INFS4203 2012 Theory

**Data mining** is the extraction of interesting (i.e. non-trivial, previously unknown, and potentially useful) patterns or knowledge hidden in large amounts of data. It is needed as the process needs to be automated due to the enormous size of today’s datasets.

# Mining Association Rules

**Association rule mining** involves discovering interesting relations between objects in large databases.

One example of this is market basket analysis - analysing customer buying habits by finding associations between the different items that are bought. That is, given a set of transactions, find rules that will predict the occurrence of an item based on the occurrence of other items.

An **item** is an item in the “basket”.

An **itemset** is a set of items (e.g. milk, bread, cereal).

A **k-itemset** is an itemset with k items (the above set is a 3-itemset).

A **transaction** is the items purchased in a basket.

A **transaction dataset** is a set of transactions.

An **association rule** is an implication of the form X → Y where:

* X, Y ⊂ I,
* I is the set of all items,
* X ⋂ Y = ∅

**Support** is the percentage of transactions that contain both X and Y. Can be absolute (the frequency or count) or relative (probability).

**Confidence** is the percentage of transactions that contain Y *given they contain* X. It measures how often items in Y appear in transactions that contain X.

**Confidence** (X → Y) = P (Y | X) = P(X ⋃ Y) / P(X)

Interesting associations are those with a support and confidence greater than some threshold. Low support and high confidence means that very few transactions contain X and Y, and X mostly only occurs with Y. High support and low confidence means that X and Y occur in a transaction often, but X also occurs without Y very frequently.

The goal of association rule mining is to find results that have support and confidence above some given thresholds.

**Brute force approach** simply lists every association rule, computes the support and confidence for each, and then prune any that fail the minimum support or confidence. Obviously computationally prohibitive.

## Apriori Algorithm

The **Apriori principle** states that if an itemset is frequent, then all of its subsets must also be frequent. So, find all itemsets with minimum support first (**frequent itemsets**), then check their confidence.

Thus, the **Apriori algorithm** is as follows:

* Initialise the candidate itemsets with single items in the database (k = 1)
* Scan the database and determine support for each
* Generate any new itemsets if the minimum support is achieved
* Repeat process, stop when there are no new itemsets generated

To make a new itemset {A, B, C}; {A, B}, {A, C}, and {B, C} are all required to meet the minimum support.

The Apriori algorithm has major computational challenges:

* Multiple scans of transactional database
* Huge number of candidates
* Tedious workload of support counting for candidates

Computational complexity factors for Apriori algorithm:

* Choice of minimum support threshold
  + Lower support threshold results in more frequent itemsets
* Dimensionality (number of items) in the data set
  + If number of frequent items increase, both computation and I/O costs may also increase
* Size of database
  + Since Apriori makes multiple passes, run time of algorithm may increase the number of transactions
* Average transaction width
  + This may increase maximum length of frequent itemsets

Candidate generation is the inherent cost of Apriori algorithms, no matter what implementation technique is applied. e.g. if there are 100 items, the Apriori algorithm may need to generate a very large number of itemsets.

## FP Growth Algorithm

Do not study!

# Classification Algorithms

Given a collection of records (training set), find a model for a class attribute as a function of the values of other attributes. The goal is to accurately assign previously unseen records to a class.

Two types: lazy and eager. **Eager** learners generate the model explicitly (decision tree), **lazy** do not (kNN).

## Performance Evaluation

**Confusion matrix:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Prediction** |  |
|  |  | **Positive** | **Negative** |
| **Actual** | **Positive** | A | B |
|  | **Negative** | C | D |

A: True positive

B: False negative

C: False positive

D: True Negative

**Accuracy** = (TP + TN) / (TP + TN + FP + FN)

**Precision** (“exactness”) = TP / (TP + FP)

**Recall** (“completeness”) = TP / (TP + FN)

To evaluate models, can use holdout or cross validation.

**Holdout** randomly takes some percentage of the examples as training and another as testing. This is then repeated multiple times. It is used for data sets with a large number of tuples.

**Cross validation** randomly partitions the data into k mutually exclusive subsets, each approximately equal size. At an iteration, one partition is used as the test set and the others as the training set. Used for moderately sized datasets.

## Nearest Neighbour Classifiers

“If it walks like a duck, quacks like a duck, then it’s *probably* a duck.”

Requires three things:

1. The set of stored records
2. A distance metric to compute distance between records
3. The value of k, the number of nearest neighbours to retrieve

To classify an unknown record:

1. Compute distance to other training records (using the given distance metric)
2. Identify k nearest neighbours
3. Use class labels of nearest neighbours to determine the class label of unknown record (e.g. by taking majority vote or by weighing the vote according to distance, i.e. w = 1/d^2)

If k is *too small*, it is sensitive to noise. If it is *too large*, it could include points from other classes.

**K-nearest neighbours** of a record *x* are data points that have the k smallest distance to x.

kNN is robust to noisy data as it can average the k-nearest neighbours. It does, however, suffer from the “curse of dimensionality”. This is where the distance between neighbours could be dominated by irrelevant attributes. It can be overcome by eliminating the least relevant attributes.

