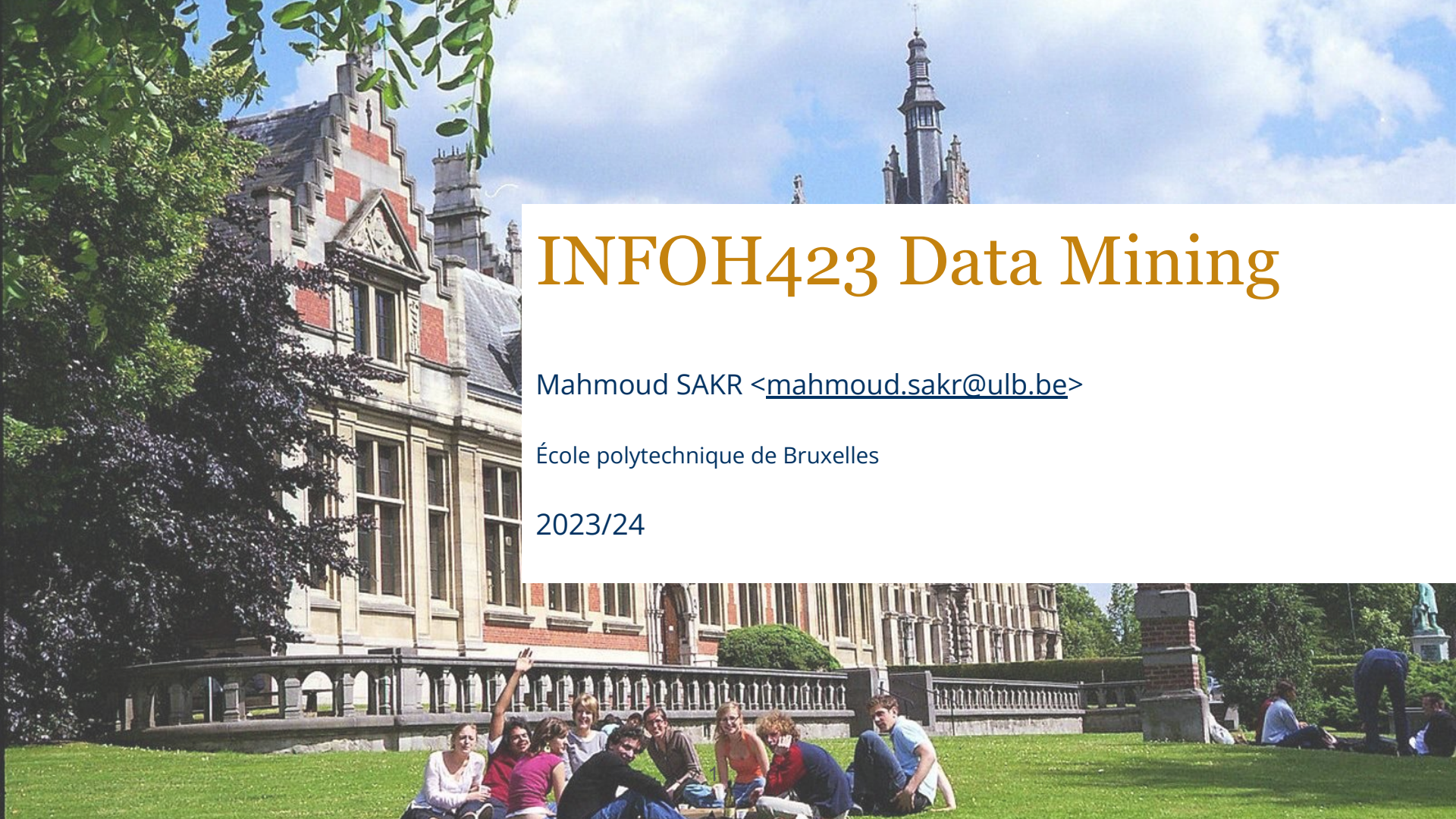


INFOH423 Data Mining

Mahmoud SAKR <mahmoud.sakr@ulb.be>

École polytechnique de Bruxelles

2023/24

What is data Mining ?

- Data mining is the study of collecting, cleaning, processing, analyzing, and gaining useful insights from data. [C.Aggarwal 2015].
- Data mining (knowledge discovery from data) is the process of nontrivial extraction of information from data, information that is implicitly present in that data, previously unknown and potentially useful for the user. [Frawley et al. Knowledge discovery in databases: an overview. 1992].

Course Goals

- To introduce the fundamental concepts and techniques of data mining
- To develop skills of using recent data mining software for solving practical problems
- To establish the main characteristics and limitations of algorithms for addressing data mining tasks
- To select the most appropriate combination of algorithms to solve a data mining problem
- To develop and execute a data mining workflow on real-life datasets, and to solve a data-driven analysis problem
- To identify promising business applications of data mining

Course Topics

- Classification.
- Model validation and data preparation
- Clustering
- Outlier mining
- Frequent pattern and association rule mining
- Stream data mining
- Applications

Prerequisites

- Good knowledge of programming
- General knowledge of Data structures, Algorithms and Complexity.
- General knowledge of Databases & SQL

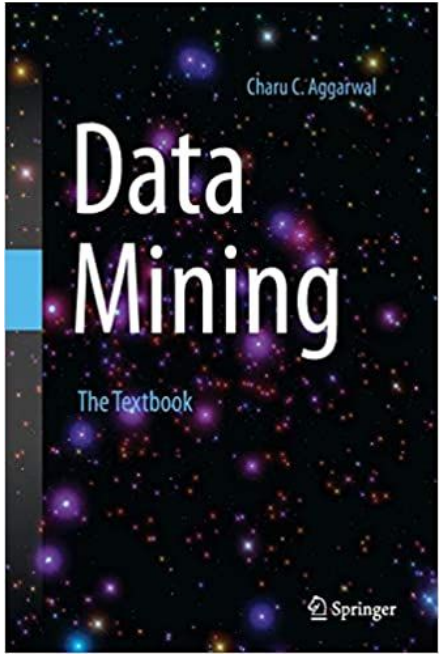
Course Organization

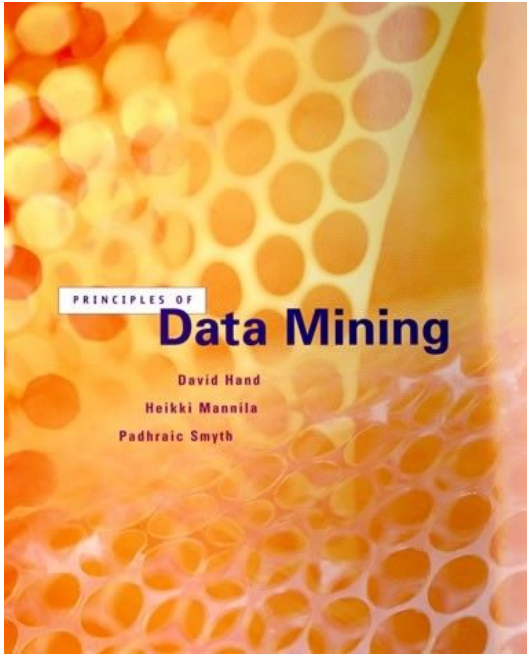
- Schedule and room on timedit: Check regularly for schedule updates
- Lectures by mahmoud.sakr@ulb.be
- Lab sessions by: Raphaël Gyori
    - Get your Rapidminer educational license
- Project
    - Practice a real world data science problem
    - Group project, 4 members
- Grading
    - Project 40%
    - Written exam 60%
- Course notes, please enroll in Université virtuelle

Skills

- Lectures cover: **Theory: Concepts, Algorithms, complexity**
- Lab sessions cover: **Rapidminer, quick prototyping, suitable for engineers**
- Project:
  - **Practice a real world data science problem**
  - **Apply the studied concepts, and further self learn**
  - **Advanced programming might be needed**

Recommended Readings

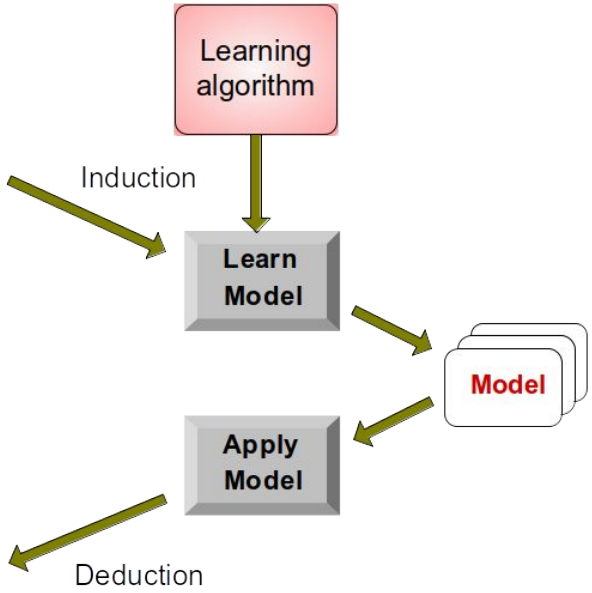




Classification

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

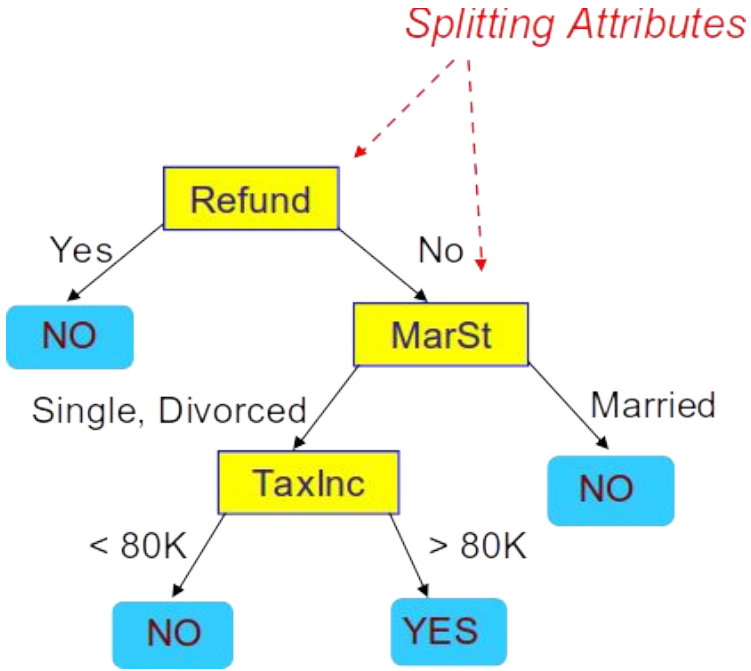
Classification

- Given a collection of records, training set, each record contains a set of attributes, one of the attributes is the class.
- Find a model for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.

Decision Tree Induction

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Cheat (class)
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

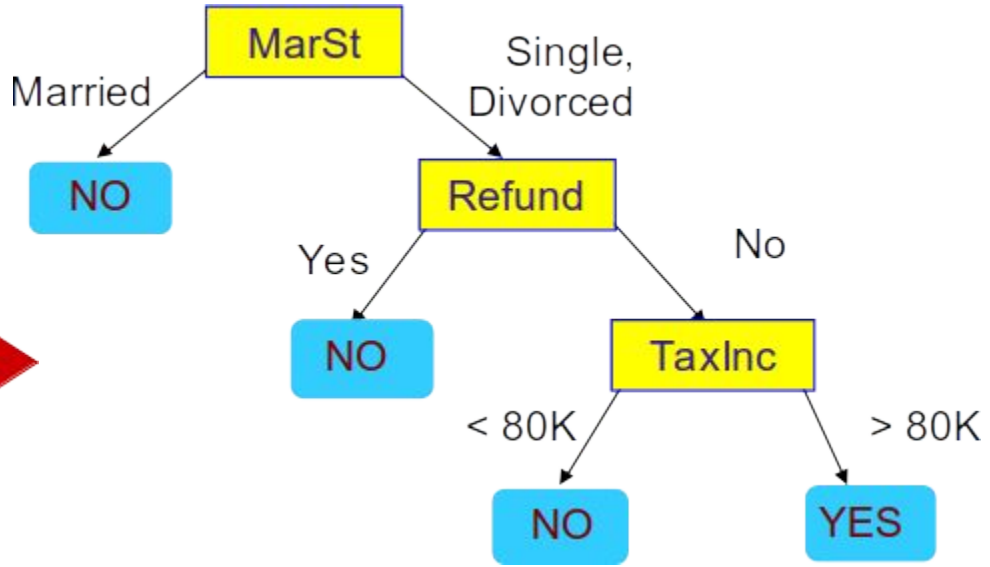
Training Data



Model: Decision tree

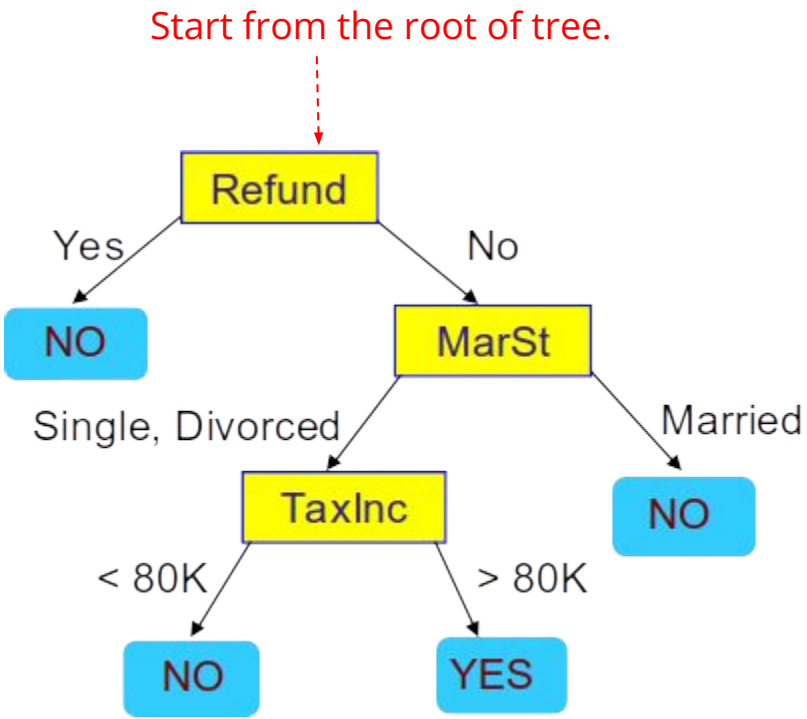
Decision Tree Induction

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Cheat (class)
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



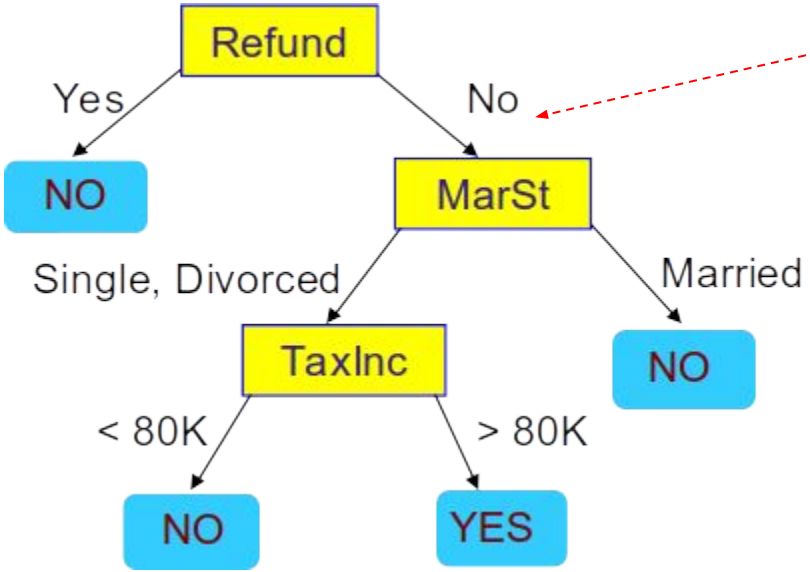
There could be more than one tree that fits the same data !

Decision Tree Prediction



Refund	Marital Status	Taxable Income	Cheat
No	Married	80 k	?

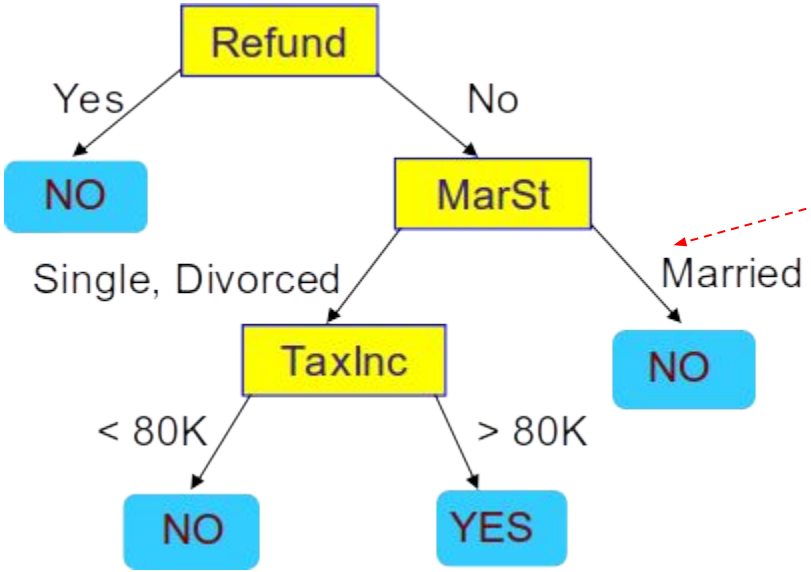
Decision Tree Prediction



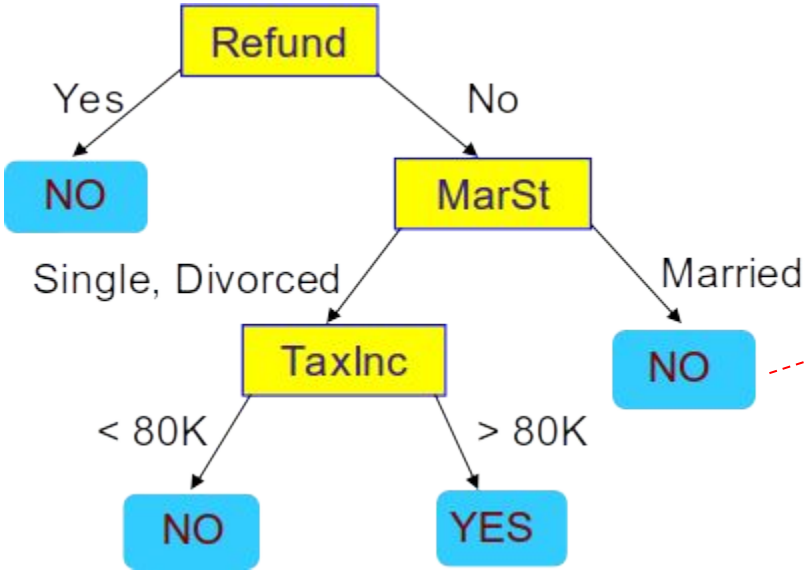
Refund	Marital Status	Taxable Income	Cheat
No	Married	80 k	?

Decision Tree Prediction

Refund	Marital Status	Taxable Income	Cheat
No	Married	80 k	?



Decision Tree Prediction



Refund	Marital Status	Taxable Income	Cheat
No	Married	80 k	?

Predict Cheat= No

Exploratory Questions

1) Data Cleaning - Remove outliers, Handle missing values
2) Feature Selection: Select relevant features: Choose features that are less susceptible to noise. Feature selection techniques can help identify and retain the most informative features.
3) Feature Engineering:
Create new features: Derive new features that might capture useful information and reduce the impact of noise.
Discretize continuous features: Convert continuous variables into discrete ones to make the decision tree more robust to noise
4) Pruning: Prune the tree: Decision trees can be pruned to remove branches that contribute to overfitting. Pruning helps simplify the model and makes it more robust to noisy data.
5) Use ensemble methods
6) Weighted Samples: Assign weights to samples: Give higher weights to cleaner samples and lower weights to noisier samples. Some decision tree algorithms allow you to assign weights to instances, influencing their importance during the training process
7) Cross-Validation: Use cross-validation: Evaluate the model using cross-validation techniques to get a more robust estimate of its performance. This helps ensure that the model generalizes well to unseen data.

- How to encode a decision tree as a computer program ?
  by using sequence of if else block
- For the same dataset, multiple decision trees can be constructed. How to choose the best ? one that requires less number of search in decision tree
- How to deal with missing values ?
  ignore if there are less null values
  check if some default value can be added based on business usecase
  estimate the missing value
  use another decision tree to find missing value
- How to deal with noisy data ?
  up
  winsorization
  mean trimming (avg timming)
  binning
- What is a best split for numerical values ?
  2 ways to do this
  1) r way split - Each split represents each distinct value
  2) binary split - Split based on binary condition x<=a for attribute value x and constant a
- Is there a way to incorporate uncertainty ?
- How to measure the correctness of a model ?
  training data can be divided into training and test data
  prepare model from training data
  test it via test data

Traditional decision trees typically output a deterministic prediction, but there are ways to account for uncertainty.
1. Probabilistic Decision Trees - Instead of providing a single prediction, some decision tree variants can estimate probabilities for each class. Techniques like Bayesian Decision Trees.
2. Bootstrap Aggregating (Bagging) - Bagging is an ensemble method that can be applied to decision trees. By training multiple decision trees on bootstrapped samples of the data and combining their predictions, you can obtain a more robust model that inherently captures some level of uncertainty.
3. Random Forests: Random Forests are an extension of bagging. They build multiple decision trees by using random subsets of features for each tree and then average the predictions.
This randomness adds a level of uncertainty to the model and makes it more robust.

Decision Tree Induction

Algorithm *GenericDecisionTree*(Data Set: $\mathcal{D}$)
begin
  Create root node containing $\mathcal{D}$;
  repeat
    Select an eligible node in the tree;
    Split the selected node into two or more nodes
      based on a pre-defined split criterion;
  until no more eligible nodes for split;
  Prune overfitting nodes from tree;
  Label each leaf node with its dominant class;
end

Many Algorithms: Hunt, CART, ID3, C4.5.

Entropy

- Entropy is a measure developed in Information Theory.
- Entropy is a measure of the uncertainty about a source of messages.
- The more uncertain a receiver is about a source of messages, the more information that receiver will need in order to know what message has been sent.
- If the source always sends exactly the same message, the entropy of such a source is zero.
- If the source send n messages, with equal probability, the uncertainty is maximized, as well as the entropy.
- In such a case, the receiver needs to ask $\log_2 n$ yes/no questions to know the message. In other words, the receiver needs to acquire $\log_2 n$ bits of information to know the message.
- Why ?

Entropy

- Consider a source S that can produce k messages (s_1, ..., s_k), independently, with equal probabilities, the entropy of such a source is:

$$E(S) = -\sum_{j=1}^{k} p_j \log_2(p_j)$$

Example

$$E(S) = -\sum_{j=1}^{k} p_j \log_2(p_j)$$

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Table 1: The weather data (Witten and Frank; 1999, p. 9).

If the class attribute (the message) is ‘play’, what is the entropy of this source ?

Example

$$E(S) = -\sum_{j=1}^{k} p_j \log_2(p_j)$$

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Table 1: The weather data (Witten and Frank; 1999, p. 9).

If the class attribute (the message) is 'play', what is the entropy of this source ?

Entropy (5/14, 9/14)= - ((5/14) $\log_2$ (5/14) + (9/14) $\log_2$(9/14))= 0.940

Entropy-Split

Now consider that we split the database into r groups (r-way split of a decision tree node). The information needed to identify a record within each group is calculated as in the previous slide. The weighted average over all groups is:

$$\text{Entropy-Split}(S \Rightarrow S_1 \dots S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} E(S_i)$$

Example

$$\text{Entropy-Split}(S \Rightarrow S_1 \ldots S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} E(S_i)$$

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Table 1: The weather data (Witten and Frank; 1999, p. 9).

