CSCI 4144

Data Mining Tasks and Solution Techniques

Gao

Alex Safatli

Thursday, February 7, 2013

Contents

Mining Association Rules

Concept Overview

Association rule mining is the finding of frequent togetherness patterns; finding interest association regularities between data items, attributes, causal structures (or sequences, etc. contained in a database. Each **association rule** is a piece of knowledge *called a model which describes a relationship between two particular item sets (or attribute sets, etc.)*.

Association rule query examples include:

1. What product items were often purchased together?
2. What are the key factors determining income ≥ $50,000 for the age group of 25 to 34?
3. What are the subsequent purchases after buying an LED TV?
4. What kinds of DNA are sensitive to a particular new drug?
5. What are the frequent click stream (session) patterns of an e-commerce website?

One type of togetherness involves **market basket analysis** which has rule:

$$X \Longrightarrow Y(sup, conf)$$

When a customer visits the superstore and purchases a set of items, we can investigate how this *togetherness* is significant. All of the patterns can be represented in a fashion as above with a confidence and support value. Questions could be asked such as "in this shopping basket, the shopper purchased a quart of orange juice, some bananas, dish detergent, window cleaner, and a sick-pack of soda – is soda typically purchased with bananas?"

The motivation of market basket analysis is that the generation of association patterns about business data objects produces intuitively obvious results that the business owner can use in order to take advantage of the knowledge within their industry to improve their business.

So, what market analysis accomplishes is that:

1. It finds *interesting relationships amount retail products*; analyzes customer buying patterns by finding associations among different items that customers place in their "baskets".
2. It finds *all possible combinations of potentially interesting product groupings* — one basket tells one about one trip shopping, but all purchases by all customers have much more information.
3. It leads way to *business improvements*: design promotions, arrange shelf or catalog items, develop cross-marketing strategies, CRM, etc.

Examples of customer transaction pattern analysis could include analyzes grocery sale transactions (see slide). The association rule is that when a customer buys orange juice, he/she will also buy soda: orange juice $\Longrightarrow$ soda $(40, 50)$ — this means there is 40% support (*how many of them have it*) transactions have both items and 50% confidence (using the subset of those that have orange juice) that transactions containing orange juice also contains soda.

Another example is a video store data analysis: {Coupon} $\Longrightarrow$ {New releases} (10%,52%). If a customer uses a coupon in a transaction, there is 52% confidence that he/she will rent a new release; this occurs in an overall 10% of all transactions.

Finally, a third example is the recommender system (as seen on `www.amazon.com`) — customers who bought some book x also bought book y and z.

Concept Overview continued on next page...

How does one define a piece (form) of association knowledge – provide an example? These are association rules: models, descriptions of relationships, togetherness between two item sets (*if... then...*). Example: if customers purchase orange juice, they also purchase soda (with given percentages).

What are the usefulness measures for each association rule – define each measure? Once you claim you have knowledge, it is a representation which covers a subset; will always need to quantify how useful this discovery is. There are two measures: **support** and **confidence**.

Support: probability that a transaction contains both X and Y. These counts can give you information about how strong a support the knowledge you found has — **support rate**. Confidence: conditional probability that when X, also Y. Measuring how accurate the rule would be if the implication holds — **confidence rate**.

What are the two major steps of mining association rules in the Apriori algorithm? (1) finding frequent itemsets and (2) generating association rules. Will be the focus of the next section.

What is Apriori knowledge, and why is it significant for the algorithm? You can imagine that the number of unique items for a typical retail store is very large. Apriori knowledge is very useful; if you look at association rule discovery solutions, all of the current solutions still need this knowledge.

Apriori Algorithm

There are two phases to the **apriori algorithm** for association rule mining, two major procedures:

1. *Find frequent itemsets.* Search all frequent itemsets from the database.
2. *Generate association rules.* Derive rules from each frequent itemset.

Before we go further, we need to define **transaction data concepts**. In a database, you will have a **set of items** (all unique items), and itemsets (**combinatorial sets of items**). Association rules are considered item relationship patterns *only if it passes the measures* of MINIMUM SUPPORT AND CONFIDENCE RATES.

