute is rare, your records will still be rare and you can be identified.

Global privacy analysis can fix all the local privacy problems. We aim to add a different noise for each unique query, with a goal of making sure that the effects of any one record on the final dataset is minimal towards the result. It consists of

- Querying data
- Considering the aggregate result
- Add random noise to hide presence of individual records
- Release noisy result

Because different queries require different records for different purposes, we need to add a variable amount of noise to preserve privacy (but too much noise makes the data useless). Noise consists of

- Changing the values
- Withholding data records

A simple method is to add a small deviation with mean zero to the aggregated result, so that on average the result is accurate, but observing a single released result doesn't give any exact knowledge.

When withholding records, we want to drop as few as possible and dropping those that are the most sensitive. Numerically, let K be the privacy loss budget, and G be the sensitivity of the data.

- Privacy loss budget K is an integer that expresses how private we want the result (the maximum information disclosed). The smaller the k, the most difficult it is to guess and the more we drop.
- Global sensitivity G is a number that expresses the difference between including the record or not on the modeling result. This is calculated by the difference in likelihood of getting a result when one or records are included or withheld.

The aim of differential privacy is that the likelihood of the noisy released result will be R does not change much upon inclusion or exclusion of a single record.

Define A as the prob of getting the result R with the record, and B without, note that A will always be greater than B for adding results increases chance of correctness.

For privacy loss budget, $k \geq 0$, we keep a record if $A \leq 2^k B$ thus guaranteeing that the ratio of results is smaller than 2^k (every increment of k doubles the upperbound likelihood of getting the correct result R when including the record). We have four cases

- Big change if removed, small budget, withhold
- Small change if removed, small budget include
- Big change if removed, big budget, include
- Small change if removed, small budget, include

Global sensitivity measures how much change we can do to the dataset when dropping information. Intuitively, we want to drop records with the most changed when removed (so what is remained has low sensitivity).

G of a query is the maximum difference in answer accuracy when adding or removing one individual, it is the summation of the global sensitivities of the subqueries.

To combine both G and k, we want to drop as little data as possible but still remain under our budget k, while dropping the most revealing data (minimize G) with minimal damage to data accuracy. In essence: drop more revealing records but fewer of them, drop less revealing records but more of them.

To use G and k to add noise in ways that are not dropping the data, we add a small noise from a random probability distribution with mean 0 and standard deviation G/k to all relevant cells. We usually sample from a laplace distribution for this noise.

When answering questions about which ones to drop, drop ones that won't remove unique records so we retain accuracy, while dropping the most identifiable records (thus reducing G), of course all under the budget.

In summary, differential privacy guarantees that the presence of a user cannot be determined after releasing the query result. However, it does not prevent people from drawing conclusions about the individuals in the aggregate results. We determine the size of the noise by deciding on the budget and looking at sensitivity.

14.3 Ethics

Ethics it the study about what is correct.

EU has laws in

- Digital service act applicable to online platforms in Europe. It protects users against manipulative algorithms that spread disinformation, by making it easy to report them.
- Digital market act identifies gatekeepers and enforces them to comply with obligations and prohibitions.
- There is an EU AI act.

US has the bills

- Gen AI copyright Disclosure Act, requires creators to submit detailed summary of copyrighted materials used in training or to alter training dataset
- Nurture originals, Foster Art, Keep Entertainment Safes (NO FAKES), protect individuals from unauthorized recreations using gen AI
- Ensuring Likeness voice and image security (ELVIS act), protect artists from the misuse of AI

Australia only have AI Ethics principles. There are only consumer, data protection, and anti-discrimination laws.

Ethical actions concern the decisions we make about our data and their impacts on others. When making a model or inference, consider

- If the classifier is always accurate, who would benefit
- If the classifier is not always accurate, who could be hurt
- Who is responsible for false predictions

A good heuristic is to consider the cost of misclassification. If getting it wrong, how much suffering could it cause (which if it happened to you?). If it is legal, it doesn't mean it is ethical.

Dual use or dual framing is about using classification algorithms outside the legitimate use cases. Also consider who should be responsible.

In summary, consider

- Who can benefit from the tech
- Who can be harmed
- Representativeness
- Privacy
- Could sharing the data directly influence people's life

- Ok if beneficial and low cost of misclassification, not ok if not too beneficial and high cost of misclassification
- Legitimate and non-legitimate uses

15 Data Linkage

Data linkage is the combination, grouping, and matching of information of different sources about the same real world entity.

Its applications include

- Matching hospital and death register data, mortality of certain diseases
- Matching hospital and job occupation data to find correlations between occupations and disease susceptibility
- Robo-debt collection, matching centrelink data and tax office data for income discrepancies. This then automatically triggered a process of a letter with no human oversight.
- Bibliographic database. Measure researcher and university productivity, counting number of publications per academic even. Need to match on the author researcher's information in different publications
- In security, to identify the person by matching their identity against trusted government sources. Useful when applying for a loan. This can identify high-risk passengers in flights (matching booking data against previous visits/visas, crime database in both countries)
- Business. Create combined database of people and their interest in a marketing collab. Checking for valid street addresses against a master database. Online shopping comparisons and matching of similar products in multiple stores
- Social media information on the same location with different names/tags.