Entropy-split (humidity)=

[(7/14 E(humidity=high) + 7/14 E(humidity=normal)) =

[(-7/14 (4/7log(4/7) + 3/7log(3/7))) + (-7/14 (1/7log(1/7) + 6/7log(6/7)))] = 0.7884504573

Gain

- In choosing the split attribute in a decision tree, we are interested in how much information we get about the output attribute if we choose a certain input attribute as a split.
- This is just the difference between the information needed to classify the record before and after knowing the value of the input attribute.
- The information gain is equal to the reduction in the entropy:

$$E(S) - \text{Entropy-Split}(S \Rightarrow S_1 \dots S_r)$$

- Large values of the gain are more desirable.

Example

$$E(S) = -\sum_{j=1}^{k} p_j \log_2(p_j)$$

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Table 1: The weather data (Witten and Frank; 1999, p. 9).

Gain (temperature)= E(S) – Entropy-Split(Temperature)= 0.94028595867 – 0.91106339301= 0.02922256565

ID3 Tree Induction Algorithm

- A mathematical algorithm for building the decision tree.
- Invented by J. Ross Quinlan in 1979.
- Uses Information Theory invented by Shannon in 1948.
- Builds the tree from the top down, with no backtracking.
- Information Gain is used to select the most useful attribute for classification.

```
function ID3 (I, O, T) {
/*  I is the set of input attributes
*   O is the output attribute
*   T is a set of training data
*
*   function ID3 returns a decision tree
*/
    if (T is empty) {
        return a single node with the value "Failure";
    }
    if (all records in T have the same value for O) {
        return a single node with that value;
    }
    if (I is empty) {
        return a single node with the value of the most frequent value of
        O in T;
        /* Note: some elements in this node will be incorrectly classified */
    }

    /* now handle the case where we can't return a single node */
    compute the information gain for each attribute in I relative to T;
    let X be the attribute with largest Gain(X, T) of the attributes in I;
    let {x_j| j=1,2, .., m} be the values of X;
    let {T_j| j=1,2, .., m} be the subsets of T when T is partitioned
        according the value of X;
        return a tree with the root node labelled X and
        arcs labelled x_1, x_2, .., x_m, where the arcs go to the
        trees ID3(I-{X}, O, T_1), ID3(I-{X}, O, T_2), .., ID3(I-{X}, O, T_m);
}
```

Example

Using ID3, induce the decision tree of:

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Table 1: The weather data (Witten and Frank; 1999, p. 9).

The problem of UID

- What will happen if a unique attribute exists in the training set ?
- In general attributes that have very many values have very high gain, but can lead to useless decision trees.
- Quinlan (1986) suggests choosing the attribute with the highest:
  - GainRatio(X, S)= Gain(X) / Entropy(S) when X is the label attribute
  - GainRatio is proportional with the Gain, favoring attributes with higher gain as before.
  - GainRatio is inversely proportional with the Entropy of the attribute, discouraging attributes with many values.
- Repeat the example using the GainRatio.

Model Validation

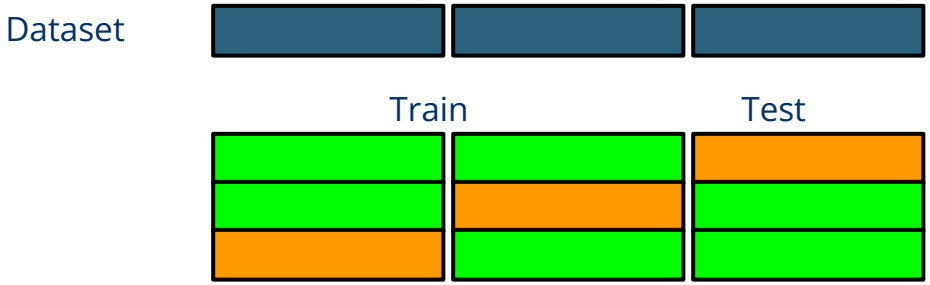
- Holdout

  Labelled data is divided into two disjoint sets, training data and testing data, typically more data in training set
  when large training data, there is high probability that model will predict test data correctly and vice versa
  There is chance that data with certain class may completely be present in training data but absent in test data in which case model will not be able to predict correctly since only subset of data is used for training, full power of training data is not reflected in error estimate
  can also repeat process n times and mean error estimate can be determined

  - Divide labeled data into two disjoint subsets (training set 60%-75%, testing set), randomly selected.
  - Validate the model accuracy using the test set.
  - Classes that are over-represented in the training, will be under-represented in the testing.
- Cross Validation
  - The labeled data is divided into m disjoint subsets of equal size n/m.
  - One of the m segments is used for testing, and the other (m − 1) segments are used for training.

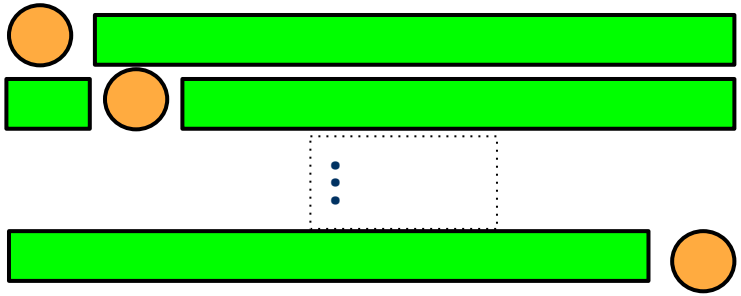
Common Splitting Strategies

- k-fold cross-validation



- Leave-one-out (n-fold cross validation)

data with n records is divided into m subset where (n=m)
each time 1 record is treated as test and m-1 records as training data



Cross Validation

- Average results reported.
- The variance helps determining the statistical confidence intervals of the error.
- Helps to qualify the model. If the model is not performing well on the average, another model should be used.
- Stratified cross validation splits the folds, so that every fold has almost the same distribution of class labels as in the complete dataset.
- Stratified cross validation leads to less pessimistic estimates.

Bootstrap

- Data is sampled uniformly with replacement to create the training set.
- A set of size n is sampled n times, each time a record is randomly selected, copies, then returned back.
- This results in n records that may contain duplicates, and that can miss some of the input records.
- The probability that a certain record is missed in one sampling is 1- 1/n.
- The probability that a certain record is missed in n samplings is $(1- 1/n)^n$. For large n, this approaches 1/e.
- The fraction of the labeled data points included at least once in the training data is therefore $1-1/e \approx 0.632$.

Bootstrap

- The training set hence consists of ≈63.2 % of original dataset, and ≈36.8 % duplicates.
- The original dataset is used as test set to estimate the model accuracy.
- This estimate is highly optimistic because of the high overlap (63.2 %) between training and testing datasets.
- To obtain the variance (and confidence intervals), the whole Bootstrap can be repeated k times.

Bootstrap

- The leave one out bootstrap is a pessimistic variant.
- The accuracy of a labeled record x in the test set is computed using only the b bootstraps (training datasets) that don’t include x.
- The overall accuracy is the average of the accuracy estimates for all possible x.

Bootstrap

- The 0.632-bootstrap further improves this accuracy with a “compromise” approach.

$$A = (0.632) \cdot A_l + (0.368) \cdot A_t$$

- A is the overall accuracy.
- A_l is the overall accuracy of leave-on-out bootstrap (pessimistic).
- A_t is the overall accuracy of bootstrap (optimistic).

Credits

- These slides are mostly made by copying material from:
- Classification: Basic Concepts and Decision Trees, slides by Ruoming Jin,

  http://www.cs.kent.edu/~jin/DM07/ClassificationDecisionTree.ppt
- The ID3 Decision Tree Algorithm, by Daniela Zaharie,

  http://staff.fmi.uvt.ro/~daniela.zaharie/dm2016/projects/DecisionTrees/DecisionTrees_ID3Tutorial.pdf

Data Preparation and Distance Measures

Data Preparation

- Feature extraction: derive meaningful features from data.
- Portability: cast data to unify the structure, and to fit the Algorithm.
- Data cleaning.
- Data Reduction.
- Feature Selection.
- Transformation.

Feature Extraction

- Raw data is not always useful when used as is. Instead, representative features need to be computed, resulting in a feature vector per data record
- The composition of the feature vector is domain specific, and problem specific

Feature Selection vs. Feature Extraction

Feature extraction is different from feature selection.

Feature selection involves choosing a subset of the original features,
while feature extraction creates new features based on the original ones.

Both techniques aim to simplify the data and improve the performance of machine learning models.

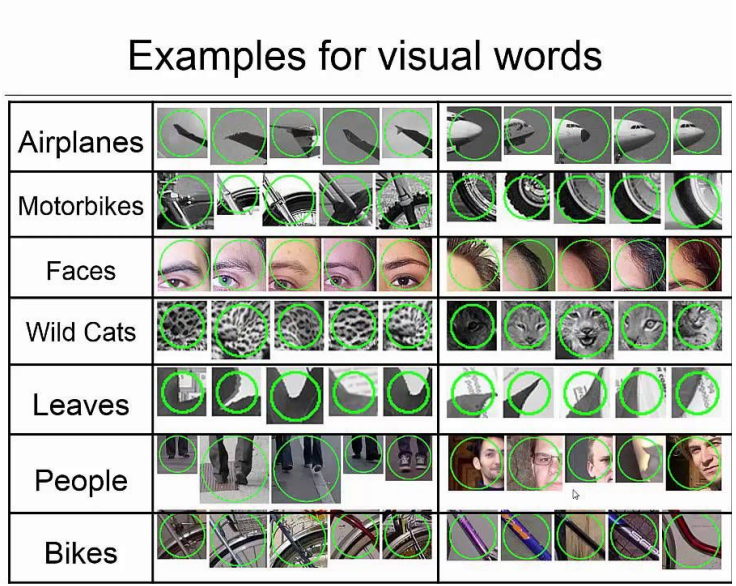
Image Feature Extraction

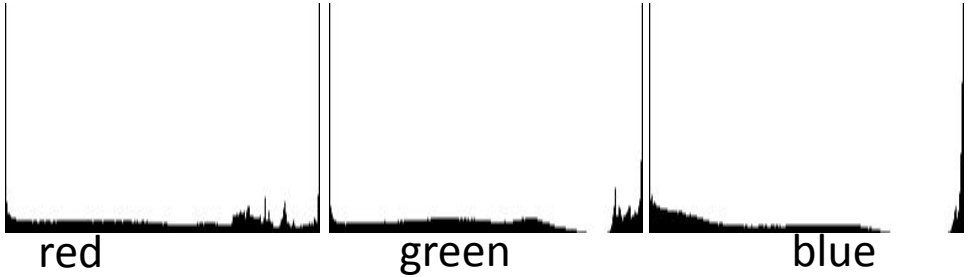
- Color histogram.
- Visual words.

Image Feature Extraction:

In image processing, feature extraction involves capturing relevant information from images.

Techniques like edge detection, texture analysis, and histogram of oriented gradients (HOG) can be used to represent images in a more compact and meaningful way.





Document Feature Extraction

- Named-entity recognition: persons, organizations, locations, actions, cities
- Stop word removal
- Word count histograms

Extract features from document data
Common techniques used for document feature extraction

1) Bag of Words
The Bag of Words model represents each document as an unordered set of words, ignoring grammar and word order but keeping track of word frequency
Problem: we loose word order

2) Extracting n gram
N-grams represent contiguous sequences of N items (usually words) in a document.
Unigrams (single words), bigrams (pairs of consecutive words), and trigrams (triplets of consecutive words) are common choices.
N-grams capture local patterns and dependencies between words.

1 gram is used for token
2 grams for token pair

Stop word removal is a preprocessing step in NLP that involves removing common, non-meaningful words like “the” and “and” from text data

Type Portability

- Text to numeric
- Time series to discrete sequence
- Time series to numeric
- Discrete sequence to numeric
- Spatial to numeric

1) Number to categorical data
Number is divided into several categories by dividing them based on range
Equi width - divided into range of a-b in such a way that b-a is always same
Equi depth - divided into range of a-b in such as way that number of records is always same in that range
Equi log ranges - Range [min,max] is divided into n ranges [ai, bi]. For all i, log(bi)- log(ai) is the same

2) Categorical to Numerical data
It is possible to convert categorical attributes to binary form and then use numeric algorithms on binarized data
Suppose, we have categorical attribute with n possible values,
then n different binary attributes are created
Each binary attribute corresponds to one possible value of categorical attribute
For a record, one of the n binary attributes take value 1 and remaining take value 0

Text to numeric
Bag of words
Stop word removal

Time Series to numeric
Trend
Seasonality
Cycle
Mean Median

Discrete Sequence to numeric
perform scaling (min max scaling)
change to continuous data

Spatial to numeric

Type Portability

Text Documents

- Are all words important ?
- Are different words equally important ?
- Entity extraction.
- Bag of words.

Type Portability - Numeric to categorical (discretization):

- Divide a continuous range into n ranges.
- Age: [0,5], [6,12], [13,18], [18,30], [30,60], 60+
- Lossy process. Age 5 and 6 are equally different as Age 1, 11.
- How to discretized ?

Type Portability - Numeric to categorical (discretization):

- Equi-width ranges. Range [min,max] is divided into n ranges $[a_i, b_i]$.
  - For all i, $b_i - a_i$ is the same.
  - Good for data that is uniformly distributed.
- Equi-log ranges. Range [min,max] is divided into n ranges $[a_i, b_i]$.
  - For all i, $\log(b_i) - \log(a_i)$ is the same.
  - Good for data that is shows an exponential distribution.
- Equi-depth ranges
  - Each range has the same number of records.
  - Sort then select the division points.

Data Cleaning?

In real world, Data can be:
1) Incomplete - Null values present in some field
2) Noisy - Error in data or outliers
3) Inconsistent - Discrepancy in data collected from two different sources, Age and DOB different
4) Duplicate

Data in the real world is dirty

- incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
  - e.g., occupation=“ ”
- noisy: containing errors or outliers
  - e.g., Salary=“-10”
- inconsistent: containing discrepancies in codes or names
  - e.g., Age=“42” Birthday=“03/07/1997”

Why Is Data Dirty?

- Incomplete data may come from
  - “Not applicable” data value when collected.
  - Different considerations between the time when the data was collected and when it is analyzed.
  - Human/hardware/software problems.
- Noisy data (incorrect values) may come from
  - Faulty data collection instruments.
  - Human or computer error at data entry.
  - Errors in data transmission.
- Inconsistent data may come from
  - Different data sources.
  - Functional dependency violation (e.g., modify some linked data).
- Duplicate records also need data cleaning.

How to Investigate ?

- Descriptive Analytics:
  - Null count.
  - Repeated data.
  - Numeric data: max, min, mean, median, variance.
  - Images: image dimensions, resolution, average color, color histogram.
  - GPS: sampling rate (max, min, mean), segment speed (max, min, mean), bounding box.

Measuring the Central Tendency (1 number summary)

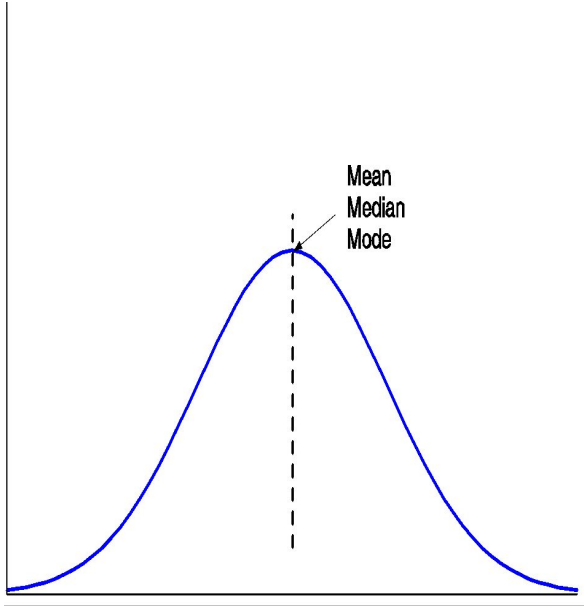
- Mean (algebraic measure) (sample vs. population):
  - Weighted arithmetic mean:
  - Trimmed mean: chopping extreme values

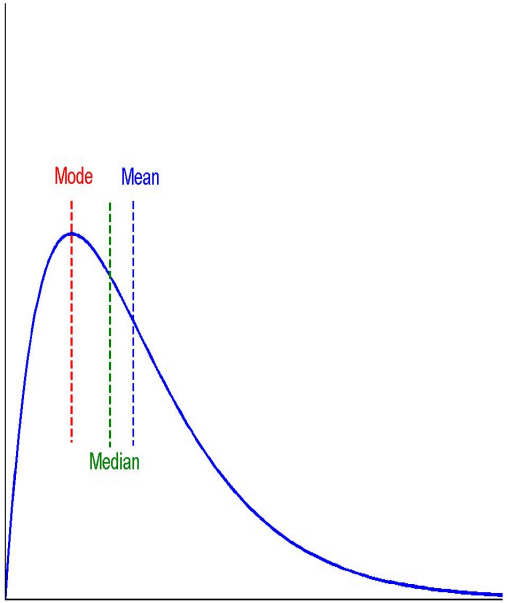
$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

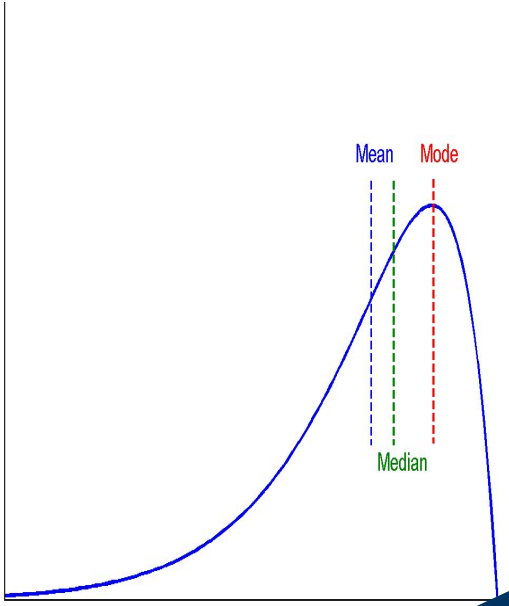
- Median: A holistic measure
  - Middle value if odd number of values, or average of the middle two values otherwise
- Mode
  - Value that occurs most frequently in the data
- 13, 13, 13, 13, 14, 14, 16, 18, 21
  - Mean= sum(13, 13, 13, 13, 14, 14, 16, 18, 21)/9
  - Median= 14, Mode= 13, Range= 8

Symmetric vs. Skewed Data (3 number summary)

- Median, mean and mode of symmetric, positively and negatively skewed data





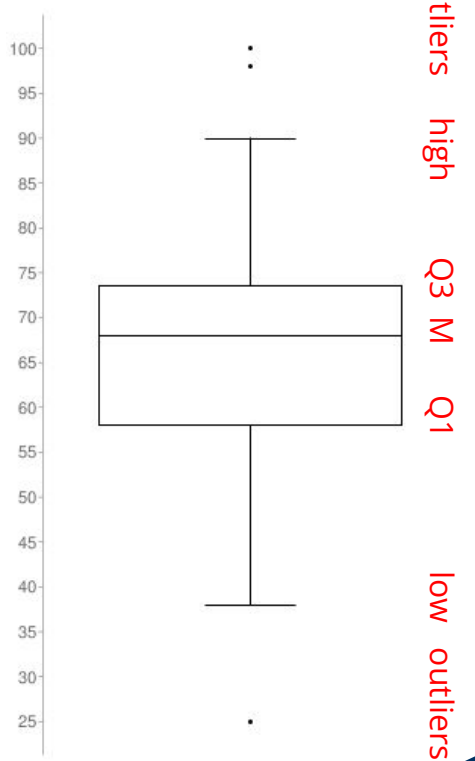


Measuring the Dispersion of Data (5 number summary)

- Quartiles: Q1 (25th percentile), Q3 (75th percentile)
- Inter-quartile range: IQR = Q3 – Q1
- Five number summary: low, Q1, median, Q3, high
- Boxplot: ends of the box are the quartiles, median is marked, whiskers, and plot outlier individually
- Outlier: *usually*, a value higher/lower than 1.5 x IQR

Data:
[25,38,52,57,57,58,58,58,58
,58,58,63,66,66,67,67,68,68
,68,68,68,68,69,70,70,70,70
,72,73,75,75,76,76,78,79,90
,98,100]

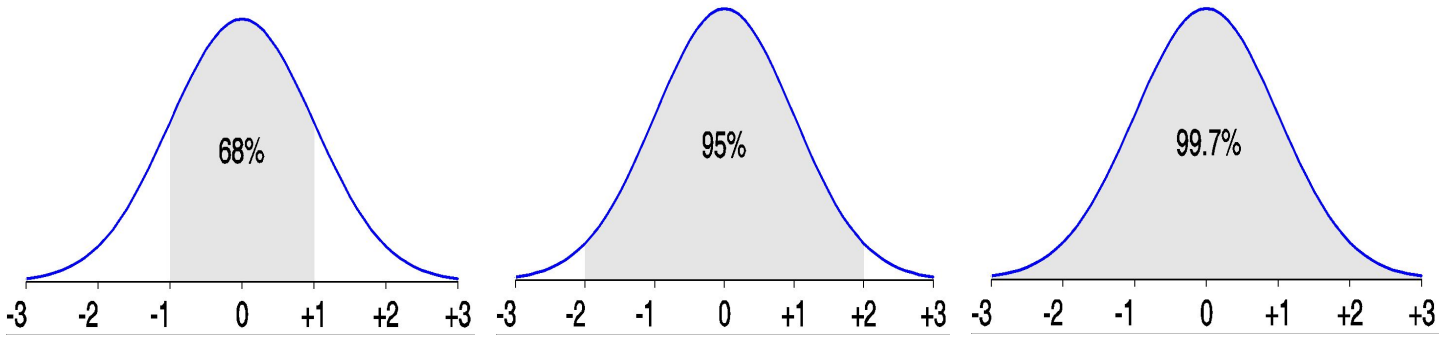
Population size: 38
Median: 68
Minimum: 25
Maximum: 100
First quartile: 58
Third quartile: 73.5
Interquartile Range: 15.5
Outliers: 25 100 98



https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

Comparing with Normal Distribution

- From μ–σ to μ+σ: contains about 68% of the measurements (μ: mean, σ: standard deviation)
- From μ–2σ to μ+2σ: contains about 95% of it
- From μ–3σ to μ+3σ: contains about 99.7% of it



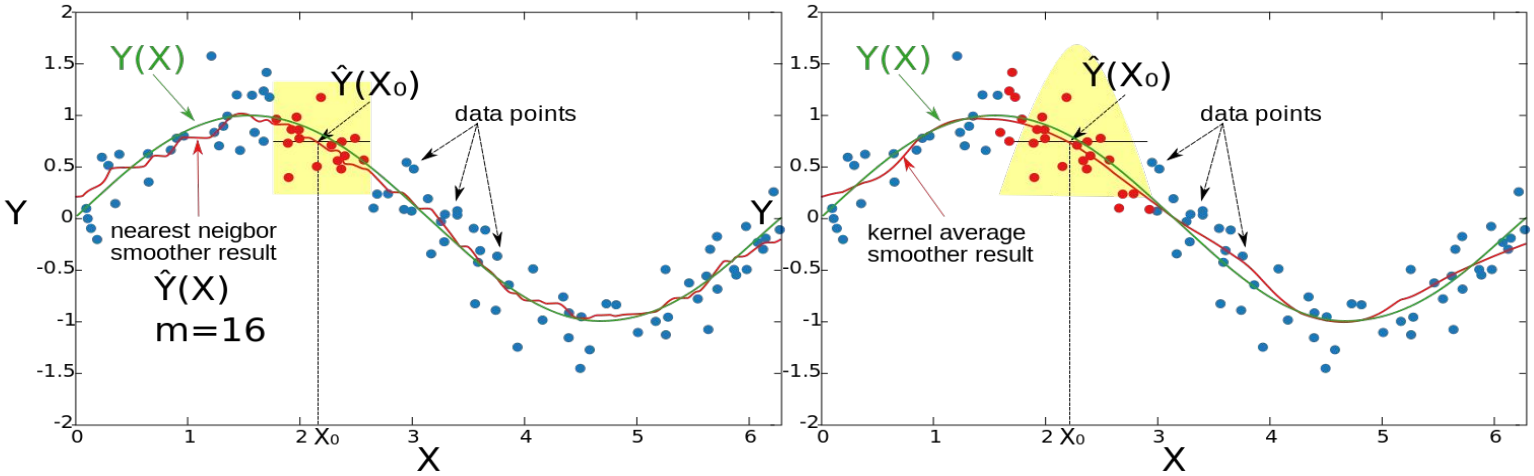
How to Clean Missing and Inconsistent Entries ?