If asked to find all rules $X \Longrightarrow Y$ with min support and min confidence, the rules found can be combined to generate other augmented rules. How do we discover association rules as such? This can be done with the **apriori algorithm**. Out of the two above procedures, the procedure that needs more effort is the first one — the second one is merely systematic, while the first one heavily depends on the search space.

Finding unique itemsets will involve:

- making sets out of every unique sets of items,
- going to 2-item itemsets,
- ...,
- going to n-item itemsets.

The number of itemsets you can generate can be predicted. If a database has n unique items, and k denotes the combination of k items ($k \leq n$), the possible combinations can be calculated using:

$$\sum_{k=1}^{N} C^K$$

Where $C^K = \frac{n!}{((n-k)! \times k!)}$. The number of itemsets in total can be calculated using:

$$I = 2^n - 1$$

Apriori Algorithm continued on next page...

Where I is the number of itemsets. For $n = 100$, this would be $2^{100} - 1$. Most of those combinations have nothing to do with what we are looking for, however (who will buy 100 items together, for instance?).

This leads us to the question: how many itemsets *actually* occurred in a database D – *ground truth*? You can then ask the question: *how many nodes need to be searched when supp = X%*? If an itemset is not frequent (e.g. does not exceed the threshold), all of its subsets are not frequent.

We can now formulate a heuristic. IF AN ITEMSET IS NOT FREQUENT, ALL OF ITS SUPERSETS ARE NOT FREQUENT, i.e. no need to search them (generate them). This can let one perform **search pruning** – a first step.

An **apriori property** is the **frequent itemset property**: any subset of a frequent itemset is frequent. Its contrapositive: if an itemset is not frequent, none of its supersets are frequent. Therefore, a subset is frequent *iff* its superset is frequent.

How do we make the first procedure computation more efficient? Do we need to search (or generate) the entire space (or generate all itemsets)? What itemsets need to be searched? The strategy is: *apply the heuristic knowledge* – apriori property to prune the search space.

A **candidate generation-and-test approach** involves the **apriori pruning principle**: *if there is any itemset which is infrequent, its superset* SHOULD NOT BE GENERATED FOR TESTING! The method is then to:

- generate length $k+1$ candidate itemsets from length k frequent itemsets only, and
- test the $k+1$ candidates against the DB;
- any subset of a frequent itemset must be frequent — if A,B,C is frequent, so are any of its subsets — **test the candidates** (**pruning**)

The efficiency of this approach has been extensively studied. See pseudo-code. `apriori_gen()` is the core method of the algorithm – comprised of JOIN operations.

What are the two key steps of finding frequent item sets in the Apriori algorithm? We are looking for togetherness patterns; so if we have a set of unique items for a database, we need to formulate combinations. We map these to the lattice of the space; very structured search space. We then prune them. This is known as *hypothesis testing*.

How is Apriori knowledge applied in the algorithm for finding frequent item sets? (Or at which steps of the algorithm is the knowledge applied, and how?) See above.

In the candidate generation procedure, how does one avoid generating redundant (k+1)-itemsets based on k-itemsets? Going layer-by-layer we find solutions, but with the knowledge measure, we know where the search space can be cut off. This is an extension of the second question. Just investigating a subspace of the full space.

How is Apriori knowledge applied again in the rule generation algorithm? This will be discussed in this class.

Given an input D *and the Apriori algorithm, how does one trace the algorithm for intermediate results and the final output*? See midterm, slides.

Mining Various Kinds of Rules

Recall there are two phases in association rule mining: the first involves finding all frequent itemsets, and the second involves GENERATING ALL STRONG RULES. To generate all rules from $\mathscr{L}$, one:

1. **Derives candidate rules** from each $k \geq 2$ itemset $\mathscr{L}_k$ in $\mathscr{L}$.
2. **Tests for strong rules by filtering** the rules with $conf < min\_conf$.