## Bayes Classifier

The **Bayes classifier** is a probabilistic model for the classification problem.

Bayes’ Theorem allows conditional probabilities to be flipped, i.e. to find P(spam | keywords) we can use P(keywords | spam).

If we assume independence, P(A1, A2, …, An) | C) = P(A1 | C) P(A2 | C) … P(An | C)

Thus, given a bunch of attributes, calculate P(A | C) for each one. Do this for all possible classes. Choose the class that maximises P(A1, A2, …, An) | C) P(C)

Naive Bayesian prediction requires each conditional probability be non-zero, otherwise the predicted probability is zero. To correct this we could add one to each case (**Laplacian correction**).

Advantages of this classifier are that it is easy to implement and obtains good results in most cases. Some disadvantages are that the independence assumption does not always hold - in practical situations, dependence exists among variables.

## Decision Tree To generate tree,split the records based on an **attribute test** that optimises certain criterion. Problems are how to split the records and how to decide when to stop splitting.

Splitting attributes depends on the type (nominal, ordinal, continuous) and can be split multiple ways (2-way, or multi-way).

Advantages of decision trees is that they are inexpensive to construct, fast, easy to interpret for small sizes, and the accuracy is comparable to other classification techniques.

### Impurity Measures

Can use **GINI index** to decide which attribute gives the best split.

**GINI(t)** = 1 - sum(P(j | t)^2 for j=1 to number of classes at a node t)

e.g. = 1 - (1/4)^2 - (3/4)^2

Then, can determine how good a split is by weighting each of the partitions in a split:

**GINI(split)** = sum(ni/n GINI(i)) where ni is the number of records at child i and n is the number of records at parent node p

Can also use classification error at a node:

**Error(t)** = 1 - max P(i|t)

e.g. = 1 - max(1/4, 3/4) = 1 - 3/4 = 1/4

GINI index gives a smoother curve than classification error (which is linear).

### Stop Expanding

Stop expanding when all records belong to same class, or all records have similar attribute values. Sometimes need early termination to avoid overfitting. Thus could stop when the number of instances at a node are less than some value, or if expanding the current node does not improve the impurity measures.

**Overfitting** is where the model starts to reflect noise and outliers.

# Clustering

**Clustering** simply clusters objects that are similar together. It is different to classification as it is unsupervised and does not need to know the actual classifications.

A good clustering has high intra-class similarity and low inter-class similarity.

Two types: partitional and hierarchical.

**Partitional clustering** is a division of data objects into *non-overlapping* subsets such that each data object is in exactly one subset.

**Hierarchical clustering** is a set of nested clusters organised as a hierarchical tree.

## K-means

Each cluster associated with a centroid, each point assigned to nearest centroid. Number of clusters (k) must be specified.

Algorithm: select k points as the initial centroids. Form k clusters by assigning points to nearest cluster, then recompute centroid. Repeat until centroids do not change.

Stopping condition is often “until relatively few points change clusters” rather than until no centroids change.

K-means converges pretty fast (usually in the first few iterations)< but this does not mean the clusters are optimal.

**Sum of Squared Error (SSE)** can be used to evaluate K-means clusters.

**SSE** = sum of SSE for each cluster, ie. sum of the (distance for each point to the centroid squared)

Choice in initial picks is important. To reduce error some post processing is often done: small clusters (outliers) are eliminated, loose clusters (high SSE) are split, and close cluster (low SSE) are merged.

K-means has problems when clusters are of different sizes, densities, or non-spherical shapes, and with outliers.

## Density-Based Clustering

**Eps** is the maximum radius of the neighbourhood, a neighbourhood with a radius of eps is defined as the eps-neighbourhood.

**MinPts** is the minimum number of points in an eps-neighbourhood of a point

**Neps(q)** is the set of points in the neighbourhood of point q

A point p is **directly density-reachable** from point q if it belongs to Neps(q) and q is a core point (Neps(q) >= MinPts).

A point p is **density-reachable** from a point q if there is a chain of points such that each one is directly density-reachable from the last.

A point p is **density-connected** to a point q if there is a point o such that both p and q are **density-reachable** from o.

**DBSCAN**: select a point p, if Neps(p) < MinPts, mark as noise. Otherwise, retrieve all points density-reachable from p and if p is a core point, a cluster if formed. If p is a border point, go to the next point.

## Hierarchical Clustering

Two types: agglomerative and divisive.

**Agglomerative clustering** starts with each point as an individual cluster and each step merges the closest pair of clusters until only one cluster is left.

**Divisive clustering** starts with one all-inclusive cluster and at each step splits a cluster until each cluster contains a point (or there are k clusters)>

These algorithms use a similarity of distance matrix, merging or splitting only one cluster at a time.

Different distance approaches: min (single linkage), max (complete linkage), group average, and distance between centroids.

Min is sensitive to noise and has the chaining phenomenon where clusters may be forced together due to a single element being close, even though many of the elements in each cluster are far apart.

Max tends to break large clusters.

Group average is a compromise between single and complete linkage.