- Delete the entire record: safe, but not practical in most cases.
- Impute/estimate the missing values: can lead to data bias.
- Global constant, mean, substitute with a value of a similar record, interpolation, extrapolation, regression, cluster representative.
- Change the mining Algorithm to tolerate missing values.

How to Clean Noisy Entries ?

Kernel smoothing:

- kNN smoother replaces a value with the average of itself and the k nearest values.
- kernel average smoother replaces a value with the weighted average of itself and its neighbors in a fixed size window



Source: https://en.wikipedia.org/wiki/Kernel_smoother

How to Clean Noisy Entries ?

- Binning:
  - sort data and partition into equally sized bins, then smooth by bin mean, median, boundary.
- Regression:
  - smooth by fitting the data into a regression function.
- Change the mining Algorithm to tolerate noise.

Binning Example

Sorted data for price: 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

- Partition into equal-frequency (equi-depth) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
- Smoothing by bin means:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29

Similarity and Distance

- Given two objects a, b, determine their similarity sim(a,b) or distance dist(a,b).
- Most of the DM Algorithms require the computation of similarity or distance.
- dist(a,b)= 1.43 is not interesting, while if we also know that dist(a,c)=0.5 it becomes interesting as we know that a is more similar to c than b.
- Useful DM analysis depends on expressive distance function, which depends on good selection of features and good normalization.

L_p-norm

$$Dist(\overline{X}, \overline{Y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p}.$$

- Euclidean (p=2), Manhattan (p=1).
- Compute the L_p-norm between (1, 2), (3, 4) for p = 1, 2,∞.

Euclidean
distance between two points

Manhattan
Distance between two points measured along axes at right angle
it is distance measure that is calculated by taking sum of distances between x and y co-ordinates

Generalized Minkowski

Generalized Lp distance
Generalized form of Euclidean and Manhattan distance

$$Dist(\overline{X}, \overline{Y}) = \left(\sum_{i=1}^{d} a_i \cdot |x_i - y_i|^p \right)^{1/p}.$$

- According to the domain, some features may be more important than others.
- Generally p is set to the number of dimensions.
- X, Y are represented as two points in the p-dimensional space. Their distance represents the length of the line connecting them.

Scaling and Normalization

- Multidimensional data has different scales for the different dimensions, resulting in features dominating others in distance computations.

Example:

P1: ULB, 1/1/2018, 10:00

P2: C. d'Ixelles, 1/1/2018, 11:00

P3: Beijing, 1/1/2018, 10:30

d(P1, P3)= (8000 km, 30 minutes).

The Euclidean distance= 8000

d(P1, P2)= (1 km, 60 minutes)=1800

The Euclidean distance= 60

Which pair is more close ?

In case of multidimensional data, different dimensions have different scales resulting in one feature to dominate other in distance computations. To solve this, use scaling and normalization

Scaling and Normalization

Normalization changes data in such a way that it has a mean of 0 and a standard deviation of 1
For this each value i of attribute j is replaced with:

- Normalization replaces each value with:

$$z_i^j = \frac{x_i^j - \mu_j}{\sigma_j}$$

- Assuming a normal distribution of the attribute values, most of the values will be normalized to the range [-3, 3].
- Scaling maps the values to the range [0, 1]

$$y_i^j = \frac{x_i^j - min_j}{max_j - min_j}$$

It is common that we obtain data that is not in same units or range. This can cause issue when we are trying to compare of combine data, or when we are trying to build model out of this

Scaling is used when we want to compare data in different units or when we want to ensure that outliers do not have much influence on our results.

Normalization ensures that data points are centered around 0 and that they are all within 1 standard deviation of the mean. It is used when we want to compare data in different units or when we want to ensure that outliers do not have much influence on our results.

Edit Distance

There are operations like
r: replace one character by another
i: insert one character
d: delete one character

Edit distance between two strings is the minimum number of string operations needed to transform string 1 to string 2

- How similar is Mahmoud to Mohammed ?

```
M a h * m o u d
M o h a m m e d
✓ r ✓ i ✓ r r
✓
```

- Edit distance= 4
- If replace costs 2 (Levenshtein), then distance is 7

Finding Minimum Edit Distance

For two sequences a and b with lengths i and j respectively, the Levenshtein distance is defined as:

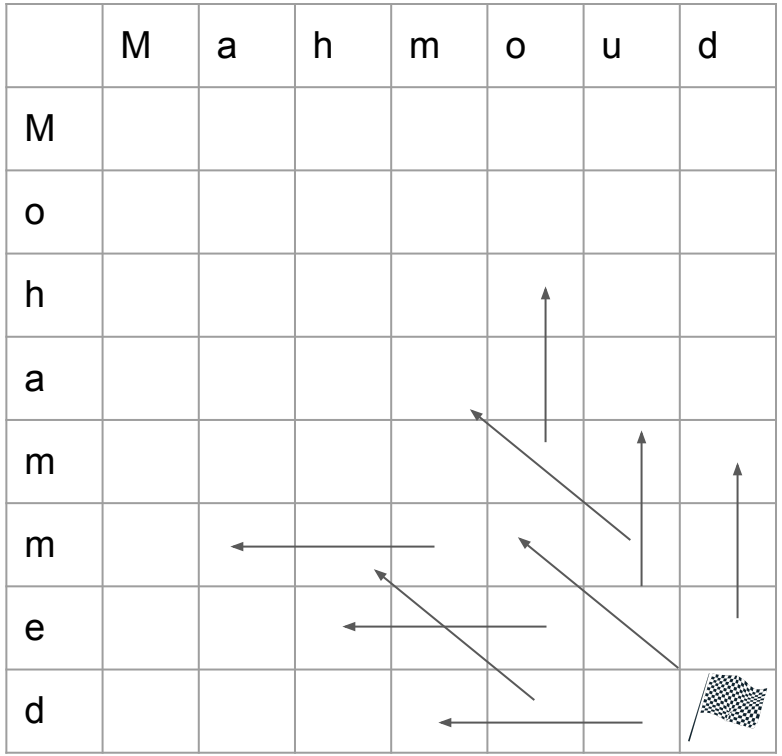
$$
\mathrm{ld}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{ld}(i-1,j) + 1 \\ \mathrm{ld}(i,j-1) + 1 \\ \mathrm{ld}(i-1,j-1) + k \end{cases} & \text{otherwise.} \end{cases}
$$

where $k = 0$ if $(a_i = b_j)$, 1 otherwise.

Recursive Computation of Edit Distance

$$\min \begin{cases} ld(i-1,j) + 1 \\ ld(i,j-1) + 1 \\ ld(i-1,j-1) + k \end{cases}$$

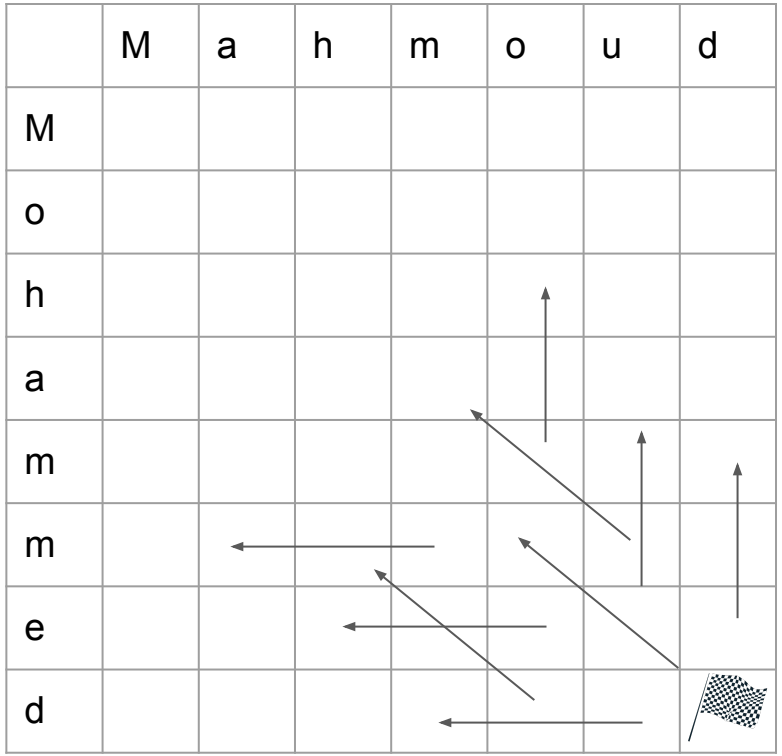
What is the complexity ?



Recursive Computation of Edit Distance

$$\min \begin{cases} ld(i-1,j) + 1 \\ ld(i,j-1) + 1 \\ ld(i-1,j-1) + k \end{cases}$$

- Time: O(i.j)
- Space: O(i.j)
- Backtrace: O(i+j)



ULB

Dynamic Programming

Initialization

	*	M	a	h	m	o	u	d
*	0	1	2	3	4	5	6	7
M	1							
o	2							
h	3							
a	4							
m	5							
m	6							
e	7							
d	8							

Dynamic Programming

row by row, or column by column

```
for i ← 1 to |s1|
do for j ← 1 to |s2|
   do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
                    m[i − 1, j] + 1,
                    m[i, j − 1] + 1}
return m[|s1|, |s2|]
```

What is the complexity ?

	*	M	a	h	m	o	u	d
*	0	1	2	3	4	5	6	7
M	1	0	1	2	3	4	5	6
o	2	1	1	2	3	3	4	5
h	3	2	2	1	2	3	4	5
a	4	3	2	2	2	3	4	5
m	5	4	3	3	2	3	4	5
m	6	5	4	4	3	3	4	5
e	7	6	5	5	4	4	4	5
d	8	7	6	6	5	5	5	

Dynamic Programming

row by row, or column by column

```
for i ← 1 to |s1|
do for j ← 1 to |s2|
   do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
                    m[i − 1, j] + 1,
                    m[i, j − 1] + 1}
return m[|s1|, |s2|]
```

- Time: O(i.j)
- Space: O(i.j)
- Backtrace: 0

Can we do better ?

	*	M	a	h	m	o	u	d
*	0	1	2	3	4	5	6	7
M	1	0	1	2	3	4	5	6
o	2	1	1	2	3	3	4	5
h	3	2	2	1	2	3	4	5
a	4	3	2	2	2	3	4	5
m	5	4	3	3	2	3	4	5
m	6	5	4	4	3	3	4	5
e	7	6	5	5	4	4	4	5
d	8	7	6	6	5	5	5	

Dynamic Programming

row by row, or column by column

```
for i ← 1 to |s1|
do for j ← 1 to |s2|
   do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
                    m[i − 1, j] + 1,
                    m[i, j − 1] + 1}
return m[|s1|, |s2|]
```

We only need two rows/columns

- Time O(m.n)
- Space: O(min(m.n))

	*	M	a	h	m	o	u	d
*								
M								
o								
h								
a	4	3	2	2	2	3	4	5
m	5							
m								
e								
d								

Other Distance Functions

- Frechet distance for spatial and spatiotemporal sequences
- Histogram distance (Earth Mover)
- Longest common sub-sequence.
- Cosine similarity
- Graph edit distance
- Supervised similarity functions

Readings

- The topics discussed in this lecture are explained in Chapter 2,3 of:
  - Charu C.Aggarwal. Data Mining The Textbook, Springer.

Clustering

What is Clustering

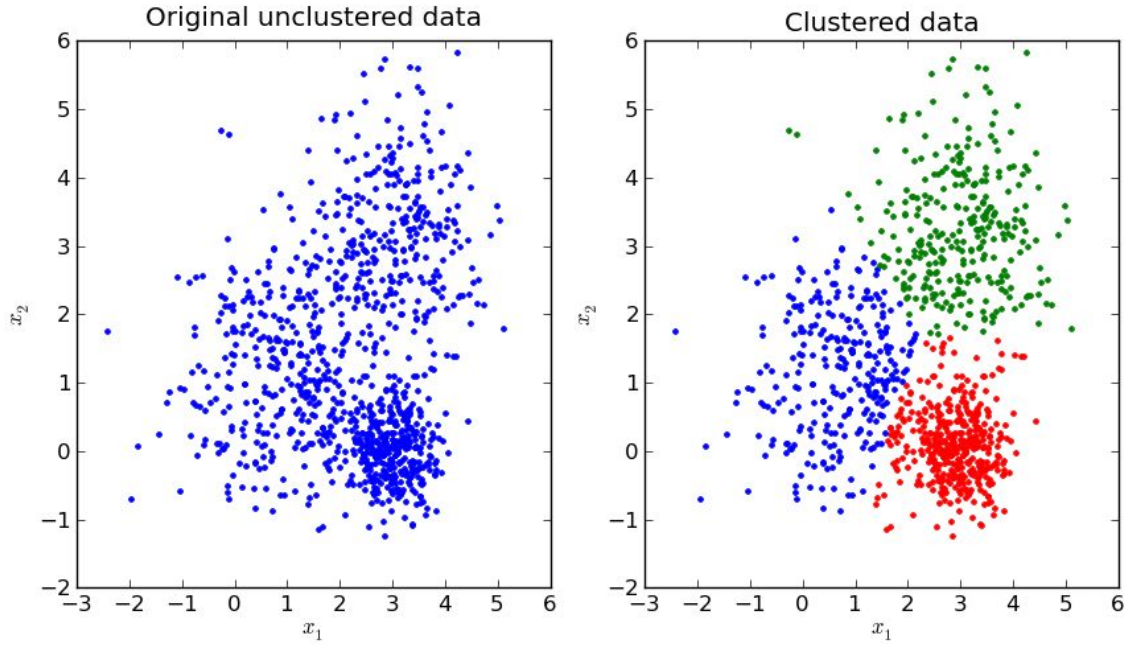
- Many applications require the partitioning of data points into intuitively similar groups.
- An informal and intuitive definition of clustering is as follows:
  - Given a set of data points, partition them into groups containing very similar data points.
- Applications include:
  - Data summarization: use cluster representitives as a summary.
  - Customer segmentation: group customers based on their attributes.
  - Social network analysis: detect communities.
  - Helper for other DM problems: e.g., clustering for outlier analysis.

Clustering Methods

- Representative-Based Algorithms
  - K-Means, K-Medians, K-Medoids.
  - Find a high quality set of representatives, then link data points to their closest representatives.
- Hierarchical Clustering Algorithms
  - Agglomerative: keep grouping similar objects/nodes.
  - Divisive: keep splitting dissimilar groups.
- Probabilistic Model-Based Algorithms:
  - Soft clustering.
- Density-Based Algorithms
  - DBSCAN, DENCLUE.

Representative-Based Algorithms

- Input: k, the number of clusters.



https://mubaris.com/posts/kmeans-clustering/

Representative-Based Algorithms

- Consider a data set *D* containing *n* data points denoted by $X_1 \ldots X_n$ in *d*-dimensional space.
- The goal is to determine *k* representatives $Y_1 \ldots Y_k$ that minimize the following objective function *O*:

$$O = \sum_{i=1}^{n} \left[\min_j Dist(\overline{X_i}, \overline{Y_j}) \right] .$$

Representative-Based Algorithms

- Note that the optimal assignment of data points to representatives (Y1…Yn) are unknown a priori.
- The assignment of points to representatives requires first finding the representatives, and finding the representatives requires first knowing the groups. Chicken and egg !
- Such optimization problems are solved with the use of an iterative approach where candidate representatives and candidate assignments are used to improve each other.

Representative-Based Algorithms

- Generic *k*-representatives approach starts by initializing the *k* representatives *S* with the use of a straightforward heuristic (such as random sampling from the original data), and then refines the representatives and the clustering assignment, iteratively, as follows:
  - (Assign step) Assign each data point to its closest representative in *S* using distance function *Dist*(·, ·), and denote the corresponding clusters by *C*1 . . . *Ck*.
  - (Optimize step) Determine the optimal representative *Yj* for each cluster *Cj* that minimizes its *local* objective function

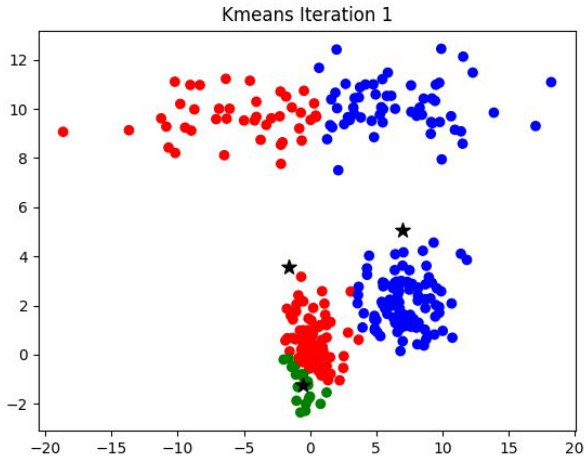
$$\sum_{\overline{X_i} \in \mathcal{C}_j} \left[Dist(\overline{X_i}, \overline{Y_j}) \right]$$

Representative-Based Algorithms

Algorithm *GenericRepresentative*(Database: $\mathcal{D}$, Number of Representatives: k
begin
  Initialize representative set S;
  repeat
    Create clusters $(\mathcal{C}_1 \ldots \mathcal{C}_k)$ by assigning each
        point in $\mathcal{D}$ to closest representative in S
        using the distance function $Dist(\cdot, \cdot)$;
    Recreate set S by determining one representative $\overline{Y_j}$ for
      each $\mathcal{C}_j$ that minimizes $\sum_{\overline{X_i} \in \mathcal{C}_j} Dist(\overline{X_i}, \overline{Y_j})$;
  until convergence;
  return $(\mathcal{C}_1 \ldots \mathcal{C}_k)$;
end

Representative-Based Algorithms

- The idea is to improve the objective function over multiple iterations.
- Typically, the increase is significant in early iterations, but it slows down in later iterations. The primary computational bottleneck of the
- approach is the assignment step where the distances need to be computed between all point representative pairs.



Whenever a cluster is created, we will calculate means of datapoints in the cluster that will be considered as centroid

The k-Means Algorithm

- A variant of the Representative-Based Algorithms, where the objective function is the sum of the squares of the Euclidean distances (L_2-norm) of data points to their closest representatives (SSE).

$$Dist(\overline{X_i}, \overline{Y_j}) = ||\overline{X_i} - \overline{Y_j}||_2^2.$$

The K-Means Algorithm

K=3, D={A1=(2,10), A2=(2,5), A3=(8,4), A4=(5,8), A5=(7,5), A6=(6,4), A7=(1,2), A8=(4,9)}. Choose your seeds.

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	$\sqrt{25}$	$\sqrt{36}$	$\sqrt{13}$	$\sqrt{50}$	$\sqrt{52}$	$\sqrt{65}$	$\sqrt{5}$
A2		0	$\sqrt{37}$	$\sqrt{18}$	$\sqrt{25}$	$\sqrt{17}$	$\sqrt{10}$	$\sqrt{20}$
A3			0	$\sqrt{25}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{53}$	$\sqrt{41}$
A4				0	$\sqrt{13}$	$\sqrt{17}$	$\sqrt{52}$	$\sqrt{2}$
A5					0	$\sqrt{2}$	$\sqrt{45}$	$\sqrt{25}$
A6						0	$\sqrt{29}$	$\sqrt{29}$
A7							0	$\sqrt{58}$
A8								0

https://webdocs.cs.ualberta.ca/~zaiane/courses/cmput695/F07/exercises/Exercises695Clus-solution.pdf

The K-Means Algorithm

- What is the run time complexity of K-Means ? ppt
- Does k-means select the representatives among the original points ?

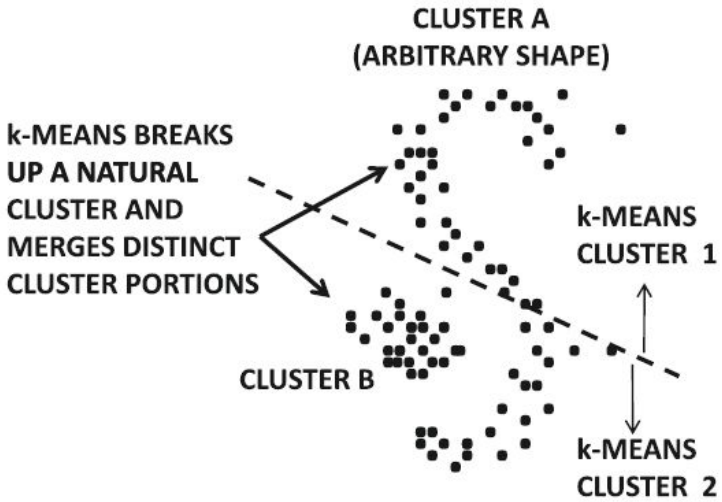
  k means - No
  k median and k medoid - Yes
- Can you think of disadvantages ?

  Mean is more sensitive to outlier
  So, k means algorithm is not immune to noise and outliers
  Less flexible
  Suitable for clusters of spherical shape

Representative data point may not be one of the input data point
Not immune to noise and outliers
Objective function selected is Euclidean distance

The K-Means Algorithm

- K-Means does not work well for clusters of arbitrary shape.
- It is biased towards finding spherical clusters.



The K-Medians Algorithm

what is median when we are solving two points?

- *Dist*(*Xi, Yj*) is the Manhattan distance:

$$Dist(\overline{X_i}, \overline{Y_j}) = ||X_i - Y_j||_1.$$

- Accordingly representative of a group is the median point. This is because the point that has the minimum sum of *L*1-distances to a set of points distributed on a line is the median of that set.
- The median is chosen independently at each dimension.
- As the median is less sensitive to outliers, K-Medians is more robust than K-Means.

The K-Medoids Algorithm

Medoid means least dissimilar
It is defined as the point in cluster, whose dissimilarities with all other points in cluster is minimum

Algorithm *GenericMedoids*(Database: $\mathcal{D}$, Number of Representatives: k)
begin
  Initialize representative set S by selecting from $\mathcal{D}$;
  repeat
    Create clusters $(\mathcal{C}_1 \ldots \mathcal{C}_k)$ by assigning
      each point in $\mathcal{D}$ to closest representative in S
      using the distance function $Dist(\cdot, \cdot)$;
    Determine a pair $\overline{X_i} \in \mathcal{D}$ and $\overline{Y_j} \in S$ such that
      replacing $\overline{Y_j} \in S$ with $\overline{X_i}$ leads to the
      greatest possible improvement in objective function;
    Perform the exchange between $\overline{X_i}$ and $\overline{Y_j}$ only
      if improvement is positive;
  until no improvement in current iteration;
  return $(\mathcal{C}_1 \ldots \mathcal{C}_k)$;
end

The K-Medoids Algorithm

Advantages 1. Simple to understand and easy to implement
2. is fast and converges in a fixed number of steps
3. is less sensitive to outliers than other clustering algorithm
When an algorithm is said to be less sensitive to outliers, it means that the presence of outliers has a smaller impact on the algorithm's performance or results.

Disadvantages 1. non suitable for clustering non spherical groups of objects
2. Result vary on initial medoid selected

- Cluster representatives are always chosen from the dataset D.
- Requires only a distance function. No mean/median is required. So it can be used for complex types.
- Uses hill-climbing strategy, in which the representative set *S* is initialized to a set of points from the original database *D*.
- Subsequently, this set *S* is iteratively improved by exchanging a single point from set *S* with a data point selected from the database *D*.
- This iterative exchange can be viewed as a hill-climbing strategy, because each exchange can be viewed as a hill-climbing step.
- At every iteration, try multiple exchanges, and choose the best.