Generating or *deriving* candidate rules from $\mathscr{L}$ involves:

- For each $\mathscr{L}_k$, generate all non-empty subsets S of $\mathscr{L}_k$.
- For every non-empty subset s in S of itemset $\mathscr{L}_k$, generate $s \Longrightarrow (\mathscr{L}_k - s)$.

Realize that, for example, given the frequent itemset $\{A, B, C\}$, we CANNOT generate $\{A, B\} \Longrightarrow \{A, C\}$. It conflicts with the condition $X \Longrightarrow Y | X \cap Y = \emptyset$. Therefore, $s \Longrightarrow (\mathscr{L}_k - s)$ guarantees the intersection is the empty set.

Testing for strong rules involves calculating the **confidence measure** $conf(X \Longrightarrow Y) = P(Y|X) = \frac{supp(X,Y)}{supp(X)}$ where *no count is needed* — as long as those support reads are known (which should have been calculated before).

Is there any room here for improving efficiency? APRIORI KNOWLEDGE guarantees that we can establish a **pruning rule**: it improves the search for generating sub-itemsets. This pruning rule, or "efficiency trick rule" can be formalized as so: If s is a subset of $\mathscr{L}_k$, we know $sup(s) \geq sup(\mathscr{L}_k)$. This means we can have $conf(s \Longrightarrow (\mathscr{L}_k - s)) \geq conf(s' \Longrightarrow (\mathscr{L}_k - s'))$ where s' is a subset of s. The reverse is true, as well: if the LHS does not pass the *min_conf*, neither will the RHS. See slide 18 of *note4.4.pdf* for a diagram of this.

NOTE Given a relational database, finding all associations has a few issues: (1) how to convert relational data into items, and (2) how to convert quantitative attributes to categorical attributes for forming items. With transaction data, attributes can be considered as binary variables (bitmasks). In a relational database, an attribute can have a domain of binary or multiple/continued values — all domain values should be considered "items".

See slides on the general architecture for an Apriori program (slide 20-21 in *note4.4.pdf*). There will be a question on this on the final exam.

How does one mine multilevel association rules? Treat concept hierarchies as items themselves.

Note we can also apply constraints to association rule mining as well. This is known as **constraint-based association mining** — see slides (this was not covered explicitly in class).

Classification Data Mining

Problem Overview & General Issues

Seeing, overall, the demands or solutions given enablers to support decision making and customers. Finding regularities and patterns (knowledge) is, similarly, a goal of this manner of data mining.

Classification is a type of knowledge that can be mined in many ways; two of them include **decision tree induction** and **Bayesian classification**. Another involves **neutral networks**, but these are typically considered black boxes and do not give clear explanations for results. There is no one universal approach; classification is a very large domain of some applications regarding one concept.

Classification deals with a target concept, always; learn knowledge to understand how the values for this target concept (an *attribute* in the data set) can be labelled and answered. An example is *risk analysis modelling* (see slide). You also need **training data**: the dataset in which you keep in order to make decisions on later data.

Therefore you divide your data into two parts: **testing** and **training data**. This is also an element of *recursion* – very much depth-first in a way.

Given a training data set $D = \{t_1, t_2, ..., t_n\}$ which has n rows (records, tuples) and m columns (features, attributes), for a given target concept (selected feature) with classes $C = \{C_1, ..., C_p\}$, classification mining is the finding of a **mapping function** $f : D \Longrightarrow C$ — classification knowledge or model — where each t_i in D is assigned to a class of C. When a model is obtained, you can then use it as a **classifier**.

Problem Overview & General Issues continued on next page...

Classification mining is to discover the decision model f by inductive learning on the trianing data; it is a divide (partition D) and conquer (find C) based approach for binding a decision model f. It is a **heuristic search** and the model f is the classes' predictor and descriptor. **Prediction** is to assign an unseen object to a class of the target concept; applying the formed model f to predict a class for a new data case.

In the example of university professors, *tenured* is the target concept.