That last data matching program from centrelink does not use any raw TFN, instead relying on matching using other quasi-identifiers, then comparing historical ATO employer reports against the centrelink records. This means that it is not subjected to the Data-Matching Program (assistance and tax) Act of 1990, and instead followed the Voluntary Guidelines on Data Matching in Australian Government. This program was appealed by an ombusdsman and won for the consumers.

In some cases, businesses will track customer information in a large database, one that is changing over time, and creating duplicated records on the same individual. They may need to know if any duplicated records are about the same person (duplicated entries

of personal information after changing departments or addresses). This can be done by matching the database against itself.

15.1 Linkage challenge

The linkage task is to align two lists of records stored in different databases (match related records across sources). The linkage problem is that: records are represented and defined differently in different databases, with potentially different features and different feature representations.

To link a database of movies against another movie database, it is only easy if there is a common attribute that is unique per movie (a key that we can join on).

The linkage challenge are in the forms of: identical names for different items, identical items with noisy values, missing attributes.

We can partially solve this by standardizing the common attribute, and combining it with another to create a truly unique common key.

In general, the algorithm has to compare the similarities of multiple features simultaneously. Then it can identify the most plausible matches even if there are partial matches (similar featured items but different).

The terminology of data linkage under different communities are

- Statistics, record linkage
- Databases, entity resolution, data integration, deduplication
- NLP, coreference resolution, named entity recognition

each with slightly different meanings and scope.

15.2 Linkage method (structured data)

The task is: given two lists of entities A and B, match those from A to their equivalents in B, and ignore the unmatched.

Some considerations are

- Different features and values
- $O(nm)$ time complexity for brute force
- No clear definition for similarity
- $O(nmk)$ if the entities have k features

The typical pipeline for data linkage is: prep, block, score, match, and merge.

For prep. Prepare the records by standardizing the format. Remove cap, fix numbers, data cleaning

For block. Need to score pairs of records for similarity, but cannot bruteforce all pairs due to quadratic complexity. Instead, decompose complex records as a series of simple values (blocks) on one or more fields. Union all the blocks across all records. The blocks can be text ngrams on a field. Assign each record to the block that they contain (each record can be linked to multiple blocks). Within each block, compare the records from A against B (in more detail later). If two records don't share a common block, they are unlikely to be equivalent.

We can have many smaller blocks or fewer larger blocks. Smaller blocks have fewer comparisons within each block but have more blocks in total. In general, smaller blocks will lead to more FN and fewer FP, and larger blocks will have more FP and fewer FN.

Empirically, we see a performance increase as the number of blocks increase and the block size decrease. The formula for the number of comparisons made is

$$n(x/n)^2$$

where n is the number of blocks and x is the number of records.

For scoring. We aim to compare records attribute by attribute in defining a similarity score between two different sets of simple values on the same attribute (remember that a record has a set of blocks). This similarity between two sets of simple values is called the block similarity (often the proportion of similar blocks). Alternatively, we can also compute field similarities without using the blocks.

Note that for strings, we can use string similarity methods like: word frequency, n-words, ngrams, prefix/suffix, soundex (assigning code to similar sounding words). Depending on the similarity method, this actually defines our blocks (for ngram similarity, we will use ngram blocks, and define block similarity as proportion of ngrams shared).

The block similarity requires us to consider: should we match only on exact set of blocks, a tolerance on the numeric field, or a tolerance on the word.

For matching, we compose individual block similarities on all features to find an overall record similarity score for the pair. We require a function with the syntax $f\colon \mathbb{R}^d \to [0,1]$. Some versions are

- Sum up the block scores
- Positive similarity and negative dissimilarity
- Weighted sum

- Label examples of nonmatching and matching record pairs, train a classifier using ML to identify pairs based on block scores.

For merging, we identify the best matches for each record by matching scores, and find an optimal partitioning. This can be done via a threshold that the matching score must exceed to match.

Summaries about the block method

- Should be efficient and capture good heuristics in the blocks for matching.
- Similar to hashing.
- The blocking heuristic should not be too general (capturing a lot of unrelated records), or too specific (capturing only the exact pairs)
- Trade-offs between block sizes are between missing true matches vs huge blocks. Some solutions are: we can manually process large blocks, can allow small blocks to overlap but avoid redundant ones for small blocks, ensure that recall doesnt suffer from false negatives if blocks are too small.

15.3 Evaluation of data linkage

The four types of linkage for evaluation are

- TP, matching results that are correctly classified
- TN, non-matching, that are correctly classified
- FP, non-matching results that are incorrectly classified as matching
- FN, matching results that are incorrectly classified as non matching

where we treat classification of matching as positive.

The precision of a record linkage result is the proportion of true positives out of all positive classifications. The recall is the proportion of true positives out of all matching pairs.