1. From n data points, select k random points as medoid
2. Associate each data point to closest medoid by using any of the most common distance methods
3. Calculate total cost for forming this cluster
4. Randomly select one non medoid point and select it as new medoid in this case
5. Calculate distance, assign each data point to closest medoid
5. Calculate total cost for forming this cluster
6. Compare previous total cost with new total cost
7. If new total_cost>prev_totalcost, so here swapping is not good
8. If cost reduces, repeat the process

in ppt

Practical Issues

- How to initialize the cluster representatives ?
  - Random.
  - Sample, then use another clustering method.
  - Sample k times, and use the centroid of each.
- K-Means can get stuck with a singleton cluster if an outlier is used in initialization. What to do ?

ppt

- It is difficult to determine a good value k. One may chose a bigger value than the analyst's k, then perform cluster merging as a post processing.

elbow method

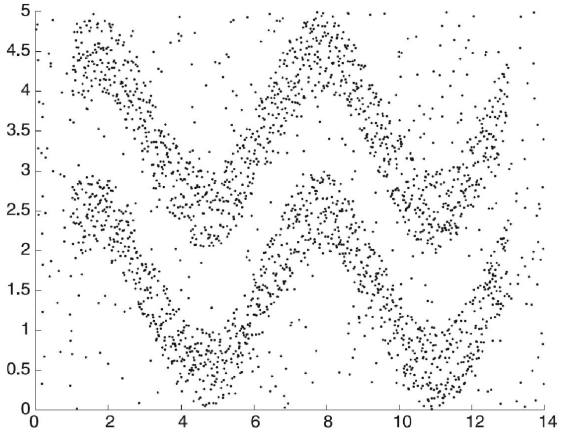
Credits and Readings

- These slides, except when explicitly stated, use material from:
  - Charu C.Aggarwal. Data Mining The Textbook, Springer.

Clustering
Grid & Density-based Algorithms

Why Grid-Density-Based ?

- Representative-Based clustering Algorithms are also distance based.
- One of the major problems is that the shape of the underlying clusters is already defined implicitly by the underlying distance function.
- For example, a *k*-means algorithm implicitly assumes a spherical shape for the cluster.
- In practice, the clusters may be hard to model with a prototypical shape implied by a distance function.

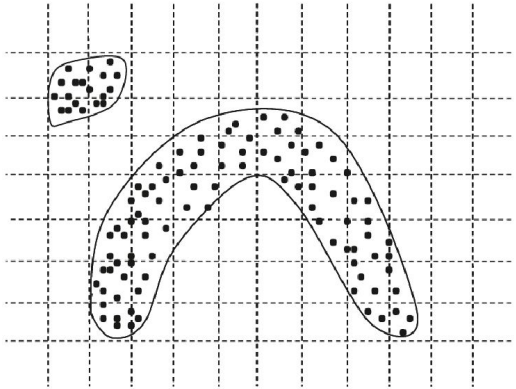


Grid & Density-based Algorithms

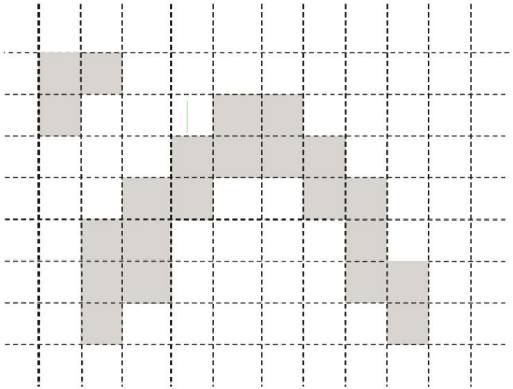
- The core idea in density-based algorithms is to perform two iterations:
    1) detect dense areas in the data,
    2) then grow and merge them.
- It is thus possible to detect clusters with complex shapes.
- Many variations of this broader principle exist.
    - DBSCAN: pre-selected data points in dense regions are clustered with a single-linkage method
    - DENCLUE: use gradient ascent on the kernel-density estimates

Grid-Based Algorithms

- Data is discretized into p intervals, typically equi-width
- A total of p^d hyper-cubes, where d is the number of dimensions in the underlying data
- Clustering is done over the hyper-cubes
- A density threshold τ is used to determine the subset of the p^d hyper-cubes that are dense



(a) Data points and grid



(b) Agglomerating adjacent grids

Grid-Based Algorithms

- *Adjacently connected*: if two cells a side/face
  - A weaker version: if they share a *corner* in common.
- *Density connected*, if a path can be found from one cell to the other, via a sequence of *adjacently connected*
- Grid-Based Clustering maps to finding the *connected regions* created in such grid
- Easy to implement as graph
  - *dense* grid cells are graph nodes
  - edges represents *adjacent connectivity*
  - => connected components are clusters
- Breadth-first or depth-first traversal on the graph for finding connected components
- What is the complexity ?

Grid-Based Algorithms

Algorithm *GenericGrid*(Data: $\mathcal{D}$, Ranges: p, Density: τ)
begin
  Discretize each dimension of data $\mathcal{D}$ into p ranges;
  Determine dense grid cells at density level τ;
  Create graph in which dense grids are connected if they are adjacent;
  Determine connected components of graph;
  return points in each connected component as a cluster;
end

Parameters

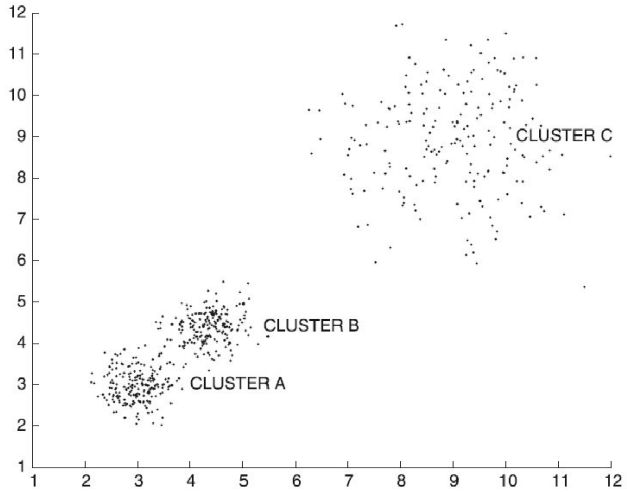
- Number of data clusters is not predefined, in contrast to representative-based
- On the other hand, two different parameters need to be defined:
  - Grid resolution, number of intervals p
  - Density threshold τ .

Choosing p & τ

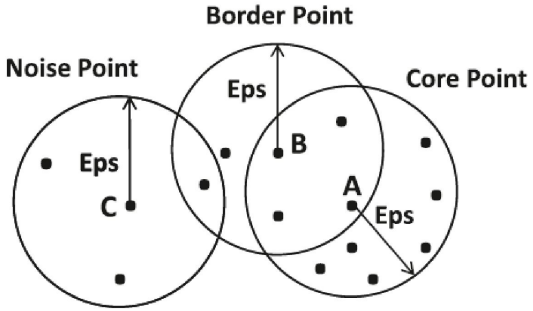
- When p is too small ?
  - points from multiple clusters will be present in the same grid region => undesirable merging of clusters.
- When p is too large ?
  - Many empty grid cells => natural clusters in the data may be disconnected
- When τ is too low ?
  - clusters including the ambient noise, will be merged
- When τ is too high ?
  - We can partially or entirely miss a cluster.

Practical Issues

- A major issue is the use a single density parameter τ
- The grid grows quickly with increasing dimensionality, becoming computationally infeasible in high dimensions



DBSCAN



- Widely used clustering Algorithm.
- Given a set of data points, and two parameters *Eps*, τ, DBSCAN is able to distinguish three types of points:
  - **Core points** are the ones that have at least τ points in their Eps-neighborhood. This is the circle centered at the point itself and has a radius of Eps.
  - **Border points** are the points that fall into the neighborhood of some core points, but themselves are not core points. These points are typically located close to the cluster border, hence their name. They belong to some dense cluster, but themselves are not in the middle of a dense area. Only some side of their neighborhood is dense, but not their complete neighborhood.
  - **Noise points** are the points that are neither core nor border. Basically noise points fall in sparse areas, where the neighborhood of the point itself is not dense, and no core points exist around in the neighborhood.

DBSCAN from Mars

Algorithm *DBSCAN*(Data: $\mathcal{D}$, Radius: Eps, Density: τ)
begin
  Determine core, border and noise points of $\mathcal{D}$ at level (Eps, τ);
  Create graph in which core points are connected
    if they are within Eps of one another;
  Determine connected components in graph;
  Assign each border point to connected component
    with which it is best connected;
  return points in each connected component as a cluster;
end

DBSCAN - Reachability

- (p_i, p_j) are *directly density reachable* if p_i is a core point, and p_j is in the Eps-neighborhood of p_i. Note that this is a non-symmetric relation.
- (p_i, p_j) are *density reachable* if p_i is a core point, and their is a chain of core points P_{i+1} ... p_n, where every consecutive pair in this chain is directly density reachable, and p_j is in the Eps-neighborhood of p_n. In this case, p_j is a border point.
- (p_i, p_j) are *density connected* if both p_i and p_j are density reachable from some point p_k. In this case, both p_i, p_j are border point. This relation is symmetric.

- A cluster is a maximal set of points that are all reachable from one another under any of these three definitions of reachability. A point that is neither directly density reachable, nor density reachable from any core point is a noise that does not belong to any cluster.

```
DBSCAN(D, eps, MinPts)
  C = 0
  for each unvisited point P in dataset D
    NeighborPts = regionQuery(P, eps)
    if sizeof(NeighborPts) < MinPts
        mark P as visited
    else
        C = next cluster
        expandCluster(P, NeighborPts, C, eps, MinPts)
expandCluster(P, NeighborPts, C, eps, MinPts)
  mark P as visited
  add P to cluster C
  for each point P' in NeighborPts
    if P' is not visited
        mark P' as visited
        NeighborPts' = regionQuery(P', eps)
        if sizeof(NeighborPts') >= MinPts
            NeighborPts = NeighborPts joined with NeighborPts'
        if P' is not yet member of any cluster
            add P' to cluster C
```

What is the complexity of DBSCAN?

- The major time complexity of *DBSCAN* is in finding the neighbors of the different data points within a distance of *Eps*.
- For a database of size n, the time complexity can be $O(n^2)$ in the worst case.
- The use of a spatial index for finding the nearest neighbors can reduce this to approximately $O(n \cdot \log(n))$ distance computations.

Practical Issues

- *Eps*, τ are related to one another, how ?
- Progressive DBSCAN:
  - Using the same value for τ, apply DBSCAN in a progressive way, with increasing values of Eps.
  - Start with a small Eps to find intersections are rather dense, then
  - iteratively relax the eps value to find less dense clusters.
  - After every iteration, the points that already belong to some cluster are removed.

Map-making example

Building road segments and detecting turns from gps tracks
M Ezzat, M Sakr, R Elgohary, ME Khalifa - Journal of computational science, 2018

DENCLUE, which stands for DENsity-based CLUstEring, is a density-based clustering algorithm. It was introduced by Hinneburg and Gabriel in 2007 as an extension of the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm. DENCLUE, like DBSCAN, operates based on the density of data points in a given feature space

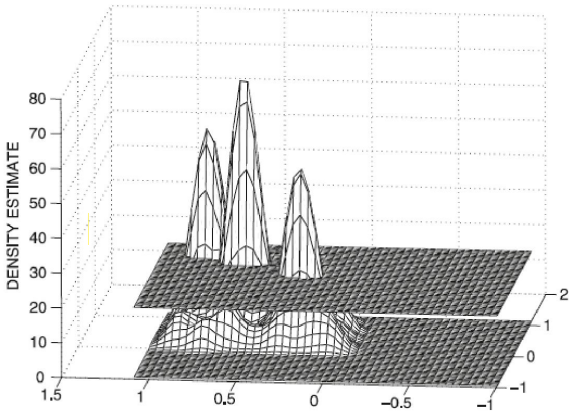
DENCLUE

- The *DENCLUE* algorithm is based on the statistical foundations of kernel density estimation.
- Kernel-density estimation creates a smooth profile of the density distribution.
- In kernel-density estimation, the density $f(X)$ at coordinate X is defined as a sum of the influence (kernel) functions $K(\cdot)$ over the n different data points in the database D:
- For a Gaussian kernel $$f(\overline{X}) = \frac{1}{n}\sum_{i=1}^{n} K(\overline{X} - \overline{X_i})$$

$$K(\overline{X} - \overline{X_i}) = \left(\frac{1}{h\sqrt{2\pi}}\right)^{d} e^{-\frac{||\overline{X} - \overline{X_i}||^2}{2 \cdot h^2}}$$

DENCLUE

- Intuitively, the effect of kernel-density estimation is to replace each discrete data point with a smooth “bump,” and the density at a point is the sum of these “bumps.”
- This results in a smooth profile of the data in which the random artifacts of the data are suppressed, and a smooth estimate of the density is obtained.
- *h* represents the bandwidth of the estimation that regulates the smoothness of the estimation. Large values of the bandwidth *h* smooth out the noisy artifacts but may also lose some detail about the distribution.
- The goal is to determine clusters by using a density threshold τ that intersects this smooth density profile.



Clustering Methods

- Representative-Based Algorithms
  - K-Means, K-Medians, K-Medoids.
  - Find a high quality set of representatives, then link data points to their closest representatives.
- Hierarchical Clustering Algorithms
  - Agglomerative: keep grouping similar objects/nodes.
  - Divisive: keep splitting dissimilar groups.
- Probabilistic Model-Based Algorithms:
  - Soft clustering.
- Density-Based Algorithms
  - DBSCAN, DENCLUE.

Probabilistic Model-Based Clustering

Probabilistic Clustering (why)

- In all the cluster analysis methods we have discussed so far, each data object can be assigned to at most one clusters.
- In what situations may a data object belong to more than one cluster?
  - Clustering product reviews: a customer review might relate to multiple products/services. If we want to cluster reviews per product/service, we must allow that a review can belong to many clusters.
  - Clustering to study user search intent: a user of an online store would typically perform some search. It is important to understand the search intent: searching for product, for customer support, for offers, etc. In one session however, the user may search with multiple intents.

Fuzzy Set and Fuzzy Cluster

Fuzzy cluster: A fuzzy set $F_S : X \rightarrow [0, 1]$ (value between 0 and 1)

Example: Popularity of cameras is defined as a fuzzy mapping

Camera	Sales (units)
A	50
B	1320
C	860
D	270

$$\text{Pop}(o) = \begin{cases} 1 & \text{if } 1,000 \text{ or more units of } o \text{ are sold} \\ \frac{i}{1000} & \text{if } i \ (i < 1000) \text{ units of } o \text{ are sold} \end{cases}$$

Function *pop()* defines a fuzzy set of popular digital cameras. The fuzzy set of digital cameras according to Pop() is {*A*(0.05), *B*(1), *C*(0.86), *D*(0.27)}, where the degree of membership is written in parentheses.

In fuzzy clustering, a cluster is a fuzzy set of objects that belong to this cluster. The degree of membership of every object indicates how strong this object is related to this cluster.

Fuzzy (Soft) Clustering

Formally, given a set of objects, $o_1, \ldots, o_n$, a **fuzzy clustering** of k **fuzzy clusters**, $C_1, \ldots, C_k$, can be represented using a **partition matrix**, $M = [w_{ij}]$. where w_{ij} is the membership degree of o_i in fuzzy cluster C_j. The partition matrix should satisfy the following three requirements:

P1: for each object o_i and cluster C_j, $0 \le w_{ij} \le 1$ (fuzzy set).

P2: for each object o_i, $\sum_{j=1}^{k} w_{ij} = 1$, equal participation in the clustering

P3: for each cluster C_j, $0 < \sum_{i=1}^{n} w_{ij} < n$ ensures there is no empty cluster.

Fuzzy (Soft) Clustering

Example: Let cluster features be

C_1 :"digital camera" and "lens"

C_2: "computer"

$$w_{ij} = \frac{|R_i \cap C_j|}{|R_i \cap (C_1 \cup C_2)|} = \frac{|R_i \cap C_j|}{|R_i \cap \{digital\ camera, lens, computer\}|}.$$

Review-id	Keywords
R_1	digital camera, lens
R_2	digital camera
R_3	lens
R_4	digital camera, lens, computer
R_5	computer, CPU
R_6	computer, computer game

$$M = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ \frac{2}{3} & \frac{1}{3} \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Fuzzy (Soft) Clustering

How to evaluate how well a fuzzy clustering describes a data set ?

Let c_1, …, c_k as the center of the k clusters

For an object o_i, sum of the squared error (SSE), p is a parameter:

$$\mathrm{SSE}(o_i) = \sum_{j=1}^{k} w_{ij}^{p} dist(o_i, c_j)^2$$

For a cluster C_j, SSE:

$$\mathrm{SSE}(C_j) = \sum_{i=1}^{n} w_{ij}^{p} dist(o_i, c_j)^2$$

For the whole clustering:

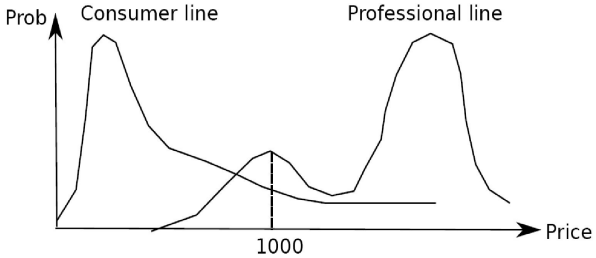
$$\mathrm{SSE}(\mathcal{C}) = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij}^{p} dist(o_i, c_j)^2$$

Probabilistic Model-Based Clustering

- Is there Algorithm to detect probabilistic clusters in the data ?
- A data set that is the subject of cluster analysis can be regarded as a sample of the possible instances of the hidden clusters, but without any cluster labels.
- Statistically, we can assume that a hidden cluster is a distribution over the data space, which can be mathematically represented using a probability density function (or distribution function).
- For a probabilistic cluster, C, its probability density function, f , and a point, o, in the data space, $f(o)$ is the relative likelihood that an instance of C appears at o.

Probabilistic Model-Based Clustering

- Example: Suppose that there are 2 categories for digital cameras sold
  - consumer line vs. professional line
  - density functions f_1, f_2 for C_1, C_2
  - obtained by probabilistic clustering



- For a price value of, say, \$1000, $f_1(1000)$ is the relative likelihood that the price of a consumer-line camera is \$1000. Similarly, $f_2(1000)$ is the relative likelihood that the price of a professional-line camera is \$1000.
- The starting point of our analysis is that we don't know the probability density function of the two clusters.

Probabilistic Model-Based Clustering

- Suppose we want to find k probabilistic clusters, $C_1, \ldots, C_k$, through cluster analysis of D.
- Conceptually, we can assume that D is formed as follows. Each cluster C_j is associated with a probability, w_j. We then run the following two steps n times to generate $D = \{o_1, \ldots, o_n\}$:
  - Choose a cluster, C_j , according to probabilities w_j.
  - Choose an instance of C_j according to its probability density function, f_j.
- This is called the **mixture model.** It assumes that a set of observed objects is a mixture of instances from multiple probabilistic clusters, and conceptually each observed object is generated independently.
- The task of *probabilistic model-based cluster analysis* is to infer infer a set of k probabilistic clusters that is mostly likely to generate D using the above data generation process.

Probabilistic Model-Based Clustering

- Consider a set C of k probabilistic clusters $C_1, ..., C_k$ with probability density functions $f_1, ..., f_k$, respectively, and their probabilities $\omega_1, ..., \omega_k$.
- The probability of an object o generated by cluster C_j is

$$P(o|C_j) = \omega_j f_j(o)$$

- The probability of o generated by the set of cluster $\boldsymbol{C}$ is

$$P(o|\boldsymbol{C}) = \sum_{j=1}^{k} \omega_j f_j(o)$$

- Since objects are assumed to be generated independently, for a data set D = {o_1, ..., o_n}, we have,

$$P(D|\boldsymbol{C}) = \prod_{i=1}^{n} P(o_i|\boldsymbol{C}) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_j f_j(o_i)$$

Probabilistic Model-Based Clustering

- Since objects are assumed to be generated independently, for a data set D = {o_1, ..., o_n}, we have,

$$P(D|\boldsymbol{C}) = \prod_{i=1}^{n} P(o_i|\boldsymbol{C}) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_j f_j(o_i)$$

- Task: Find a set *C* of *k* probabilistic clusters s.t. $P(D|\boldsymbol{C})$ is maximized.
- However, maximizing $P(D|\boldsymbol{C})$ is often intractable since the probability density function of a cluster can take an arbitrarily complicated form.
- To make it computationally feasible (as a compromise), assume the probability density functions being some parameterized distributions

Univariate Gaussian Mixture Model

- Assume that the probability density function of each cluster follows a 1-d Gaussian distribution. Suppose that there are k clusters.
- The probability density function of each cluster are centered at μ_j with standard deviation σ_j, θ_j, = (μ_j, σ_j) is:

$$P(o_i|\Theta_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

- Assuming that each cluster has the same probability w_j:

$$P(o_i|\Theta) = \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

- The task is then to minimize:

$$P(\mathbf{O}|\Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(o_i-\mu_j)^2}{2\sigma^2}}$$

The EM (Expectation Maximization) Algorithm

The k-means algorithm has two steps at each iteration:

- **Expectation Step** (E-step): Given the current cluster centers, each object is assigned to the cluster whose center is closest to the object: An object is *expected to belong to the closest cluster*
- **Maximization Step** (M-step): Given the cluster assignment, for each cluster, the algorithm *adjusts the center* so that *the sum of distance* from the objects assigned to this cluster and the new center is minimized

The (EM) algorithm: A framework to approach maximum likelihood or maximum a posteriori estimates of parameters in statistical models.

- **E-step** assigns objects to clusters according to the current fuzzy clustering or parameters of probabilistic clusters
- **M-step** finds the new clustering or parameters that minimize the sum of squared error (SSE) or the expected likelihood

Quality: What Is Good Clustering?

A good clustering method will produce high quality clusters

- high intra-class similarity: cohesive within clusters
- low inter-class similarity: distinctive between clusters

The quality of a clustering method depends on

- the similarity measure used by the method
- its implementation, and
- Its ability to discover some or all of the hidden patterns

Examples of Quality Measures

- *Sum of square distances to centroids:*
  - The squared distance between from the representative to every other point in the cluster is calculated, then summed over all points.
  - Suitable for representative based methods.
  - Favors spherical clusters.
- *Intracluster to intercluster distance ratio:*
  - Sample pairs of points in D.
  - Let P denote the pairs in the same cluster, and Q denote the pairs in different clusters.
  - Compute:
    - Intra/Inter
  - Smaller values are better.

$$Intra = \sum_{(\overline{X_i},\overline{X_j})\in P} dist(\overline{X_i},\overline{X_j})/|P|$$

$$Inter = \sum_{(\overline{X_i},\overline{X_j})\in Q} dist(\overline{X_i},\overline{X_j})/|Q|$$

Examples of Quality Measures

- Silhouette coefficient:
  - Let $D_{avg\text{-}in}$ denote the average distance between a point in the cluster and every other point in the same cluster.
  - Let $D_{avg\text{-}out\_i}$ denote the average distance between a point in the cluster and every other point in the cluster i.
  - Let $D_{avg\text{-}out\text{-}min}$ be the minimum Davg-out_i.

$$S_i = \frac{Dmin_i^{out} - Davg_i^{in}}{\max\{Dmin_i^{out}, Davg_i^{in}\}}$$

- The silhouette coefficient will be drawn from the range (−1, 1). Large positive values indicate highly separated clustering. Negative values are indicative of some level of “mixing” of data points from different clusters.

Credits and Readings

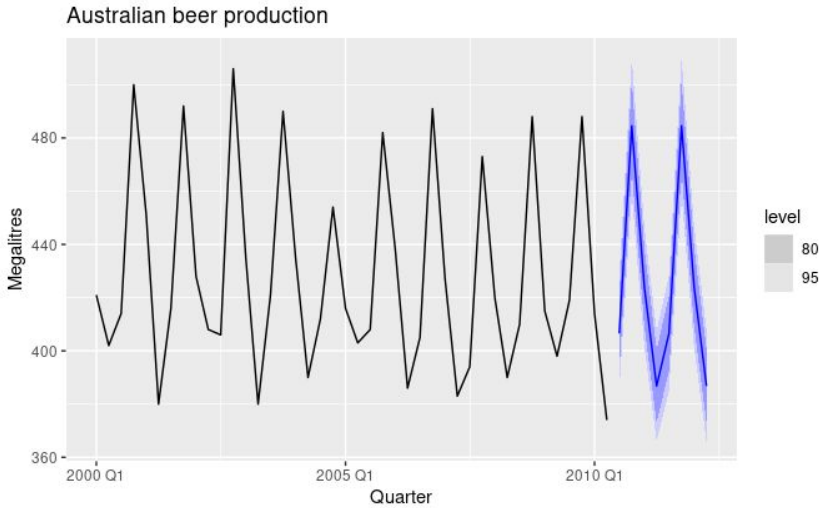
- These slides, except when explicitly stated, use material from:
  - Charu C.Aggarwal. Data Mining The Textbook, Springer
  - Han, J., Kamber, M. and Pei, J. Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, Burlington.