Classification application development is a three-phase process:

1. Training,
2. Testing, and
3. Prediction

The last two use the created model from the first step. *Note that classification is very much analogous to the **empirical cycle***: the process starts from an observation (preprocessing will occur, as well – creates prepared data) that is defined. An analysis then occurs regarding regularities (knowledge) — this is the *search problem*. Theory is the using of some selective language or representative scheme to describe the knowledge (for example, a decision tree is a tree). Prediction is, for all intents and purposes, the finding of knowledge for explanation from unseen objects.

NOTE All data mining processes can be linked to the empirical cycle.

How can classification methods be evaluated? Its predictive accuracy, speed/scalability, robustness, interpretability, and goodness of rules (see slide).

Decision Tree Induction

WHAT IS INDUCTION? It is a reasoning approach in that a concept can be learned or supported by gathering evidences from observations. However, it can not prove the concept. This indicates the process of reason from observation — to make a statement based on the observation of facts. It is not a sound logical reasoning but a *plausible reasoning technique*.

Decision Tree Induction is a supervised inductive learning process. Training data is partitioned based on a divide-and-conquer strategy. D is continually divided into subsets until each subset has only one label (all examples in the subset share a same class label). Each tuple of D is placed into a class group based on the leaf node or the region within which it falls.

Given a training set D with n tuples, a schema of D with m attributes, and classes C of a target concept, a **decision tree** is defined as a root node that has no incoming arcs and zero or more outgoing arcs, internal nodes of an attribute A_i, each arc labelled with an attribute value (predicate condition) of the parent node, and each leaf node is labelled with a class C_j.

Note that this is a classification problem; the knowledge is a set of rules that is dependent on the kind of approach — the representation method will therefore be different. It is a divide-and-conquer strategy, so every time a situation is found (a problem is mapped into a search problem), a decision has to be made.

Why is mining classification rules a supervised learning process? How about association rule mining? What is the difference between these two concepts: supervised and unsupervised? This is the idea of machine learning. Training, or a concept to learn, is present in supervised learning (it needs a very particular target concept and you have answers for it already).

So: (1) you need a target concept, and (2) you have present answers there and you need to find the rules.

Decision Tree Induction continued on next page...

Unsupervised learning is, for instance, association rule mining, but it could be treated as supervised as well. It just means there are no particular attributes or training data to use.

What are the major phases of conducting a classification data mining application? There are different phases; it does not matter what approach you use. To perform classification, you need to pre-process, train, test, and then predict.

Can you describe a mapping between a classification task and the empirical cycle? You have to be clear here; this is very important. No matter the knowledge discovery process, you need to use this model. See above.

What is the general strategy and approach of DT induction for classification data mining? Decision Tree Induction is a solution method, part of graph searching, where you are searching for the *best* tree. The strategy is **divide-and-conquer** — the ID3 algorithm is all about graph searching; it is a heuristic search (**depth-first search**, **greedy search**). How to divide is based on the heuristic and what the attribute value set is.

Note that region images can be used to explain the process of divide-and-conquer. See slides 8 to 9 of *note5.1.pdf*.

A general DT Induction Algorithm can be represented in this pseudo-code:

```
DTBuild(D) {

   Input:
      D // training data
   Output:
      T // decision tree

   T = empty;
   determine best splitting criterion; // choose an attribute (to split)
   T = create root node and label with splitting attribute;
   T = add arc to root node for each split predicate and label; // split

   for each arc do: // test (stop or continue)
      D = database created by applying splitting predicate to D;
      if stopping point reached for this path: // up to you to decide
         T' = create leaf node and label with appropriate class;
      else:
         T' = DTBuild(D); // recursively repeat
      T = add T' to arc; // to choose an attribute to split

}
```

Technique issues of this method include:

- choosing a **splitting attribute** (how to divide),
- **ordering** of splitting attributes,
- **splits**,
- **tree structure**,
- **stopping criteria**,
- **training data**, and
- **pruning**.

General issues of the method of DT Induction are:

1. how to **construct the tree** (greedy search),
2. how to **choose an attribute** to split (information gain), and
3. how to determine **when to stop** splitting (reach leaf node, majority votes).