# Distances

Many types of distance metrics: Manhattan (L1 norm), Euclidean (L2 norm), max of any dimension (Lmax norm). Also Jaccard (used for binary), cosine (difference in angles), and edit (number of operations needed to change one string into another)

**Jaccard** = number of terms in intersection of two sets / number of terms in their union

**Cosine** = dot product of sets / magnitude of each set multiplied together

# Outlier Detection

Useful for credit card fraud detection, fault detection, etc.

General assumption is that there are considerably more “normal” observations than “abnormal” observations.

A profile is built based on “normal” behaviour

Few types: statistical, proximity, cluster

## Statistical

Statistical assumes a parametric model and the test depends on data distribution, mean, variance, and number of expected outliers.

Limitations: data distribution may not be known, most tests are for single attributes

## Proximity

Distance and density based schemes.

For distance, an object is an outlier if its neighbourhood does not have enough other points.

For density, an object is an outlier if its density is relatively much lower than that of its neighbours.

The effectiveness of proximity-based methods rely highly on the proximity measure. In some cases, these measures cannot be easily obtained. FInding groups of outliers which stay close to each other is also difficult.

## Cluster

Outliers belong to small or sparse clusters, or not to any cluster

Expensive: need to find clusters first

# Text Mining

**Text mining** refers to data mining using text documents as data. It uses information retrieval methods to pre-process text documents.

Text mining has unstructured data, so the obvious idea would be to convert this into structured data so it can be mined.

A document can be represented by a number of methods.

In a **vector space model (VSM)** each word/term is a dimension.

Each document is treated as a “bag” of words or terms. Given a collection of documents, the vocabulary is the set of distinctive words/terms in the collection. Finally, a weight is associated with each term of a document.

Unfortunately, there are many words in the English language. This can partly be solved by removing any “stop” words, as well as “stemming”.

Another problem is that the importance of a term has to be determined.

For this, **TF-IDF (term frequency-inverse document frequency)** scheme can be used. This increases the importance of words that occur rarely in the collection.

**TF** = number of times word appears in document

**IDF** = log(total documents/document frequency)

**TF-IDF** = TF x IDF

So steps are:

1. Remove stop words
2. Stemming
3. Construct a VSM
4. Compute the weight of each word
5. Normalise the VSM

Similarity can then be found by using a distance measure, e.g. cosine similarity.

Issues: how to define stop words (“A’ in “Vitamin A”), how to perform stemming (“booking” and “book”), and spelling errors.

# Web Mining

Text mining not enough. Web pages have intra-page structure, inter-page structures, usage data, profiles, registration information, cookies, etc.

## Content Mining

**Content mining** extends the functionality of basic search engines.

**Web crawlers** are a web robot (spider) that traverses the hypertext structure in Web pages (that is, follows the links) and collects information from the visited pages. They use this information to construct indices for search engines.

Three types: traditional (visits everything), incremental (selective search), focused (only visits pages related to a particular subject).

**Focused crawlers** only visit those links from a page that are determined to be relevant to the originating page - classifier is static after learning phase.

Web access or contents can be tuned to fit the preferences of each user.

**Content-based filtering** retrieves pages based on similarity matching between the pages and the user’s profile.

**Collaborative filtering** identifies preferences based on ratings of similar users (e.g. which pages did they visit). Challenges are scalability (many users and items), robustness (there will be noise), sparsity (user-item rating matrix is very sparse), and cold start (how to make recommendations to new users)

## Structure Mining

Mines the structure (links, graph) of the Web. One technique is PageRank by Google.

In PageRank, the importance of a page is calculated numerically from the number of pages that point to it (backlinks).

Weighting is used to provide even more importance to those backlinks that come from other important pages.

PageRank requires solving a huge linear programming problem

A **cyber community** on the Web is a group of web pages sharing a common interest. Pages in the same community should be similar in content and differ from other communities (similar to clustering).

**Co-citation** defines the similarity of a web page A and B by measuring the number of pages that *cite* both A and B.

**Bibliographic coupling** defines the similarity of A and B as the measure of pages *cited by* both A and B.

## Usage Mining

Base upon how the Web is used

Data can be from clickstreams, user sessions, or a server session.

To keep track of this data, use a log. But first, must cleanse and sessionize the data.

**Cleansing** involves replacing the source IP address with a unique but non-identifying ID. Do the same for URLs. Delete any error records and records containing non-page data.

**Sessionizing** involves dividing the web log up into sessions. Two methods: identify consecutive page references from a single source IP occurring within a predefined interval. Or, identify consecutive page references from a single IP where the interclick time is less than some threshold.

Issues: identification of exact user not possible, single session isn’t well defined, and exact sequence of pages references is unavailable due to caching of web pages. There is also privacy and legal issues.

## 

## 

## 3.X Similarity and Dissimilarity

### Proximity refers to a similarity or dissimilarity.

### Similarity

Similarity is the numerical measure of how alike two data objects are. It is higher when objects are more alike.

### Dissimilarity

Dissimilarity is the numerical measure of how different two data objects are. It is lower when objects are more alike. Minimum dissimilarity is 0.