Time Series Forecasting

Time Series Forecasting

Time series refers to a sequence of data points collected, recorded, or otherwise measured over time intervals. Each data point in a time series is associated with a specific timestamp, allowing for the chronological organization of the data. Time series data is commonly used in various fields to analyze trends, patterns, and behaviors that evolve over time.
Time series forecasting - process of using historical time series data to make predictions about future values. The goal is to analyze patterns and trends within the data to build models that can then be used to forecast future points in the time series.

- Anything that is observed sequentially over time is a time series
- Forecasting aims to estimate how the sequence of observations will continue into the future
- Forecasting can be accompanied with indication of uncertainty



Types of Forecasting

Explanatory model: predictor variables capture reasons of the change[1]. The error term accounts for missing variables;

Electricity Demand = f(current temperature, strength of economy, population,time of day, day of week, error).

Typical models include classifiers and regression.

Timeseries model: prediction of the future is based on past values of a variable, but not on external variables that may affect the system.

Electricity Demand $ED_{t+1} = f(ED_t, ED_{t-1}, ED_{t-2}, ED_{t-3}, \ldots, \text{error})$

Mixed Models:

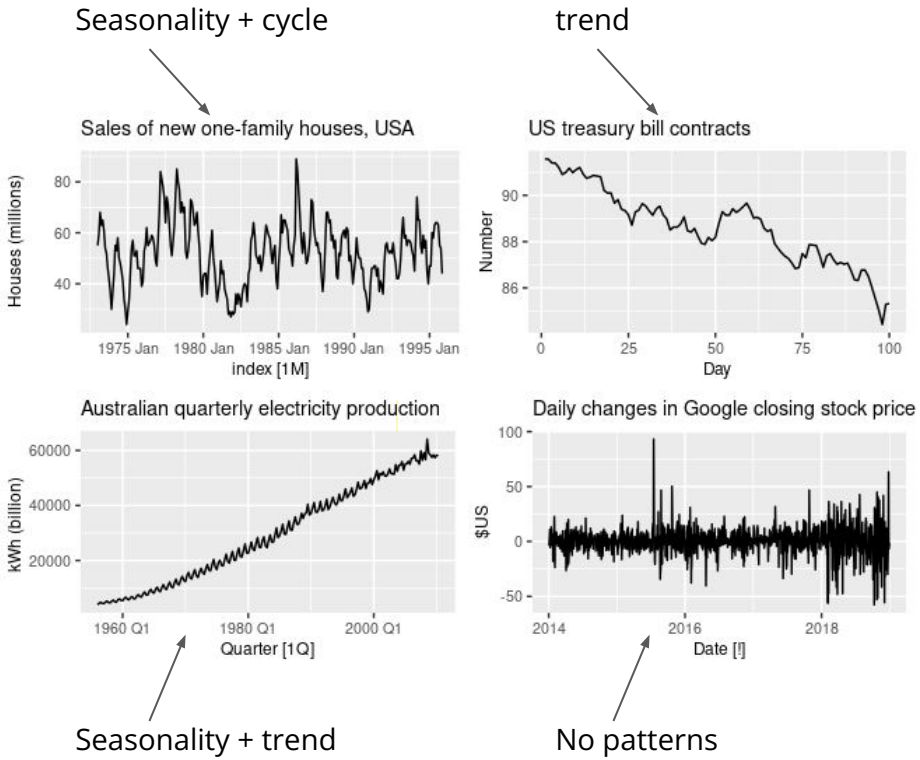
$ED_{t+1} = f$ (ED_t, current temperature, time of day, day of week, error).

[1] https://www.sciencedirect.com/science/article/pii/S136481521730542X

Time Series Components

A time series may consist of these components:

- **Trend:** long-term increase or decrease in the data.
- **Seasonality:** when a time series is affected by seasonal factors such as the time of the year or the day of the week. Seasonality is always of a fixed and known period.
- **Cycles:** similar to seasonality, yet lacks a fixed frequency. These fluctuations are usually due to economic conditions, and are often related to the "business cycle." The duration of these fluctuations is usually at least 2 years.



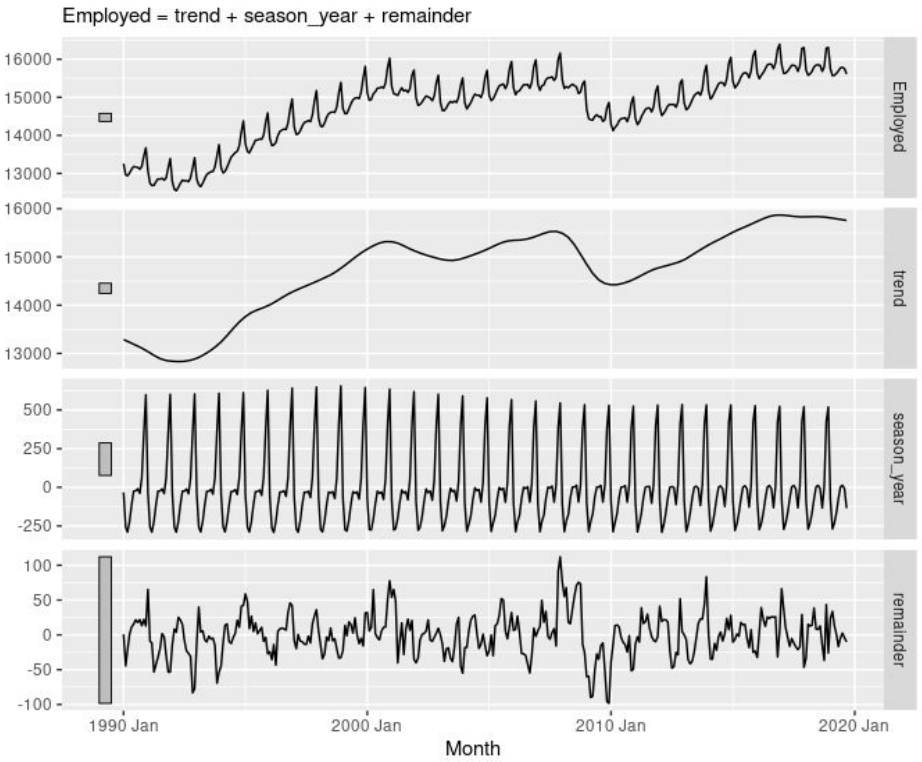
Time Series Decomposition

- We usually combine the trend and cycle into a single **trend-cycle** component (often just called the **trend** for simplicity)
- The seasonal components may slowly change over time
- The remainder component is what is left over when the seasonal and trend-cycle components have been subtracted from the data.
- additive decomposition

$$y_t = S_t + T_t + R_t$$

- multiplicative decomposition

$$y_t = S_t \times T_t \times R_t.$$



Adjustments Before Decomposition

- Calendar adjustments
  - Sum of monthly sales => daily sales

    Not all months have the same number of days/work days
- Population adjustments
  - => per-capita
- Inflation adjustment
  - If z_t denotes the price index and y_t denotes the original house price in year t, then

$$x_t = y_t / z_t * z_{2000}$$

Moving Averages to Estimate the Trend-Cycle

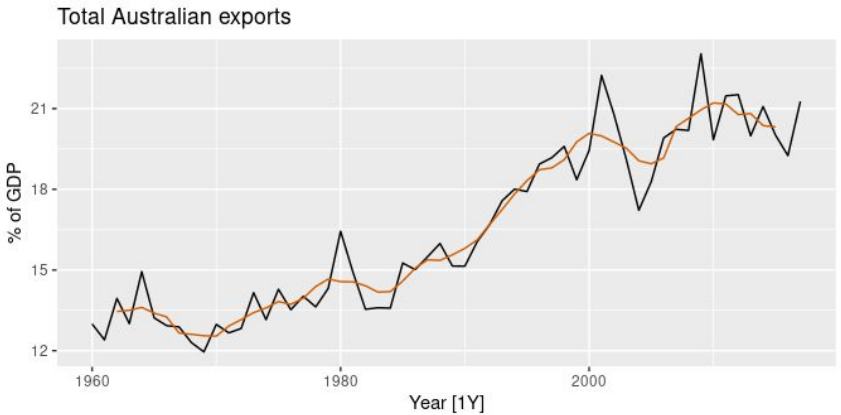
$$\hat{T}_t = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j},$$

- m=2k+1, k is number of neighbours
- Every value is replaced by the average of itself and its k nearest neighbourhood
- Results in a smooth trend-cycle component
- Ex: K=2, m=5, the result is called 5-MA
- Averaging is applied using a sliding window

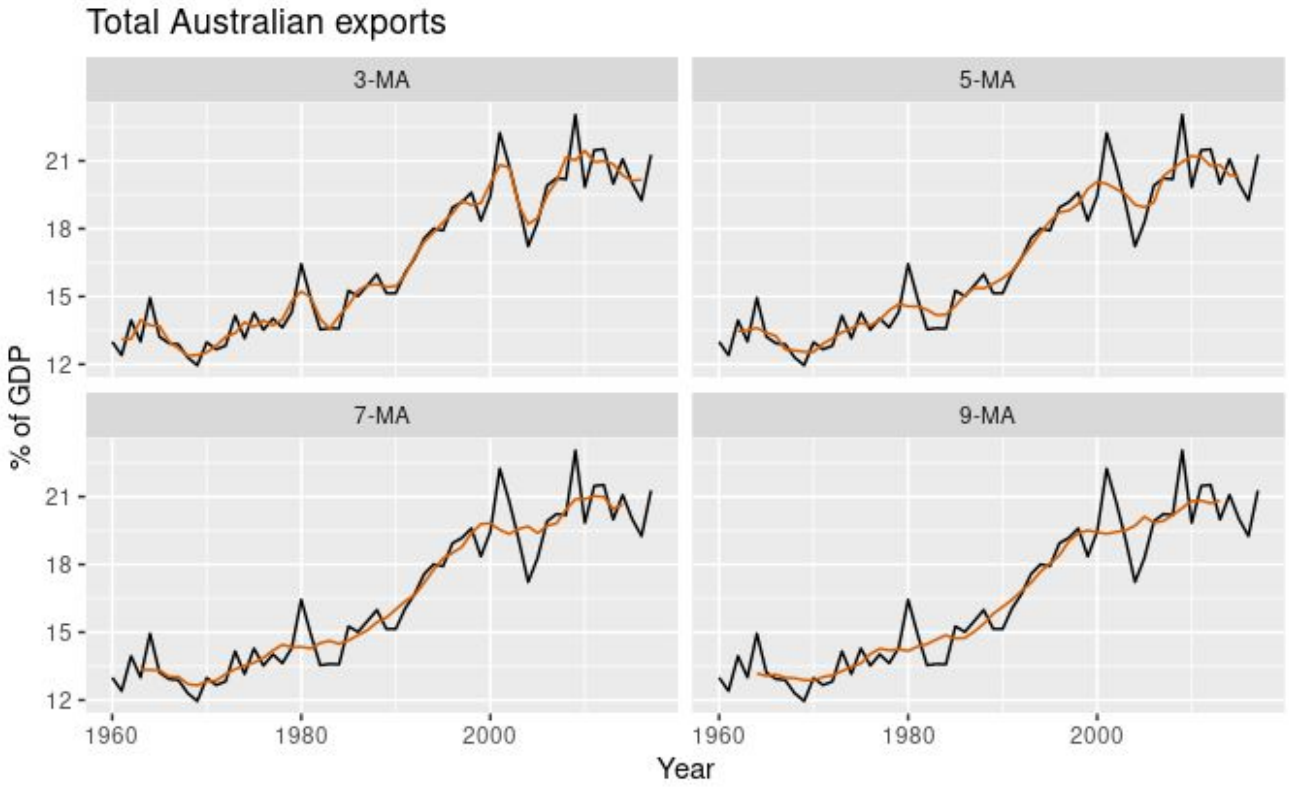
Moving Averages to Estimate the Trend-Cycle

Table 3.1: Annual Australian exports of goods and services: 1960–2017.

Year	Exports	5-MA
1960	12.99	
1961	12.40	
1962	13.94	13.46
1963	13.01	13.50
1964	14.94	13.61
1965	13.22	13.40
1966	12.93	13.25
1967	12.88	12.66
...	...	...
2010	19.84	21.21
2011	21.47	21.17
2012	21.52	20.78
2013	19.99	20.81
2014	21.08	20.37
2015	20.01	20.32
2016	19.25	
2017	21.27	



Moving Averages to Estimate the Trend-Cycle



Moving Averages of Moving Averages

Quarter	Beer	4-MA	2x4-MA
1992 Q1	443.00		
1992 Q2	410.00	451.25	
1992 Q3	420.00	448.75	450.00
1992 Q4	532.00	451.50	450.12
1993 Q1	433.00	449.00	450.25
1993 Q2	421.00	444.00	446.50
...	...	...	...
2009 Q1	415.00	430.00	428.88
2009 Q2	398.00	430.00	430.00
2009 Q3	419.00	429.75	429.88
2009 Q4	488.00	423.75	426.75
2010 Q1	414.00		
2010 Q2	374.00		

$$T_t = \tfrac{1}{2}\left[\tfrac{1}{4}(y_{t-2}+y_{t-1}+y_t+y_{t+1}) + \tfrac{1}{4}(y_{t-1}+y_t+y_{t+1}+y_{t+2})\right] = \tfrac{1}{8}y_{t-2} + \tfrac{1}{4}y_{t-1} + \tfrac{1}{4}y_t + \tfrac{1}{4}y_{t+1} + \tfrac{1}{8}y_{t+2}$$

Common for quarterly data

Moving Averages of Moving Averages

- Odd-order moving averages are symmetric
- Even-order MA are not symmetric. To make symmetric, we compute MA of MA
- The choice depends on the seasonality of data, e.g.,
  - Quarterly data with annual seasonality => 2 x 4-MA
  - Monthly data with annual seasonality ?
  - Daily data with weekly seasonality ?
- The goal is to average out seasonal variations, so that the smoothed timeseries has only the trend-cycle component.
- Other choices for the order of the MA will usually result in trend-cycle estimates being contaminated by the seasonality in the data.

how to even use garne ki odd kati order ko use garne?

Classical Additive Decomposition

Step 1

Compute the trend-cycle component T_t using a 2×m-MA or m-MA, for even-odd seasonality respectively.

Step 2

Calculate the detrended series: $y_t - T_t$

Step 3

Seasonal component for each season= average of detrended values for that season in all data

Adjust the seasonal components to ensure that they add to zero

Concat all seasonal components to form S_t

Step 4

The remainder component is $R_t = y_t - T_t - S_t$

Classical Decomposition

- Multiplicative decomposition works in a similar way
- The estimate of the trend-cycle is unavailable for the first few and last few observations.

  How many ?
- Assumes that the seasonal components repeats unchanged
- Tends to over smooth rapid rises/falls
- No special handling of unusual values, e.g., covid effect on air transport
- STL is currently commonly used:

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. J. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, *6*(1), 3–33. http://bit.ly/stl1990

Simple Forecasting

- Mean Method: all future data are equal to the average of the history

$$y_{T+h} = (y_1 + \cdots + y_T) / T$$

- Naive Method: all future data are equal to last observed value

$$y_{T+h} = y_T$$

- Seasonal Naive: each forecast is equal to the last observed value in same season
- Drift Method based on the assumption that the historical trend will continue into the future

$$y_{T+h} = y_T + h \left(\frac{y_T - y_1}{T - 1} \right)$$

variation of naive method
allows forecast to increase or decrease over time
where amount of change over time (called drift) is set to average change seen in historical data

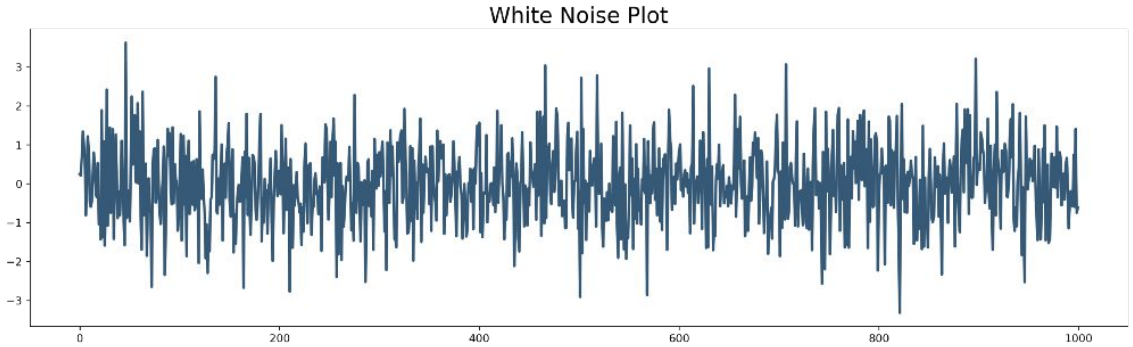
Forecasting With Decomposition

- Seasonally adjusted component $A_t = T_t + R_t$
- To forecast a decomposed time series, we separately forecast the seasonal component, S_t, and the seasonally adjusted component A_t
- The seasonal component is simply repeated
- The seasonally adjusted component is forecast using any method, e.g., drift

Auto Regression & Moving Average Forecasting

Stationary Timeseries

- Time series can be either stationary or nonstationary
- White noise is the strongest form of stationarity: zero mean, constant variance, and zero covariance between series values separated by a fixed lag.
- A strictly stationary: the probabilistic distribution of the values in any time interval [a, b] is identical to that in the shifted interval [a + h, b + h] for any value of the time shift h.



https://towardsdatascience.com/time-series-from-scratch-white-noise-and-random-walk-5c96270514d3

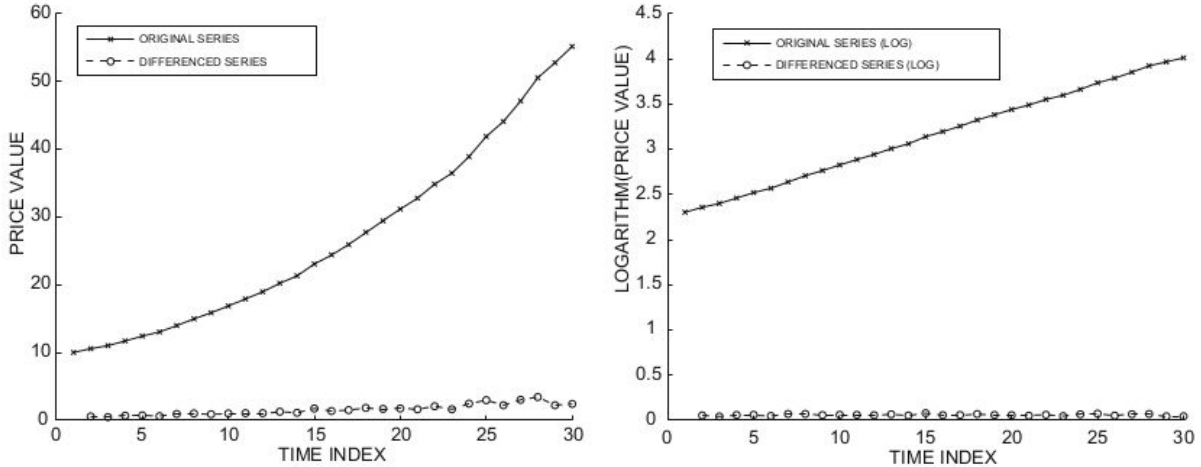
Time series can be either stationary or non stationary
Stationary timeseries is the one whose statistical properties like mean and variance does not change with time at which the series is observed.
Series with trend and seasonality is not stationary
White noise is stationary
Stationary Timeseires has constant mean and variance and no trend and seasonality

Stationary Timeseries

- In practice, timeseries are nonstationary.
- Forecasting can be better performed over stationary timeseries, since the statistical parameters (mean, variance, etc) do not vary over different time windows, and hence can be used in generating accurate forecasts.
- Therefore, it is often advantageous to convert nonstationary series to stationary ones before forecasting.
- After forecasting on the stationary series, the predicted values are transformed back to the original representation, using the inverse transformation.

Differencing - Converting Time Series to Stationary

- In differencing, the time series value y_i is replaced by the difference between it and the previous value. $y'_i = y_i - y_{i-1}$
- Differencing can be combined with log-scale



(a) Unscaled series (b) Logarithmic scaling

Differencing - Converting Time Series to Stationary

- Higher order differencing can be used to achieve stationarity in second order changes.
- Therefore, the higher order differenced value y_i is defined as follows:

$$y''_i = y'_i - y'_{i-1}$$

$$= y_i - 2 * y_{i-1} + y_{i-2}$$

- If the series is stationary after differencing, then an appropriate model for the data is:

$$y_{i+1} = y_i + c + e_{i+1}$$

- Here, e_{i+1} corresponds to white noise with zero mean, c is a non-zero constant that accounts for the drift
- It is rare to use differences beyond the second order.

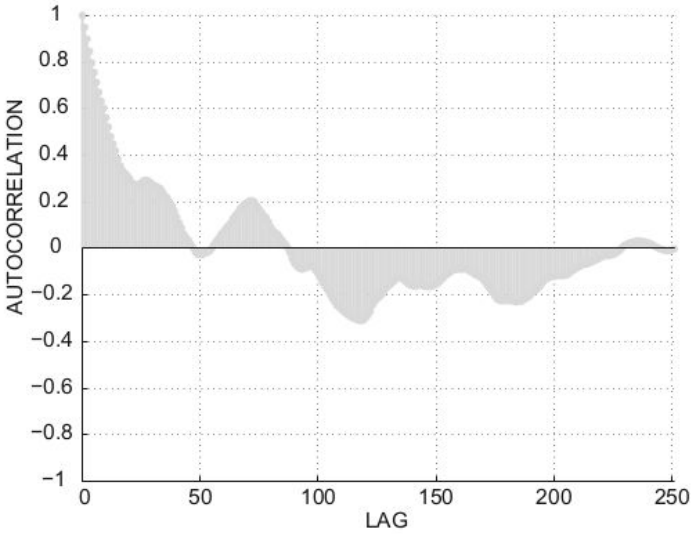
Autoregressive Models - Autocorrelation

- Use autocorrelation to predict future time series values (univariate forecasting)
- Autocorrelations represent the correlations between adjacently located timestamps in a series. Adjacency is defined in terms of *lag L*, e.g. Pearson coefficient,

$$\text{Autocorrelation}(L) = \frac{\text{Covariance}_t(y_t, y_{t+L})}{\text{Variance}_t(y_t)}$$

- Autocorrelation lies in the range [−1, 1]
- Typically positive for small values of L, and gradually drops, because nearby values in a timeseries tend to be similar.

Autocorrolation Function (ACF) plot



(a) IBM stock

In autoregression model, variable of interest is forecasted using linear combination of past values of that variable
The term autoregression refers to regression of variable against itself

Autoregressive Models AR(p)

- In the autoregressive model, the value of y_t at time t is defined as a linear combination of the values in the immediately preceding window of length p.

$$y_t = \sum_{i=1}^{p} a_i \cdot y_{t-i} + c + \epsilon_t$$

- A model that uses the preceding window of length p is referred to as an AR(p) model
- The parameters a_i, c are learned from the training data.