DT algorithms include:

- **ID3** (Interactive Dichotomiser 3) — Quinlan, J.R. (1986),
- **C4.5 (C5.0)** — Quinlan, J.R. (1993),
- and others include CART, CHAID, Chi-squared, etc.

HOW DOES ONE SPECIFY TEST CONDITIONS? It depends on attribute types: nominal, ordinal, or continuous. It also depends on the number of ways to split: 2-way (binary) split or multi-way split.

Splitting based on nominal attributes, **multi-way split** uses as many partitions as distinct values and **binary split** divides values into two subsets — need to find the optimal partitioning.

HOW TO DETERMINE THE BEST SPLIT. Which attribute is the best classifier? This depends on:

- how well a given attribute split the records for classification, and
- we want to select the best one to split.

What is a good quantitative measure? **Information gain** (a probability property). ID3 uses information gain to select among the candidate attributes at each step while growing the tree. Constantly asking, "is information gained" when a certain path is taken.

How does one measure the information contained in an attribute for the classification of the target concept? In general, **information** is associated with a **question** and the **message** for answering the question. Information measure is to quantify the **outcome** of some **expectation** from the *message* for answering the *question*.

In information theory, information is measured in **bits**. One bit of information is enough to answer a yes or no question about which one has no idea, such as the flip of a fair coin. To derive the quantity of information (Q), we consider information to be defined in terms of the number of bits necessary to encode the factor by which the probability of the particular outcome has been increased.

$$Q(message) = P_2(outcome_{after}) - P_1(outcome_{before})$$

See slide for an example (a cow in a pasture problem). For classification, information measure is about **class purity**. If the answer gives a subset more closer to pure (single), more information is obtained. **Entropy** is the quantification of the impurity, or uncertainty, of an arbitrary set of data. For classification, entropy is used to estimate how close a data set is to a single class. **Information gain** measures the expected reduction in Entropy:

$$Gain(S, A) = Entropy(S) - \sum^{k} \frac{S_j}{S} Entropy(S_j)$$

Gain(S,A) is the information about the purity change of the examples about the classification on the target using attribute A. k is the number of different values of the attribute currently under consideration and j ranges from 1 to k. A good partition will produce a small expected entropy and, in turn, gain will be larger.

Therefore, choosing the best attribute involves finding the best partition of the set by the nature of *Gain*. The partitioned subsets will be, overall, more pure (more homogenous).

Decision Tree Induction continued on next page...

In SUMMARY, DT Induction is:

- based on a **greedy search strategy** — depth-first & heuristic function and split records based on an attribute test that optimizes certain criterion, and
- has issues on **partitioning data sets** — need to determine *how to split the records* (specifying the attribute test condition and determining the best split) and *when to stop splitting*.

The three major steps of ID3 are:

1. **choosing an attribute for splitting** — calculate information gain for choosing the best attribute,
2. **splitting** — multi-way dividing based on the values of categorical attributes, and
3. **testing** — end or call again at each move (added additional condition: majority voting).

How does one quantify information contained in a message? Information gain and entropy are used to quantify information.

How is this concept applied in a classification method such as ID3? Impurity of classes is investigated in this matter by determining entropy and then information gain. Attributes are evaluated and split based upon a heuristic involving this. As the data set gets smaller, the dataset becomes more pure — a tendency towards a single class.

What is entropy and information gain? See above for the formula.

What is ID3's induction bias? In terms of the nature of impurity, there is this issue that shorter trees are preferred over larger ones. ID3 chooses the first acceptable tree it encounters (greedy search strategy). Trees that place highest information gain attributes close to the root are preferred over those that do not. It has a tendency of choosing an attribute with more distinct values than one with fewer. Because attributes with many values (many divisions or subsets) are preferred, this may lead to **overfitting**.

If, for instance, `record-id`, is used as an attribute and it is an attribute that has a unique value for each tuple in the training set, it is preferred.

The hypothesis space search in DT Induction involves a search space of all possible decision trees. How does one measure the quality of a decision tree? Trees should be *shallow*: decisions are quick to be reached (smaller). This means knowledge is more tense. If leaf nodes are larger, this is also indication of tense knowledge (more power).