Autoregressive Models AR(p)

- In the autoregressive model, the value of y t at time t is defined as a linear combination of the values in the immediately preceding window of length p.

$$y_t = \sum_{i=1}^{p} a_i \cdot y_{t-i} + c + \epsilon_t$$

- A model that uses the preceding window of length p is referred to as an AR(p) model
- The parameters a_i, c are learned from the training data.
- The effectiveness of the forecasting model may be quantified by using the noise level in the estimated coefficients, for instance, it is desirable that the coefficient of determination R^2 is close to 1.

$$R^2 = 1 - \frac{\text{Mean}_t(\epsilon_t^2)}{\text{Variance}_t(y_t)}$$

past forecast error

Moving Average Models MA(q)

- The autoregressive models does not always explain all the variations, specially the unexpected component of the variations (shocks)
- This component can be captured with the use of a moving average model (MA)
- The autoregressive model can therefore be made more robust by combining it with an MA, so called ARMA model.
- The MA model is defined as $$y_t = \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + c + \epsilon_t$$
- $\mathcal{E}_{t-1}$ is a deviation from a predicted value, which can be viewed as white noise, or a shock
- AR relates the current value to the previous series values, while MA relates the current value to the previous history of deviations from forecasts.
- Series values cannot be predicted in terms of the shocks only. It needs to be added to the autocorrelations.

ARMA(p, q), ARIMA(p,d,q)

- The ARMA model is defined as $y_t = \sum_{i=1}^{p} a_i \cdot y_{t-i} + \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + c + \epsilon_t$
- Recall that AR & MA can only work if the series is stationary
- For non-stationary series, we need to add differencing
- Combining differencing with ARMA leads to ARIMA *autoregressive integrated moving average model*
- If the order of the differencing is *d*, then this model is referred to as the ARIMA(p, d, q), e.g., when d=1

$$y'_t = \sum_{i=1}^{p} a_i \cdot y'_{t-i} + \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + c + \epsilon_t$$

- ARIMA is thus the most general model for timeseries that can be made stationary.
- The predictors consist of lags of the dependent variable and/or lags of the forecast errors

Identifying p, d, q is Challenging

- More guidelines for your project here:

  https://people.duke.edu/~rnau/411arim2.htm

  https://people.duke.edu/~rnau/411arim3.htm

Forecasting at scale - the Prophet Model

- Recent model (and opensource) from Facebook, 2019
    - https://peerj.com/preprints/3190/
- The main design goals are: (1) ease of use, (2) speed
- At its core, the Prophet procedure is an additive regression model with four main components:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t.$$

    - g(t): a piecewise linear or logistic growth curve trend. Prophet automatically detects changes in trends by selecting changepoints from the data.
    - s(t): represents periodic changes (e.g., weekly and yearly seasonality), modeled using Fourier series.
    - h(t): represents the effects of holidays which occur on potentially irregular schedules over one or more days. The user provides the list of important holidays. Holiday list can be manually provided by user
    - ε_t: the error term represents the remainder which is not accommodated by the model, assumed to be normally distributed.

Credits & Readings

The slides of this lecture use material from

- CH14.3, Charu C.Aggarwal. Data Mining The Textbook, Springer.
- Robert Nau, Statistical forecasting: notes on regression and time series analysis

  https://people.duke.edu/~rnau/411arim.htm

- Rob J Hyndman and George Athanasopoulos. Forecasting: Principles and Practice (3rd ed)

  https://otexts.com/fpp3/

- The prophet model: https://peerj.com/preprints/3190/

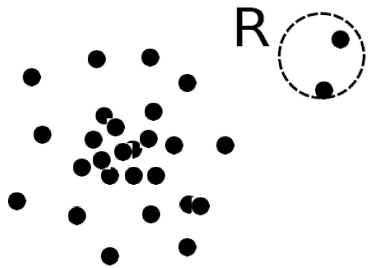
Outlier Mining

Noise refers to random variations or irregularities in data that do not carry meaningful information.

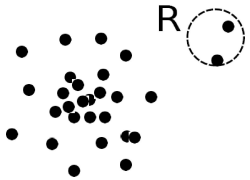
What Are Outliers?

noise - random

- Outlier: A data object that **deviates significantly** from the normal objects as if it were **generated by a different mechanism**
  - Ex.: Unusual credit card purchase, sports: Mo Salah, …
- Outliers are different from the noise data
  - Noise is random error or variance in a measured variable
  - Noise should be removed before outlier detection
- Outliers are interesting: They violate the mechanism that generates the normal data
- Outlier detection vs. novelty detection: early stage, outlier; but later merged into the model
- Applications:
  - Credit card fraud detection
  - Telecom fraud detection
  - Medical analysis



Types of Outliers (I)



Global Outlier

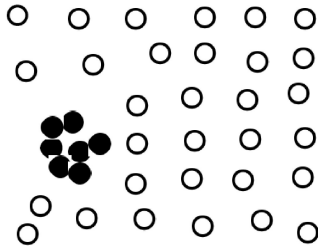
- Three kinds: *global*, *contextual* and *collective* outliers
- **Global outlier** (or point anomaly) significantly deviates from the rest of the data set
  - ex. Very high speed
  - How to find ? Assuming data is clustered, find an appropriate measurement of deviation.

- **Contextual outlier** (or *conditional outlier*) deviates significantly based on a selected context
  - Is 30^{o} C an outlier ? Is 40km/h on the highway and outlier ?
  - Attributes of data objects should be divided into two groups
    - Contextual attributes: defines the context, e.g., time & location
    - Behavioral attributes: characteristics of the object, used in outlier evaluation, e.g., temperature
  - Issue: How to define or formulate meaningful context?

Types of Outliers (II)

Collective Outliers, a subset of data objects *collectively* deviate significantly from the whole data set, even if the individual data objects may not be outliers

e.g., intrusion detection: (checking the personal data of one client vs. of 100 clients), one car deviating from the shortest path vs. 100 cars doing so.



Collective Outlier

- **Detection** of collective outliers
  - Consider not only behavior of individual objects, but also that of groups of objects
  - Need to have the background knowledge on the relationship among data objects, such as a distance or similarity measure on objects.
- A data set may have multiple types of outlier

Challenges of Outlier Detection

- Modeling normal objects and outliers properly
  - Hard to enumerate all possible normal behaviors in an application
  - The border between normal and outlier objects is often a gray area
- Application-specific outlier detection
  - Choice of distance measure among objects and the model of relationship among objects are often application-dependent
  - E.g., clinic data: a small deviation could be an outlier; while in marketing analysis, larger fluctuations
- Handling noise in outlier detection
  - Noise may distort the normal objects and blur the distinction between normal objects and outliers. The detection can confuse noise and outliers.
- Understandability
  - Understand why these are outliers: Justification of the detection
  - Specify the degree of an outlier: the unlikelihood of the object being generated by a normal mechanism

Outlier Detection I: Supervised Methods

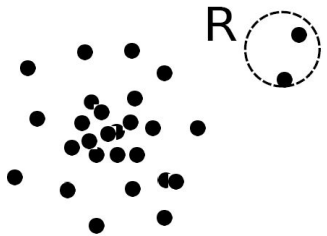
- Two ways to categorize outlier detection methods:
  - Based on whether user-labeled examples of outliers can be obtained:
    - Supervised, semi-supervised vs. unsupervised methods
  - Based on assumptions about normal data and outliers:
    - Statistical, proximity-based, and clustering-based methods
- Outlier Detection I: **Supervised Methods**
  - Modeling outlier detection as a classification problem
    - Samples examined by domain experts used for training & testing
  - One class classification:
    - Model normal objects & report those not matching the model as outliers, or
  - Challenges
    - Imbalanced classes, i.e., outliers are rare
    - Catch as many outliers as possible, i.e., recall is more important than accuracy

Outlier Detection II: Unsupervised Methods

- Assume the normal objects are somewhat *clustered* into multiple groups, each having some distinct features
- An outlier is expected to be far away from any groups of normal objects
- Weakness: Cannot detect collective outlier effectively
  - Normal objects may not share any strong patterns, but the collective outliers may share high similarity in a small area

- Many clustering methods can be adapted for unsupervised methods
  - Find clusters, then outliers: not belonging to any cluster
  - Problem 1: Hard to distinguish noise from outliers
  - Problem 2: Costly since first clustering: but far less outliers than normal objects

Outlier Detection (2): Proximity-Based Methods



- An object is an outlier if the nearest neighbors of the object are far away
- Model the proximity of an object using its 3 nearest neighbors
  - Objects in region R are substantially different from other objects in the data set.
  - Thus the objects in R are outliers

- The effectiveness of proximity-based methods highly relies on the proximity measure.
- Often have a difficulty in finding a group of outliers which stay close to each other
- Two major types of proximity-based outlier detection
  - Distance-based vs. density-based
  - Distance-based outlier detection: An object o is an outlier if its k-nearest neighborhood happens at a relatively large distance
  - Density-based outlier detection: An object o is an outlier if its density is relatively much lower than that of its neighbors

Distance-Based Outlier Detection

Formally, let r $(r \geq 0)$ be a *distance threshold* and π $(0 < \pi \leq 1)$ be a fraction threshold. An object, $\boldsymbol{o}$, is a $DB(r, \pi)$-**outlier** if

$$\frac{\|\{\boldsymbol{o}' | dist(\boldsymbol{o}, \boldsymbol{o}') \leq r\}\|}{\|D\|} \leq \pi, \tag{12.10}$$

where $dist(\cdot, \cdot)$ is a distance measure.

Equivalently, we can determine whether an object, $\boldsymbol{o}$, is a $DB(r, \pi)$-outlier by checking the distance between $\boldsymbol{o}$ and its k-nearest neighbor, $\boldsymbol{o}_k$, where $k = \lceil \pi \|D\| \rceil$. Object $\boldsymbol{o}$ is an outlier if $dist(\boldsymbol{o}, \boldsymbol{o}_k) > r$, because in such a case, there are fewer than k objects except for $\boldsymbol{o}$ that are in the r-neighborhood of $\boldsymbol{o}$.

Distance-Based Outlier Detection

Algorithm: Distance-based outlier detection.

Input:

- a set of objects $D = \{o_1, \ldots, o_n\}$, threshold r ($r > 0$) and π ($0 < \pi \leq 1$);

Output: $DB(r, \pi)$ outliers in D.

Method:

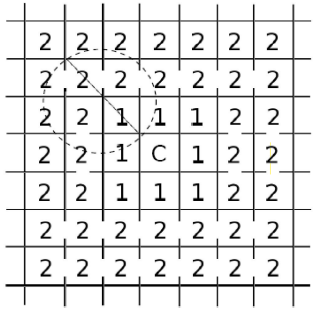
```
for i = 1 to n do
    count ← 0
    for j = 1 to n do
        if i ≠ j and dist(o_i, o_j) ≤ r then
            count ← count + 1
            if count ≥ π · n then
                exit {o_i cannot be a DB(r, π) outlier}
            endif
        endif
    endfor
    print o_i {o_i is a DB(r, π) outlier according to (Eq. 12.10)}
endfor;
```

Distance-Based Outlier: A Grid-Based Method

- Why efficiency is still a concern? When the complete set of objects cannot be held into main memory, cost I/O swapping
- The major cost: (1) each object tests against the whole data set, why not only its close neighbor? (2) check objects one by one, why not group by group?
- Grid-based method (CELL): Data space is partitioned into a multi-D grid. Each cell is a hyper cube with diagonal length r/2

Level-1 cell pruning rule: Based on the level-1 cell property, if $a + b_1 > \lceil \pi n \rceil$, then every object $\boldsymbol{o}$ in C is not a $DB(r, \pi)$-outlier because all those objects in C and the level-1 cells are in the r-neighborhood of $\boldsymbol{o}$, and there are at least $\lceil \pi n \rceil$ such neighbors.

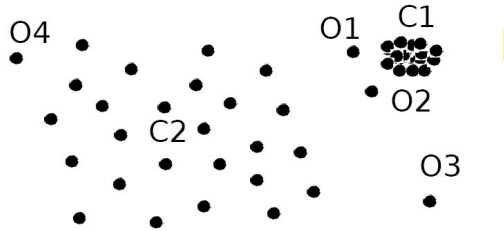
Level-2 cell pruning rule: Based on the level-2 cell property, if $a + b_1 + b_2 < \lceil \pi n \rceil + 1$, then all objects in C are $DB(r, \pi)$-outliers because each of their r-neighborhoods has less than $\lceil \pi n \rceil$ other objects.



Density-Based Outlier Detection

Difference between distance and density based outlier detection??

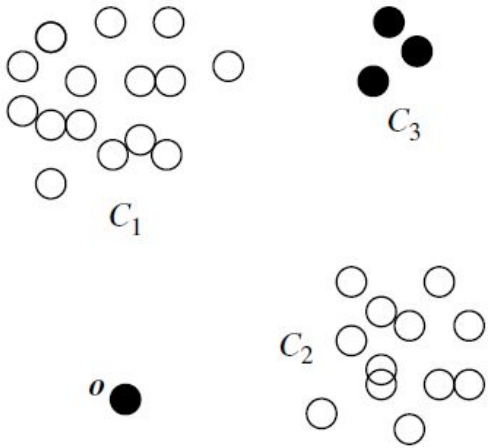
- Contextual outliers: Outliers comparing to their local neighborhoods, instead of the global data distribution
- Distance-based outlier detection won't work.



- Intuition (density-based outlier detection): The density around an outlier object is significantly different from the density around its neighbors
- Method: Use the relative density of an object against its neighbors as the indicator of the degree of the object being outliers
- k-distance of an object o, $dist_k(o)$: distance between o and its k-th NN
- k-distance neighborhood of o, $N_k(o) >= \{o' |\ o' \text{ in } D, dist(o, o') \leq dist_k(o)\}$
  - $N_k(o)$ could be bigger than k since multiple objects may have identical distance to o

Clustering-Based Methods

- An object is an outlier if
  - (1) it does not belong to any cluster,
  - (2) there is a large distance between the object and its closest cluster ,
  - or (3) it belongs to a small or sparse cluster

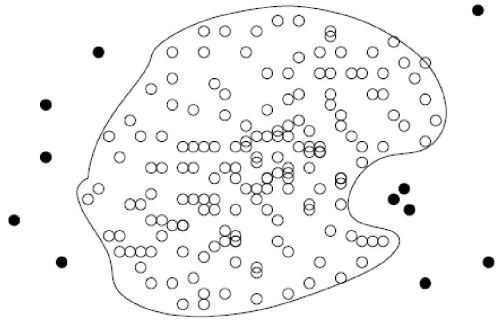


One-Class Model- classify instances into a single class or category. This means that the model is trained to recognize and assign instances to one specific class, and all other instances are considered as anomalies or outliers.
One-class models are often used in anomaly detection or novelty detection tasks, where the focus is on identifying instances that deviate from the norm rather than classifying them into multiple categories.

Two-Class Model- In a two-class model, the objective is to classify instances into one of two classes or categories. The most common scenario is binary classification, where the two classes are often labeled as positive and negative, or class 0 and class 1. Binary classification is used in a wide range of applications, such as spam detection (spam or not spam), disease diagnosis (diseased or not diseased), and many others.

Classification-Based Method I: One-Class Model

- Two-class classification
  Idea: Train a classification model that can distinguish "normal" data from outliers
  - Requires many abnormal samples.
  - Abnormal might not well cluster.



- One-class model: A classifier is built to describe only the normal class.
  - Learn the decision boundary of the normal class
  - Any samples that do not belong to the normal class (not within the decision boundary) are declared as outliers
  - Adv: can detect new outliers that may not appear close to any outlier objects in the training set
  - Extension: Normal objects may belong to multiple classes

Credits and Readings

- These slides, except when explicitly stated, use material from:
  - Charu C.Aggarwal. Data Mining The Textbook, Springer.
  - Data Mining: Concepts and Techniques, 3rd ed. Slides by Jiawei Han.

Association Pattern Mining - Frequent Itemset Mining

Problem

- A dataset of transactions
- One transaction is a set of items
- How to find interesting associations between items ?

Image from https://datacrunch.nl

Problem - Original Text

This paper introduces the problem of "mining" a large collection of basket data type transactions for association rules between sets of items with some minimum specified confidence, and presents an efficient algorithm for this purpose. An example of such an association rule is the statement that 90% of transactions that purchase bread and butter also purchase milk. The antecedent of this rule consists of bread and butter and the consequent consists of milk alone. The number 90% is the confidence factor of the rule.

Mining association rules between sets of items in large databases.
Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. *ACM SIGMOD 1993.*
Citations 9799, Downloads 26k
Last 12 Months 1,6k
Nov 8 2021

Application Domains

if *itemset1* then *itemset2* (degree of certainty)

- Market basket analysis
  - {Bread, Cheese, yogurt} => {milk, eggs}
- Medical diagnosis
  - {Coronavirus} => {fever, cough, shortness of breath}
- Power plant failure prediction [1].
- Text mining, Web log mining, Classification, ...

[1] Maik Reder, Nurseda Y. Yürüşen, Julio J. Melero, Data-driven learning framework for associating weather conditions and wind turbine failures, Reliability Engineering & System Safety, Volume 169, 2018.

Mining Association Rules - Solution Pipeline

Mining Association Rules - Solution Pipeline

Definitions

- The input is a database T that contains a set of n transactions, denoted by $T_1 \ldots T_n$
- Each transaction T_i is a subset drawn on the universe of items U
- Transactions can be represented in a binary format.
- Let U= {Bread, Butter, Cheese, Eggs, Milk, Yogurt}

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

Table from Charu C. Aggarwal, Data Mining The Textbook, 2015.

Definitions - Support

- Support of an itemset I is the fraction of the transactions in the database T that contain I as a subset

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

- Support({Milk, Yogurt})= 3/5= 0.6

Definitions - Support

how frequently the itemset appears in transaction

- Support of an itemset I is the fraction of the transactions in the database T that contain I as a subset

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

- Support({Milk, Yogurt})= 3/5= 0.6

Frequent Itemset Mining

- Given a database T, determine all itemsets I that have support of at least *minsup*

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

Find all frequent itemsets with minsup= 0.6

{Milk}, {Eggs}, {Yogurt},

{Milk, Yogurt}, {Milk, Eggs}

Apriori Property

Every subset of a frequent itemset is also frequent
Or
Every superset of a infrequent itemset is also infrequent

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

{Milk, Yogurt} is frequent => {Milk} is frequent, {Yogurt} is frequent

{Butter} is infrequent => {Butter, x} is infrequent

Apriori Property

Every subset of a frequent itemset is also frequent

Or

Every superset of a infrequent itemset is also infrequent

Itemset lattice:

Size= $2^{|U|}$

Location of maximal frequent itemsets ?

Any frequent itemset mining Algorithms conceptually maps to a (search) traversal of this lattice.

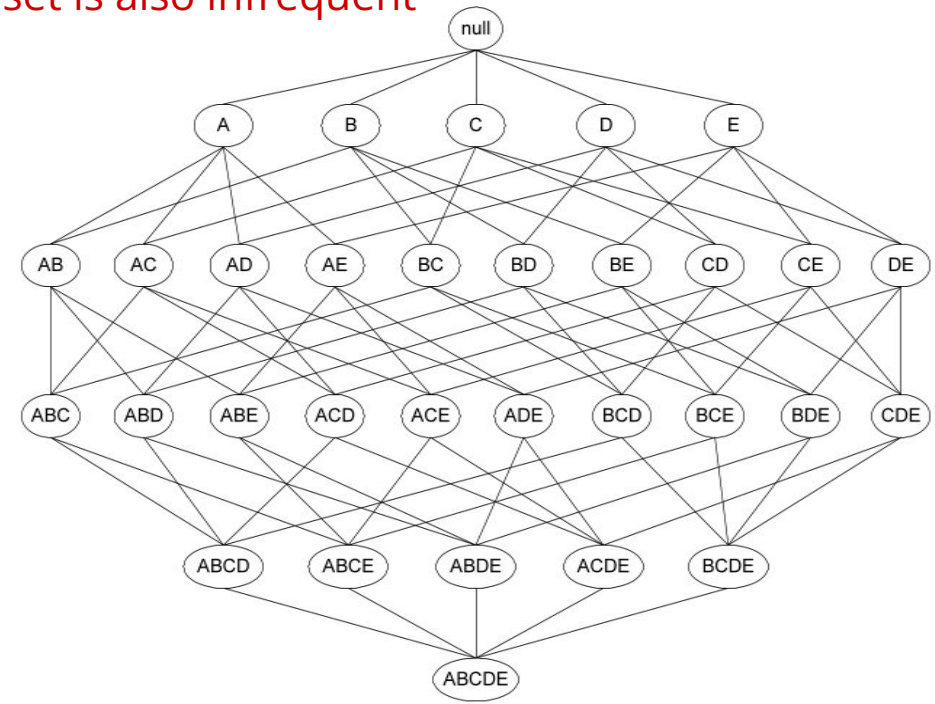


Figure from Han et al., Data Mining: Concepts and Techniques, 2012.

Apriori Property

Every subset of a frequent itemset is also frequent

Or

Every superset of a infrequent itemset is also infrequent

Itemset lattice:

Size= $2^{|U|}$

Location of maximal frequent itemsets ?

Any frequent itemset mining Algorithms conceptually maps to a (search) traversal of this lattice.

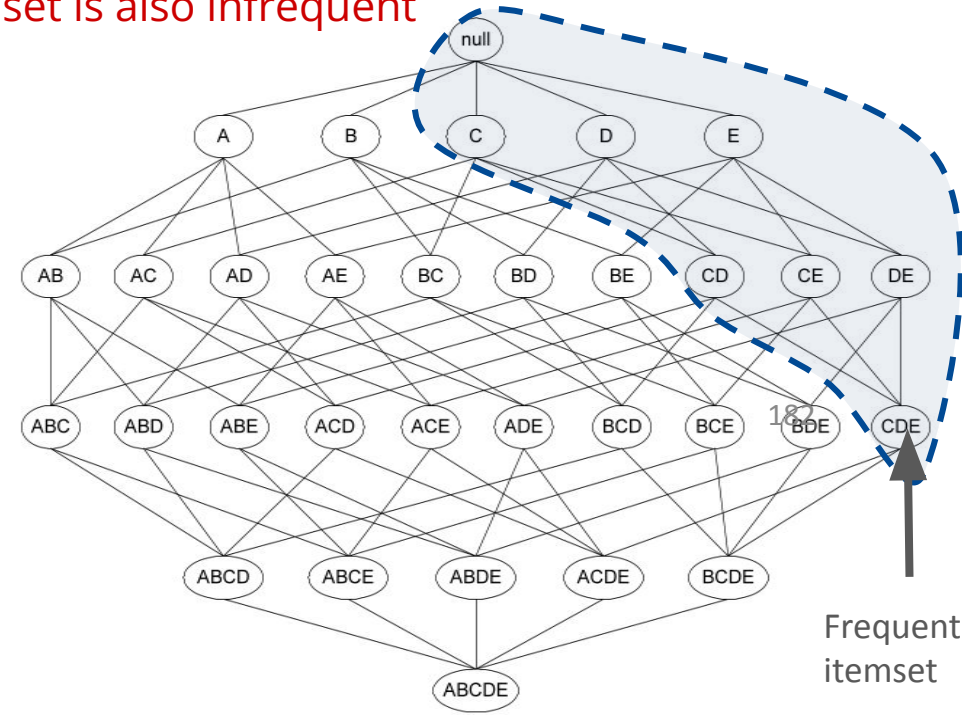


Figure from Han et al., Data Mining: Concepts and Techniques, 2012.

Apriori Property

Every subset of a frequent itemset is also frequent

Or

Every superset of a infrequent itemset is also infrequent

Itemset lattice:

Size= $2^{|U|}$

Location of maximal frequent itemsets ?

Any frequent itemset mining Algorithms conceptually maps to a (search) traversal of this lattice.

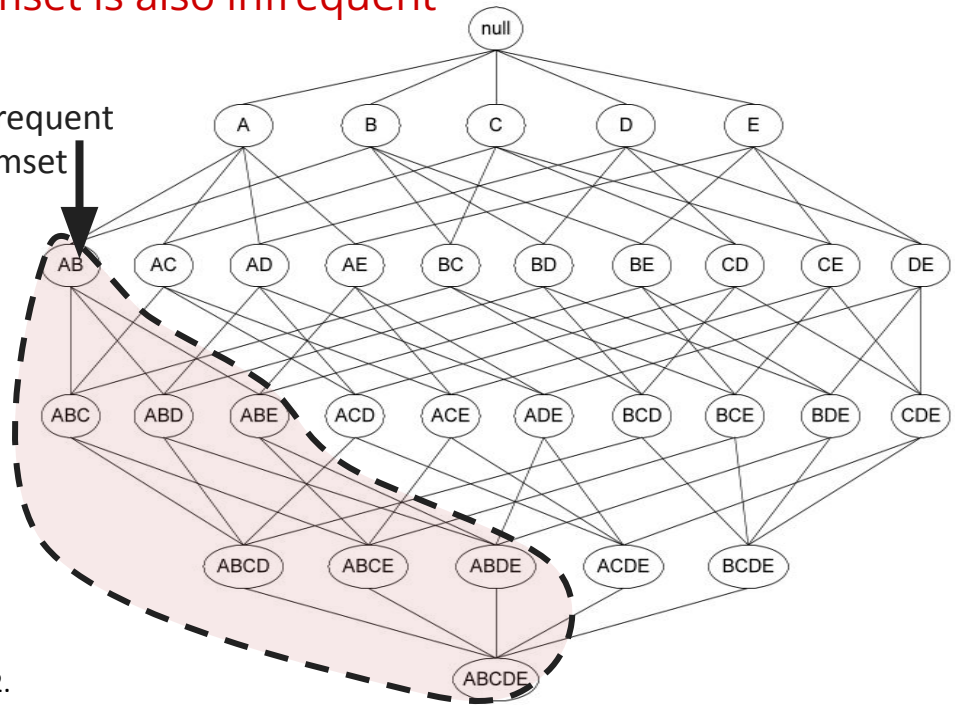


Figure from Han et al., Data Mining: Concepts and Techniques, 2012.

Association Rules

A frequent itemset does not necessarily imply that its items are associated.

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

{Milk, Yogurt} is frequent with Sup 0.6, but Milk does not imply Yogurt because 0.4 of the transactions have Milk but no Yogurt.

It indicates the likelihood that
an item Y will be purchased when item X is purchased.

Association Rules

$$conf(X \Rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)}.$$

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

Conf({Eggs, Milk} ⇒ {Yogurt})= Sup({Eggs, Milk,

Yogurt})/ Sup({Eggs, Milk}) = 0.4/0.6= 2/3

Association Rules

(Association Rules) Let X and Y be two sets of items. Then, the rule $X \Rightarrow Y$ is said to be an

association rule at a minsup and a minconf, if:

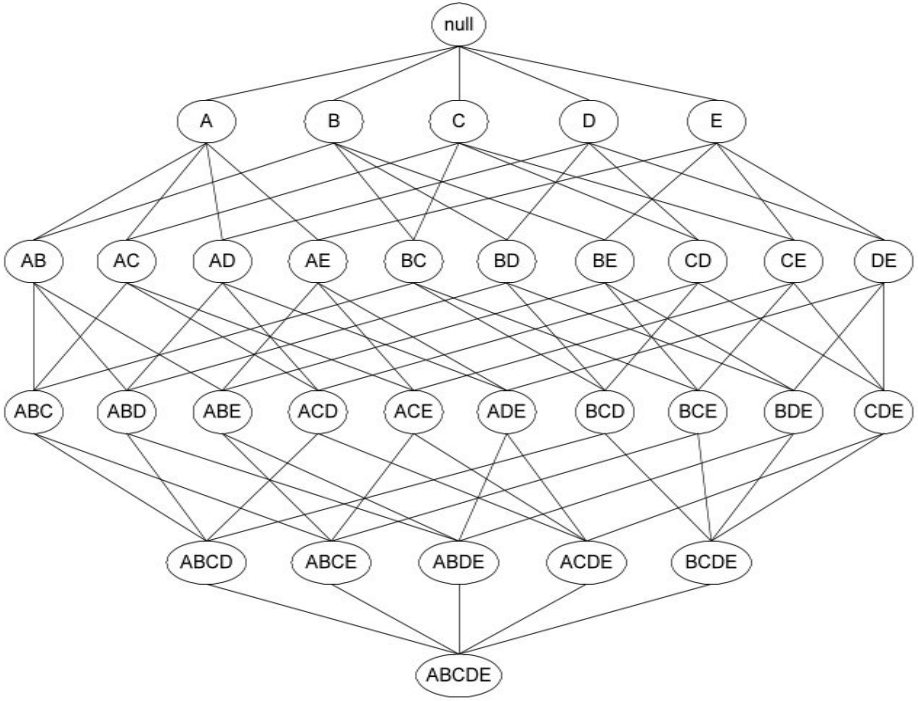
1. $Sup(X \cup Y) >= minsup$, and
2. $Conf(X \Rightarrow Y) >= minconf$.

How to mine for association rules given minsup, minconf ?

1. $F \leftarrow$ all frequent itemsets with sup >= minsup.
2. Using F, generate rules with conf >= minconf.

Mining Frequent Itemsets

- The bruteforce way is to try the whole lattice
- Complexity ? How to optimize ?
  - Pruning candidate itemsets (lattice nodes) using the downward closure property.
  - Counting the support of each candidate more efficiently.
  - Using compact data structures that support efficient counting.



Apriori Algorithm

minimum support count= 2

C - Candidate Itemset
F - Frequent Itemset

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

C1

Items et	#
A	6
B	7
C	6
D	2
E	2

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

Itemset	#
A, B, C	
A, B, D	
A, B, E	
A, C, D	
A, C, E	
B, C, D	
B, C, E	
B, D, E	

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

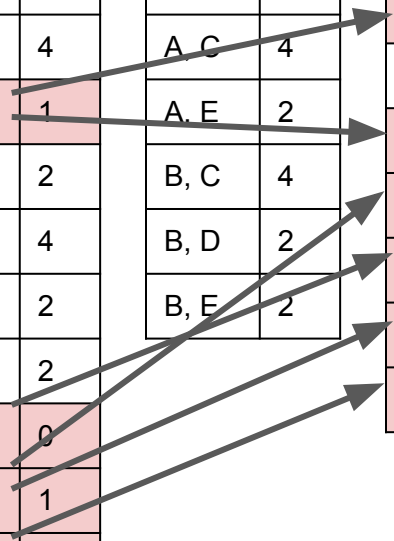
Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

Itemset	#
A, B, C	
A, B, **D**	
A, B, E	
A, C, **D**	
A, **C, E**	
B, **C, D**	
B, **C, E**	
B, **D, E**	



Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

Itemset	#
A, B, C	2
A, B, **D**	
A, B, E	2
A, C, **D**	
A, **C, E**	
B, **C, D**	
B, **C, E**	
B, **D, E**	

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

Itemset	#
A, B, C	2
A, B, **D**	
A, B, E	2
A, C, **D**	
A, **C, E**	
B, **C, D**	
B, **C, E**	
B, **D, E**	

F3

Itemset	#
A, B, C	2
A, B, E	2

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

Itemset	#
A, B, C	2
A, B, **D**	
A, B, E	2
A, C, **D**	
A, **C, E**	
B, **C, D**	
B, **C, E**	
B, **D, E**	

F3

Itemset	#
A, B, C	2
A, B, E	2

C4

Itemset	#
A, B, C, E	

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

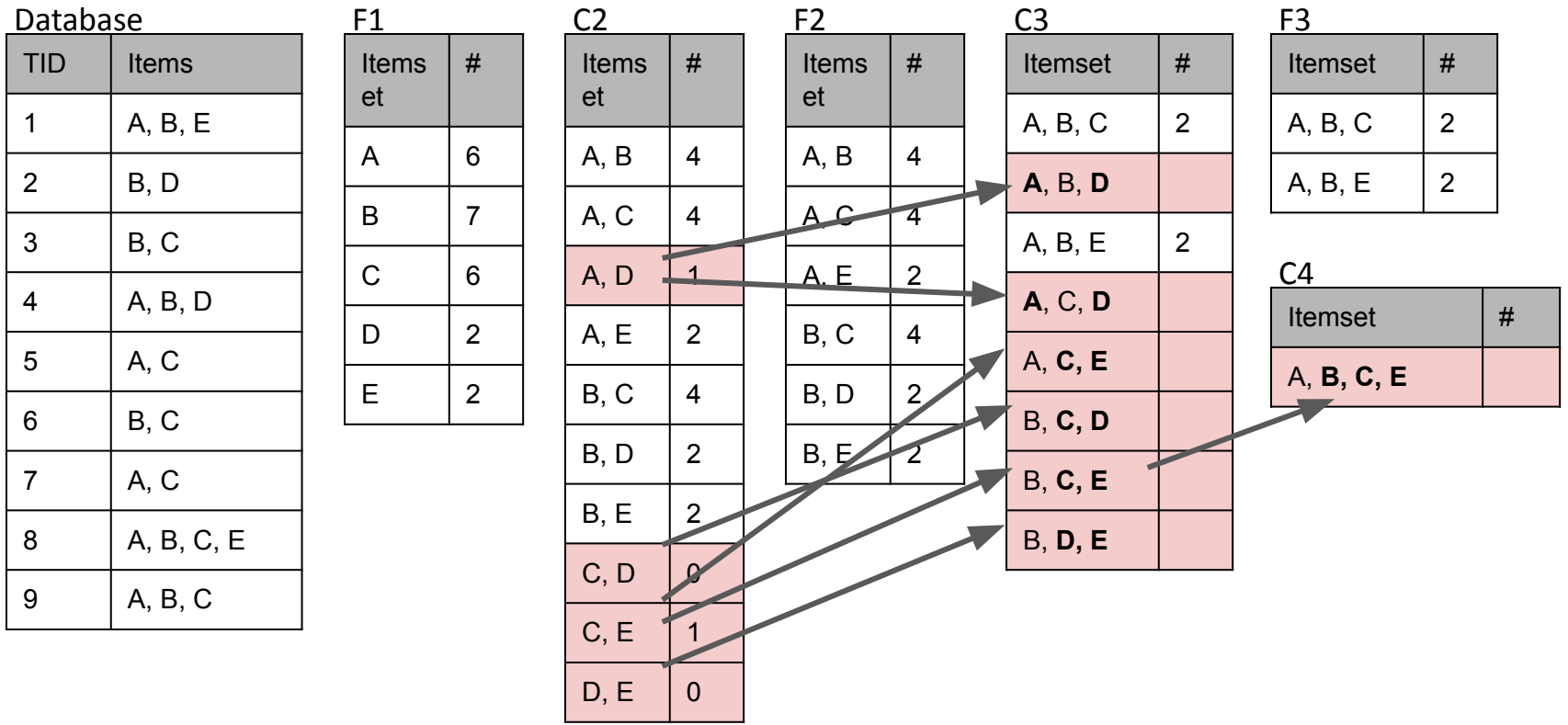
Itemset	#
A, B, C	2
A, B, **D**	
A, B, E	2
A, C, **D**	
A, **C, E**	
B, **C, D**	
B, **C, E**	
B, **D, E**	

F3

Itemset	#
A, B, C	2
A, B, E	2

C4

Itemset	#
A, **B, C, E**	



Apriori Algorithm

minimum support count= 2

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

F3

Itemset	#
A, B, C	2
A, B, E	2

Apriori Algorithm - Pseudocode

Algorithm *Apriori*(Transactions: $\mathcal{T}$, Minimum Support: *minsup*)
begin
  $k = 1$;
  $\mathcal{F}_1 = \{$ All Frequent 1-itemsets $\}$;
  while $\mathcal{F}_k$ is not empty **do begin**
    Generate $\mathcal{C}_{k+1}$ by joining itemset-pairs in $\mathcal{F}_k$;
    Prune itemsets from $\mathcal{C}_{k+1}$ that violate downward closure;
    Determine $\mathcal{F}_{k+1}$ by support counting on $(\mathcal{C}_{k+1}, \mathcal{T})$ and retaining
      itemsets from $\mathcal{C}_{k+1}$ with support at least *minsup*;
    $k = k + 1$;
  end;
  return$(\cup_{i=1}^{k} \mathcal{F}_i)$;
end

Figure from Charu C. Aggarwal, Data Mining The Textbook, 2015.

Limits of Apriori Algorithm

- Uses the apriori property.
  - nice property.
- General-to-specific traversal of the itemset lattice.
  - bi-directional traversal.
- Breadth-first search.
  - depth-first search.
- Generate-and-test strategy.
  - C2 is O(|U| Choose 2)
  - test is count.

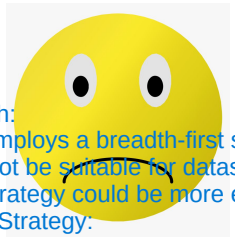
In the worst case, Apriori may generate a large number of candidate itemsets, especially in the early iterations of the algorithm.

In order to calculate support, we need to access database multiple times.

Need for Multiple Passes

Binary Representation:
Apriori often uses binary representation (presence or absence of an item) to represent transactions, which may not capture quantitative information about items.

Apriori Property Dependency:
Limitation: The algorithm relies on the Apriori property, which states that if an itemset is frequent, then all of its subsets must also be frequent. This property can lead to a large number of candidate itemsets being generated and tested, especially in cases where the dataset has a large number of items.
General-to-Specific Traversal:
Limitation: Apriori uses a general-to-specific traversal strategy of the itemset lattice. This means that the algorithm starts with smaller itemsets and gradually explores larger ones. While this approach is efficient in certain cases, it may miss interesting patterns that involve larger itemsets, especially if the support threshold is set too high.



Breadth-First Search:
Limitation: Apriori employs a breadth-first search strategy. While this helps in discovering frequent itemsets efficiently, it might not be suitable for datasets with a skewed distribution of itemset sizes. In such cases, a depth-first search strategy could be more effective.
Generate-and-Test Strategy:
Limitation: The algorithm follows a generate-and-test strategy, where candidate itemsets are generated and then tested for their frequency. This approach can be computationally expensive, especially when dealing with large datasets. Additionally, the size of the candidate itemsets grows rapidly, leading to a potentially large number of combinations to be tested.

Apriori + Tricks

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- *min_sup*, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ φ; k++) {
(3)       Ck = apriori_gen(Lk−1);
(4)       for each transaction t ∈ D { // scan D for counts
(5)           Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)           for each candidate c ∈ Ct
(7)               c.count++;
(8)       }
(9)       Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;
```

Apriori + Tricks

```
procedure apriori_gen(L_{k-1}:frequent (k-1)-itemsets)
(1)     for each itemset l_1 ∈ L_{k-1}
(2)         for each itemset l_2 ∈ L_{k-1}
(3)             if (l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2])
                    ∧... ∧ (l_1[k-2] = l_2[k-2]) ∧ (l_1[k-1] < l_2[k-1]) then {
(4)                 c = l_1 ⋈ l_2; // join step: generate candidates
(5)                 if has_infrequent_subset(c, L_{k-1}) then
(6)                     delete c; // prune step: remove unfruitful candidate
(7)                 else add c to C_k;
(8)             }
(9)     return C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
            L_{k-1}: frequent (k-1)-itemsets); // use prior knowledge
(1)     for each (k-1)-subset s of c
(2)         if s ∉ L_{k-1} then
(3)             return TRUE;
(4)     return FALSE;
```

More Apriori tricks

- Transaction reduction: A transaction that does not include any k-frequent itemsets, cannot contain (k+1)-frequent itemsets, and can thus be removed to make the counting faster.
- Partitioning: divide the database D into a set of disjoint partitions. Find all frequent itemsets in every partition (note that minsup remains the same). A local frequent itemset may not be frequent for the whole D, but a frequent itemset in D must be frequent in at least one partition. The improvement of partitioning is that it allows for parallel processing, and that the partition size can be designed to fit in memory.
- Sampling: perform Apriori on a random sample (say 10%). Rule accuracy is not guaranteed, but efficiency is improved. Certain applications can tolerate this.

Mining Association Rules: Step2

tid	Set of items	Binary representation
1	$\{Bread, Butter, Milk\}$	110010
2	$\{Eggs, Milk, Yogurt\}$	000111
3	$\{Bread, Cheese, Eggs, Milk\}$	101110
4	$\{Eggs, Milk, Yogurt\}$	000111
5	$\{Cheese, Milk, Yogurt\}$	001011

- Minsup= 0.4, minconf= 0.8.
- Frequent itemsets: {Bread, Milk}, {Milk, Cheese}, {Eggs, Milk}, {Eggs, Yogurt}, {Eggs, Milk, Yogurt}.
  - Conf(Bread □ Milk)= 0.4/0.4= 1
  - Conf(Milk □Bread)= 0.4/1= 0.4
  - …
  - Conf({Eggs, Milk} □ Yogurt)= 0.4/0.6= 2/3

Mining Association Rules: Step2

- Once the frequent itemsets have been found, foreach of them (let be called *l*):
  - Generate all nonempty subsets of *l*.
  - For every subset *s* of *l*, Output the rule *s* -> *(l – s)* if its confidence >= minconf.

Recommended Readings

- Aggarwal, Charu C. Data Mining: The Textbook. Springer (2015).
- Han, Jiawei and Kamber, Micheline and Pei, Jian: Data Mining: Concepts and Techniques (3rd. ed.). Morgan Kaufmann Publishers Inc. (2011).

Association Pattern Mining - Part 2

But…

- Indeed, Apriori significantly reduces the size of candidate sets, leading to good performance gain.
- However, it can suffer from a nontrivial costs:
  - *It may still need to generate a huge number of candidate sets.* For example, if there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7 candidate 2-itemsets.
- *Can we design a method that mines the complete set of frequent itemsets without such a costly candidate generation process?*

FP-growth (finding frequent itemsets without candidate generation)

Frequent Pattern Growth is a method that:

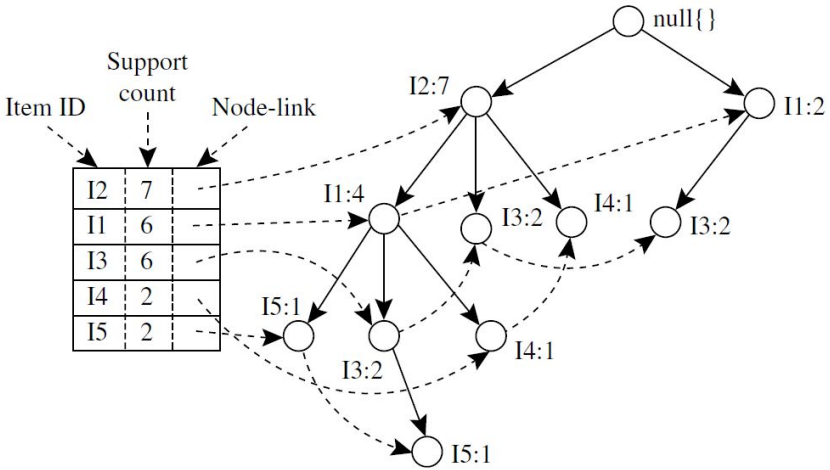
1. Transforms the database into a compressed data structure FP-tree, which retains frequent itemsets information.
2. Mines the FP-tree for frequent itemsets by:
    a. Divide it into a set of conditional databases (called conditional pattern base), each associated with one frequent item.
    b. Mines each of these patterns separately to generate frequent itemsets, without the need to re-count the original database.

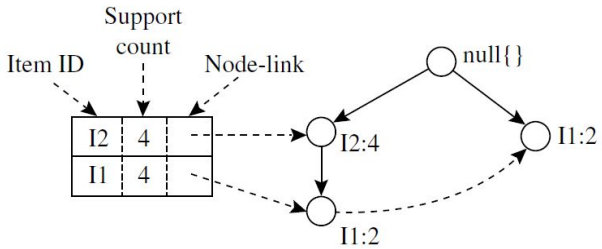
FP-growth (finding frequent itemsets without candidate generation)

Transactional Data for an *AllElectronics* Branch

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

FP-growth





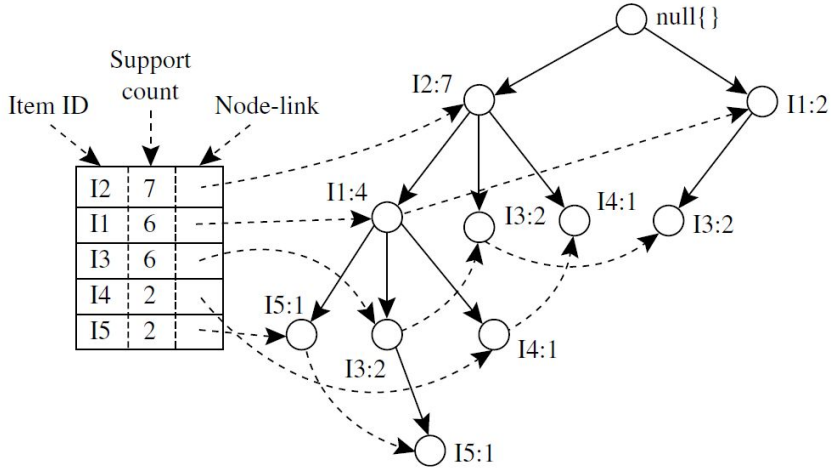
Item	*Conditional Pattern Base*	*Conditional FP-tree*	*Frequent Patterns Generated*
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

How is Conditional FP tree for I3 calculated? In video says take common

Figure 6.7, 6.8 in Jiawei Han, Micheline Kamber and Jian Pe, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2012.

Transactional Data for an *AllElectronics* Branch

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



FP-tree:

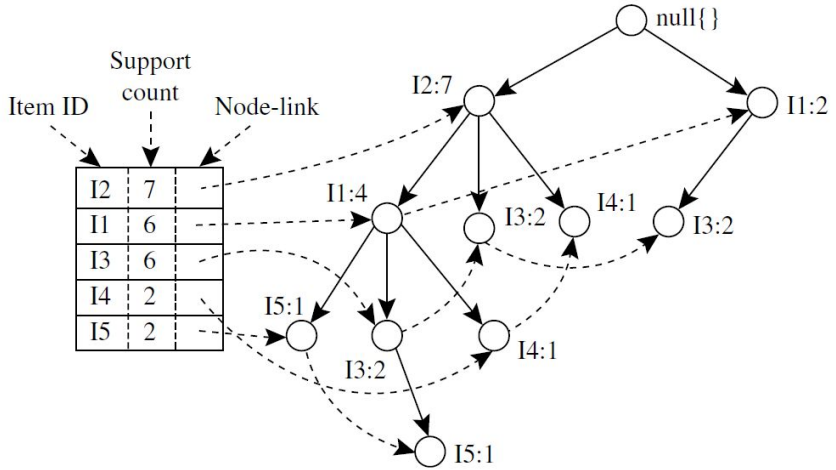
- A trie (a prefix tree) data structure.
- A compressed representation of a conditional DB.
- The path from the root to a leaf represents a repeated sub-transaction (frequent pattern) in the database.
- The path from the root to an internal node represent either a frequent pattern or a prefix.
- Each node is associated with a count representing the number of transactions in the original database that contain its path.
- The prefixes are sorted in the order from the most frequent to the least frequent to maximize the advantages of prefix-based compression.

FP-Growth pseudo code

Algorithm *FP-growth*(FP-Tree of frequent items: $\mathcal{FPT}$, Minimum Support: *minsup*, Current Suffix: P)
begin
  if $\mathcal{FPT}$ is a single path
    then determine all combinations C of nodes on the path, and report $C \cup P$ as frequent;
  else (Case when $\mathcal{FPT}$ is not a single path)
  for each item i in $\mathcal{FPT}$ **do begin**
    report itemset $P_i = \{i\} \cup P$ as frequent;
    Use pointers to extract conditional prefix paths from $\mathcal{FPT}$ containing item i;
    Readjust counts of prefix paths and remove i;
    Remove infrequent items from prefix paths and reconstruct conditional FP-Tree $\mathcal{FPT}_i$;
    if $(\mathcal{FPT}_i \neq \phi)$ **then** *FP-growth*$(\mathcal{FPT}_i, minsup, P_i)$;
  end
end

Transactional Data for an *AllElectronics* Branch

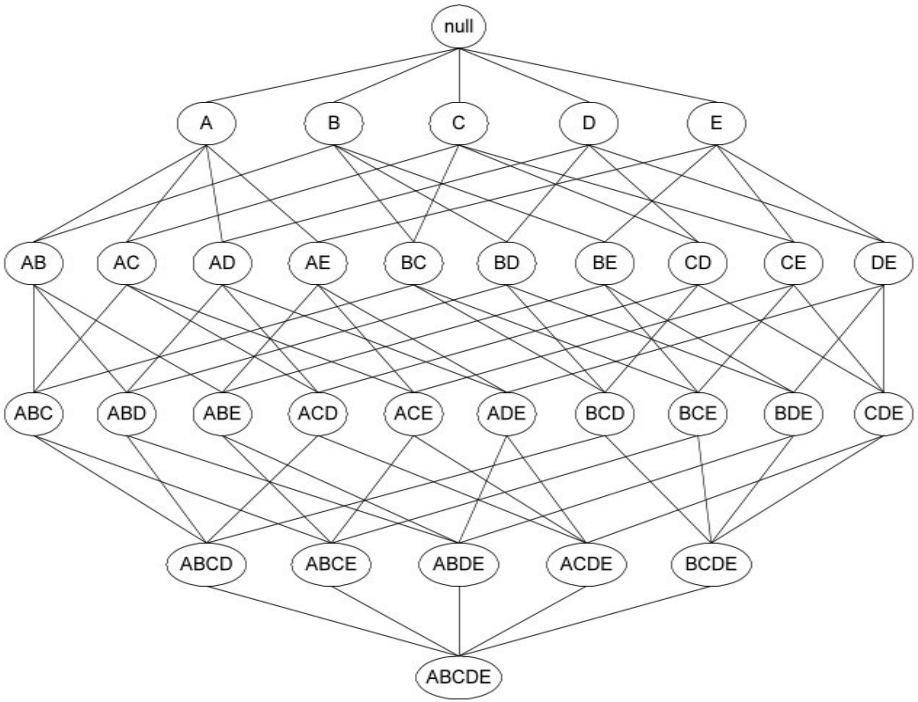
TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



FP-Growth:

- FP-Growth is a recursive Algorithm.
- FP-Growth find all frequent itemsets ending with a particular suffix by splitting the problem into smaller sub-problems.
- Suppose we are interested in finding all frequent itemsets ending in ‘I3’.
  - First make sure that I3 is itself frequent.
  - Solve the sub-problems of finding frequent itemsets ending with I1 I3, I2 I3, I4 I3, I5 I3.
  - In turn, each of these sub-problems is recursively decomposed into smaller subproblems, until they all stop recursing.
  - Merge the solutions of these sub-problems.

Does FP-Growth traverse the lattice ? How ?



Why does FP-Growth outperform Apriori ?

FP-Growth counts the database only once
Apriori creates more candidates to be tested than FP-Growth
Apriori creates candidates, and candidate generation can be expensive
Space complexity of FP-Growth is smaller, because no candidate generation/storage is needed
FP-Growth compresses the database, thus dealing with a smaller structure, on which we can mine the pattern directly

Evaluating Association Rules

- Support-confidence
- Lift
- Correlation analysis.
- IS Measure.
- etc

Table 5.9. Examples of objective measures for the itemset $\{A, B\}$.

Measure (Symbol)	Definition
Correlation (ϕ)	$\frac{Nf_{11}-f_{1+}f_{+1}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}$
Odds ratio (α)	$(f_{11}f_{00})/(f_{10}f_{01})$
Kappa (κ)	$\frac{Nf_{11}+Nf_{00}-f_{1+}f_{+1}-f_{0+}f_{+0}}{N^2-f_{1+}f_{+1}-f_{0+}f_{+0}}$
Interest (I)	$(Nf_{11})/(f_{1+}f_{+1})$
Cosine (IS)	$(f_{11})/(\sqrt{f_{1+}f_{+1}})$
Piatetsky-Shapiro (PS)	$\frac{f_{11}}{N} - \frac{f_{1+}f_{+1}}{N^2}$
Collective strength (S)	$\frac{f_{11}+f_{00}}{f_{1+}f_{+1}+f_{0+}f_{+0}} \times \frac{N-f_{1+}f_{+1}-f_{0+}f_{+0}}{N-f_{11}-f_{00}}$
Jaccard (ζ)	$f_{11}/(f_{1+}+f_{+1}-f_{11})$
All-confidence (h)	$\min\left[\frac{f_{11}}{f_{1+}}, \frac{f_{11}}{f_{+1}}\right]$

Michael Steinbach, Pang-Ning Tan, and Vipin Kumar, Introduction to Data Mining, Pearson 2005.
https://www-users.cs.umn.edu/~kumar001/dmbook/ch5_association_analysis.pdf

Limitation of Support-confidence

What is the limitation?

Consider this tea-coffee example, How strong is the rule:

{Tea} => {Coffee} ?

Conf({Tea} => {Coffee})= 150/200= 75% (high confidence)

But Sup(coffee)=80%

So actually drinking tea decreases the probability of drinking coffee !!!

How strong is the rule:

{Tea} => {Honey} ?

	B	¬B	Total
A	f11	f10	f1-
¬A	f01	f00	f0-
Total	f-1	f-0	

	Coffee	¬Coffee	Total
Tea	150	50	200
¬Tea	650	150	800
Total	800	200	

	Honey	¬Honey	Total
Tea	100	100	200
¬Tea	20	780	800
Total	120	880	

Lift Sup(A U B)/ (Sup(A) * Sup(B))

How strong is the rule:

{Tea} => {Coffee} ?

Lift({Tea} => {Coffee})=

0.15/(0.2*0.8) = 0.15/0.16 < 1

Tea and Coffee negatively correlated

How strong is the rule:

{Tea} => {Honey} ?

	B	¬B	Total
A	f11	f10	f1-
¬A	f01	f00	f0-
Total	f-1	f-0	

	Coffee	¬Coffee	Total
Tea	150	50	200
¬Tea	650	150	800
Total	800	200	

	Honey	¬Honey	Total
Tea	100	100	200
¬Tea	20	780	800
Total	120	880	

Limitation of Lift

Using Lift, How strong is the rule:

{p} => {q} ?

Lift({p} => {q})= 0.88/(0.93*0.93) = 1.02

How strong is the rule:

{r} => {s} ?

Lift({r} => {s})= 0.02/(0.07*0.07) = 4.08

But p,q appear together 88% of the time, while r,s appear together only 2% of the time.

Confidence is a better indicator than Lift in this case.

	p	¬p	Total
q	880	50	930
¬q	50	20	70
Total	930	70	

	r	¬r	Total
s	20	50	70
¬s	50	880	930
Total	70	930	

Correlation Analysis

- $\Phi = \frac{(f11 * f00 - f10 * f01)}{\sqrt{f\_1 * f\_0 * f1\_ * f0\_}}$
- -1 perfect negative correlation,
  1 perfect positive correlation,
  0 statistically independent.

- How strong is the rule:

  {p} => {q} ?

= (0.88*0.02) –(0.05*0.05)
/(0.93*0.93*0.07*0.07) = 0.232

	p	¬p	Total
q	880	50	930
¬q	50	20	70
Total	930	70	

	r	¬r	Total
s	20	50	70
¬s	50	880	930
Total	70	930	

How to choose a measure ?

- There are many more measures beyond what we have studied.
- These measures are not consistent. Applying two measures to the same set of rules, yield different sorting of rules from most interesting to least interesting.
- A good choice must be based on a clear understanding of the measure and its properties (inversion invariance, null addition invariance, etc).
- All these measures all called objective measures. They are good for automatic filtering of rules.
- One might additionally need to assess the rules using interactive visualizations, subjective measures based on domain experience, and template based search (i.e., report only the rules that can increase the sales of bio products).

Multi-level Association Rule Mining

Table 7.1 Task-Relevant Data, D

TID	Items Purchased
T100	Apple 17″ MacBook Pro Notebook, HP Photosmart Pro b9180
T200	Microsoft Office Professional 2010, Microsoft Wireless Optical Mouse 5000
T300	Logitech VX Nano Cordless Laser Mouse, Fellowes GEL Wrist Rest
T400	Dell Studio XPS 16 Notebook, Canon PowerShot SD1400
T500	Lenovo ThinkPad X200 Tablet PC, Symantec Norton Antivirus 2010
…	…

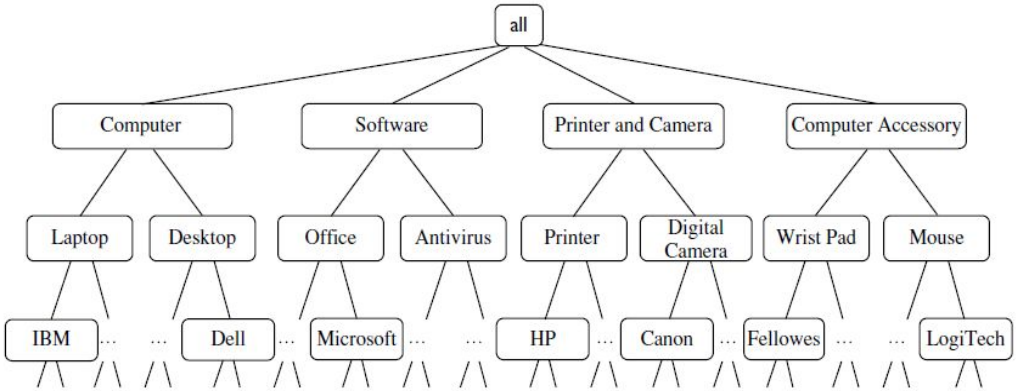


Figure 7.2 Concept hierarchy for *AllElectronics* computer items.

Jiawei Han, Micheline Kamber and Jian Pe, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2012.

Concept Hierarchies

- A concept hierarchy defines a sequence of mappings from a set of low-level concepts to a higher-level, more general concept set.
- Data can be generalized by replacing low-level concepts within the data by their corresponding higher-level concepts, or *ancestors*, from a concept hierarchy.
- level 0 at the root node is the most general abstraction level). Level k at the leaves is the most specific abstraction level.
- Concept hierarchies for numeric attributes can be generated using discretization techniques (data preprocessing).
- Concept hierarchies may be specified by users familiar with the data such as store managers in the case of the example.

Jiawei Han, Micheline Kamber and Jian Pe, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2012.

Multi-level Association Rule Mining

- Rules that are generated at multiple abstraction levels.
- In general, a top-down strategy is employed, where counts are accumulated for the calculation of frequent itemsets at each concept level, starting at concept level 1 and working downward in the hierarchy toward the more specific concept levels, until no more frequent itemsets can be found.
- For each level, any algorithm for discovering frequent itemsets may be used, such as Apriori or its variations.

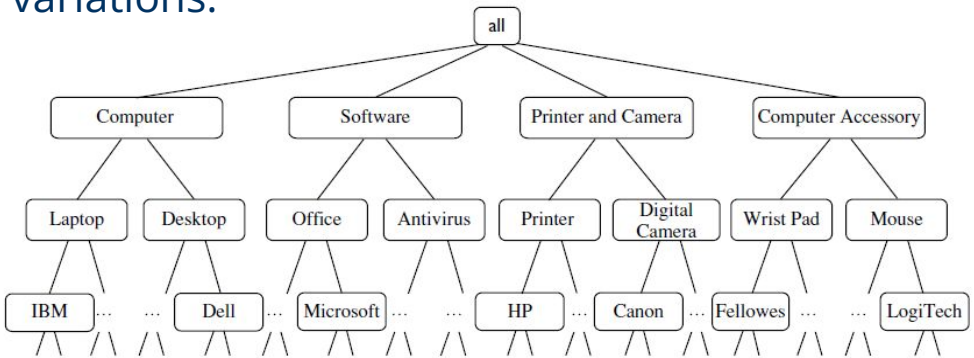


Figure 7.2 Concept hierarchy for *AllElectronics* computer items.

Jiawei Han, Micheline Kamber and Jian Pe, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2012.

Multi-level Association Rule Mining

- How to set the support threshold ?
- Using same support for all levels
  - + Simple for users as it requires a single parameter (minsup).
  - + Apriori (or alternative) can do pruning using the parent-child relationship. If a parent is not frequent, … ?
  - If minsup is too high, only higher levels of abstraction can generate rules.
  - If minsup is too low, redundant and uninteresting rules are generated.

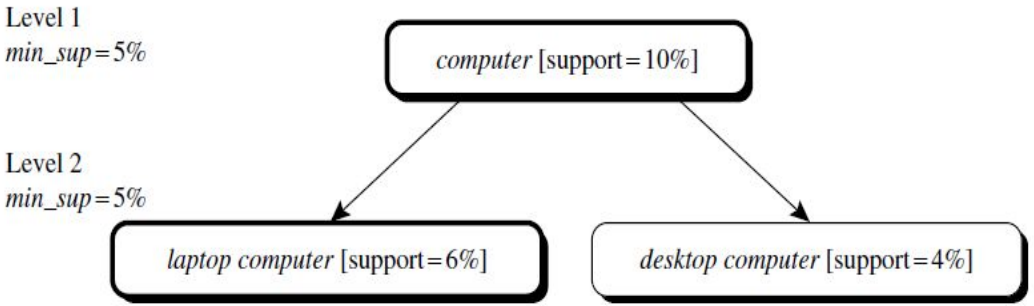


Figure 7.3 Multilevel mining with uniform support.

Jiawei Han, Micheline Kamber and Jian Pe, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2012.

Multi-level Association Rule Mining

- How to set the support threshold ?
  - Using reduced minsup at lower levels
  - Using group-based support
    - Domain experts know the frequently sold items and the rare ones.
    - Set higher minsup for the frequently sold items and lower minsup for the rarely sold ones.
    - For example: Cameras > 1000$ are rarely sold. Interesting rules that involve this item can be lost if a minsup that considers the average is used.
    - But Apriori accepts a single minsup parameter ???

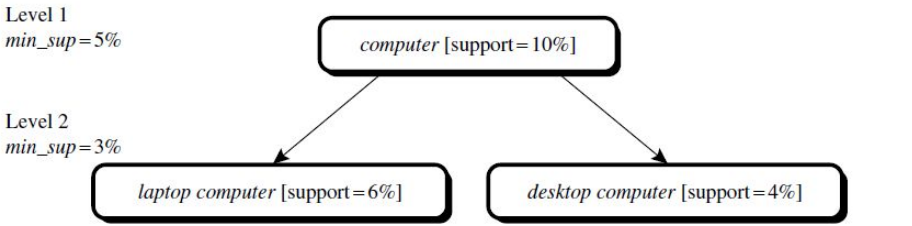


Figure 7.4 Multilevel mining with reduced support.

Multi-Dimenstional Association Rule Mining

- So far we have studied rules in the form:
    - buys(X, Camera) => buys(X, Printer).
- These rules are mined over the dimension “buys”.
- What about:
    - age(X, “20-29”) and occupation(X, “student”) => buys(X, “laptop”).
- Basically the mining is done in the same way, except that the counting is changed from counting occurrences of items into counting predicate fulfillment.
- This is typical for mining association rules from transactional databases.

Frequent Pattern Mining Applications

- Market basket analysis.
- Noise filtering (data cleaning):
  - Frequent itemsets are less probable to be noise.
- Data reduction (compression):
  - Very frequent itemsets are more probable to be non-interesting (e.g., stop words), and can be removed.
- Recommender systems:
  - Based on doing X, the system recommends doing Y.
- Cluster discovery:
  - Co-authors in DBLP.

Credits and Readings

These slides, except when explicitly stated, use material from:

- Charu C.Aggarwal. Data Mining The Textbook, Springer.
- Jiawei Han, Micheline Kamber and Jian Pe, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2012.
- Michael Steinbach, Pang-Ning Tan, and Vipin Kumar, Introduction to Data Mining, Pearson 2005.

Classification - Part 2

Invited Talk - December 6th 2023

Bayesian Classification

- A statistical classifier to predicts class membership probabilities
- Based on Bayes' Theorem.

Bayes' Theorem: Basics

Bayes' Theorem: $$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

- Let **X** be a data sample ("*evidence*"): class label is unknown
- Let H be a *hypothesis* that X belongs to class C_i
  - e.g., X will buy computer
- Classification is to determine P(H|**X**), (i.e., *posterior probability):* the probability that the hypothesis holds given the observed data sample **X**
  - e.g., X will buy computer, knowing that the age of X is 35 and the income is 40k, ...
- P(H) (*prior probability*): the initial probability
  - E.g., **X** will buy computer, regardless of age, income, ...
- P(**X**): probability that sample data is observed
- P(**X**|H) (likelihood): the probability of observing the sample **X**, given that the hypothesis holds
  - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Classification Is to Derive the Maximum Posteriori

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = (x_1, x_2, …, x_n)
- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

  needs to be maximized

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times ... \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ $g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

and $P(x_k|C_i)$ is $P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$

Naïve Bayes Classifier: Training Dataset

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

Data to be classified:
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Classifier: An Example

age	income	student	credit_rating	com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643

P(buys_computer = "no") = 5/14= 0.357

Compute $P(X|C_i)$ for each class

P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222

P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6

P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444

P(income = "medium" | buys_computer = "no") = 2/5 = 0.4

P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667

P(student = "yes" | buys_computer = "no") = 1/5 = 0.2

P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667

P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

X = (age <= 30 , income = medium, student = yes, credit_rating = fair)

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028

P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be **non-zero**. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian estimator)
  - *Adding 1 to each case*
    - Prob(income = low) = 1/1003
    - Prob(income = medium) = 991/1003
    - Prob(income = high) = 11/1003
  - The “corrected” prob. estimates are close to their “uncorrected” counterparts

Naïve Bayes Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
      - Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier
- How to deal with these dependencies? Bayesian Belief Networks

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier accuracy:
  - Holdout method
  - Cross-validation
  - Bootstrap

FN - Actually yes predicted no
FP - Actually no predicted yes
False Negative (FN): The number of instances incorrectly predicted as negative (but are actually positive)

Classifier Evaluation Metrics: Confusion Matrix

A confusion matrix is a table used in machine learning and statistics to evaluate the performance of a classification algorithm.

Confusion Matrix:

Predicted

Actual class\Predicted class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

Positive, Negative - predicted

TP here stands for True Positive predictions, for a binary classification problem like classifying the fraudulent transactions as 1, TP will give the count of the number of 1 s that were correctly classified as 1, i.e., number of fraudulent transactions that were classified as fraudulent. TN stands for true negative predictions, i.e., number of 0 s, non-fraudulent transactions, classified as 0. FP (False Positive) is the count of number of non-fraudulent transactions that were classified as fraudulent and FN (False Negative) is the count of number of fraudulent transactions that were classified as non-fraudulent.

Sensitivity, also known as True Positive Rate, Recall, or Sensitivity, is a performance metric used in binary and multiclass classification models. It measures the ability of a model to correctly identify positive instances out of the total actual positive instances in a dataset.
Specificity is a performance metric used in binary and multiclass classification models. It measures the ability of a model to correctly identify negative instances out of the total actual negative instances in a dataset. Specificity is also referred to as True Negative Rate

Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified
  - **Accuracy = (TP + TN)/All**
- **Error rate:** *1 – accuracy*, or
  - **Error rate = (FP + FN)/All**

Class Imbalance Problem:

- One class may be *rare*, e.g. fraud, or HIV-positive
- Significant *majority of the negative class* and minority of the positive class
- **Sensitivity**: True Positive recognition rate
  - **Sensitivity = TP/P** Sensitivity = TP/(TP+FN)
- **Specificity**: True Negative recognition rate
  - **Specificity = TN/N**

Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

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

- **Recall:** completeness – what % of positive tuples did the classifier label as positive. Perfect score is 1.0

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

- ***F* measure (*F_1* or *F*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- ***F_β*:** weighted measure of precision and recall. Assigns ß times as much weight to recall as to precision

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

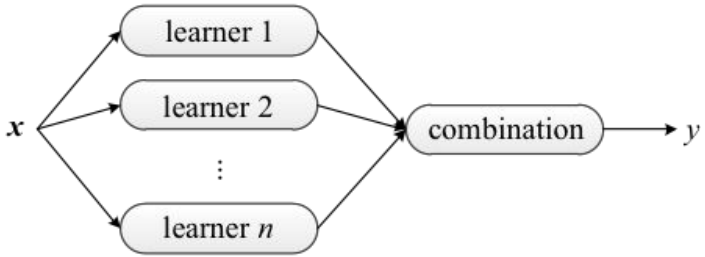
Classifier Evaluation Metrics: Example

Actual Class\ Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

Precision = 90/230 = 39.13% *Recall* = 90/300 = 30.00%

Ensemble Methods: Increasing the Accuracy

combining multiple models



- An ensemble for classification is a composite model, made up of a combination of classifiers.
- The individual classifiers vote, and a class label prediction is returned by the ensemble based on the collection of votes.
- Ensembles tend to be more accurate than their component classifiers.
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Why it work ?

Assume 3 independent classifiers, each of which can classify a sample x correctly with 70% accuracy. The probability that their ensemble classifier correctly classify x is:

$$0.7^3 + {}^3C_2 \text{ X } 0.7^2 \text{ X } 0.3 = 0.78$$

What if we use 5 independent classifiers ?

Bagging

Also bootstrap aggregation
one of the technique in bagging is Random Forest

Algorithm: Bagging. The bagging algorithm—create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction.

Input:

- D, a set of d training tuples;
- k, the number of models in the ensemble;
- a classification learning scheme (decision tree algorithm, naïve Bayesian, etc.).

In sampling with replacement, each item selected from the population is returned to the population before the next selection.

Output: The ensemble—a composite model, $M*$.

Method:

(1) **for** $i = 1$ to k **do** // create k models:
(2) create bootstrap sample, D_i, by sampling D with replacement;
(3) use D_i and the learning scheme to derive a model, M_i;
(4) **endfor**

Lets suppose, for a given problem statement, we have D datasets
We will create multiple models
For each and every model, we provide sample of dataset D'
For another model, dataset is resampled and provided to second model
this is called as row sampling with replacement
so, each model will get trained on different set of data
Once model is prepared, we need to predict some value on given test data
Test data is sent across all models and each of them will make its own prediction

To use the ensemble to classify a tuple, X:

let each of the k models classify X and return the majority vote;

Correct prediction is considered to be one with majority vote

Boosting

- **Weights** are assigned to each training tuple.
- Training sets are sampled with replacement, according to weights.
- A series of k classifiers is iteratively learned.
- After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to **pay more attention to the training tuples that were misclassified by M_i**
- The final **M* combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy.
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost (Freund and Schapire, 1997)

Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)

Initially, all the weights of tuples are set the same (1/d)

Generate k classifiers in k rounds. At round i,

- Tuples from D are sampled (with replacement) to form a training set D_i of the same size
- Each tuple's chance of being selected is based on its weight
- A classification model M_i is derived from D_i
- Its error rate is calculated using D as a test set
- If a tuple is misclassified, its weight is increased, o.w. it is decreased

Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$. Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

The weight of classifier M_i's vote is

$$\log \frac{1 - error(M_i)}{error(M_i)}$$

Random Forest (Breiman 2001)

- Bagging is not well combined with decision trees, because the ID3 Algorithm will generate highly similar/correlated trees
- Randomness is added to the tree induction Algorithm, as follows:
  - Before each split, a random selection of L attributes out of the available D attributes is made.
  - The split attribute is only selected among these L attributes.
- When L is much smaller than D, the trees in the forest are highly independent, so the ensemble functions well. But a single classifier will have a poor accuracy.
- What happens when L is close to D ?
- What if D is small ?

Credits and Readings

- These slides use material from:
  - Han, J., Kamber, M. and Pei, J. Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, Burlington.

problem with decision tree
if data is changed slightly, it will create different decision tree
i.e. our data is highly sensitive to training data

In random forest, we combine multiple decision trees which is less sensitive to data
why forest -> because we use multiple trees
n number of datasets are created to created n models,
each dataset contains same number of rows by performing random sampling with replacement

only subset of features are used to train each decision tree
for new input provided, we pass this dataset to each trees and note down prediction
and the result with be determined by maximum voting

why random? used two random processes - bootstrapping and random feature selection
bootstrapping makes sure that we are not using same data each time
feature selection - subset of feature is selected each time
if same feature is selected, each time they may end up having same decision node and that will behave very similarity which increases variance
how many features to select - researchers have found that number of features equal to log or square root of feature works well
how to use this for regression? while combining prediction, take average