Overfitting is about a statistics model that describes random error or noise instead of an underlying relationship. It generally occurs when a model is excessively complex (too many parameters relative to number of observations). A model which has ben overfit will have a poor predictive performance (it can exaggerate minor fluctuations in the data).

To overcome this problem, alternative measures for selective attributes have been proposed including `GainRatio`, used in the C4.5 algorithm. Also, **tree pruning** could be done to make rules more general (short trees). **Pre-pruning** involves, during the process of constructing the tree, stopping splits of subsets of training samples at a given node according to threshold of some statistical measures. **Post-pruning** involves removing branches from a full tree according to the statistical significance of distributed classes.

Bayesian Classification

Decision tree induction involves finding the most probable classification model (hypothesis) given training data where hypotheses are classification models and the number of hypotheses are too many.

Probabilistic prediction invovles finding the most probable classification (**class**) of the new instance given training data — hypotheses here are *classes of a new instance* and the number of hypotheses are few. Classes, for instance, for PlayTennis are {Yes,No}.

Bayesian Classification continued on next page...

How different a classification task is done by DT Induction compared to the naive Bayes classifier? Three differences are:

1. the representation of knowledge differs – decision tree knowledge is a tree which becomes a set of different rules while the Bayes-based approach provides probabilities,
2. considering the hypotheses, the decision tree has a hypothesis in the form of a tree (searching a large space to acquire an optimal tree for classification), while the hypothesis in the Bayes-based approach is not about a tree but about deciding the class of an instance (yes or no), and
3. no cascading occurs during the Bayes algorithm — e.g., there is no bifurcation of a tree and partitioning of training sets.

What are the two assumptions for using NB classifier? One assumption is that among attributes, there is an assumption that they are *independent* (but this is not entirely true; however, for the most part, this will not affect the problem very much). The other assumption is that the quantities of interest are governed by probability distributions and that optimal decisions can be made with reasoning about these probabilities together with observed data.

Why is the naive Bayes algorithm more suitable for high dimensional data? There is a very small search space in terms of a hypothesis — you do not have to worry about the overall complexity of the search space or representation.

The idea of the naive Bayes classifier is to use **Bayes' Rule** to classify by the statement $e \Longrightarrow h$ which is to calculate $P(h|e)$ — the posterior probability in that probability h is true given e.

$$P(h|e) = \frac{P(e|h) \times P(h)}{P(e)}$$
$$P(e) = P(e|h) \times P(h) + P(e|!h) \times P(!h)$$

Bayesian classification provides a quantitative approach to weighing the evidence supporting alternative hypotheses.

Are there any problems with this method? If none of the tuples have a given parameter (e.g., outlook = sunny), this leads to a probability $P(\text{sunny}|\text{yes}) = 0$. This leads to the use of the **M-estimate** for handling missing values in training data. When a probability estimate is zero, this probability term will dominate the Bayes classifier. The m-estimate of probability is on the slides.

Other Methods & Summary

Text classification solves a classification problem in which instances are text documents (training data, target classes, *predict*: label an unseen text to one of the classes). Examples of this application is **spam filtering**, **text document categorization** (placing e-articles into categories), and classifying survey comments into positive or negative classes.

Given an arbitrary text document, each word at every word position is defined as an attribute; this will require a larger number of attributes than short documents but this will not cause any trouble. Text documents is typically considered as *unstructured data*. See slides.

What is the basic idea to convert unstructured text data into structured text data for classification? To make text data be structured, we need to find ways to structure text so that it can be understood by a computer; each text document is treated as an object (**data record**), each word position as an attribute, and words as domain.

Other Methods & Summary continued on next page...

How does one estimate the number of prior probabilities which need to be calculated for text classification? With an assumption that is defined on the slides, estimation can be done by the formula $C \times K$ with K being the number of words and C the number of classes. A commonly used estimator to see how important a keyword is comes with the formula:

$$P(w_k|c_j) = \frac{n_k + 1}{n + Vocabulary}$$

What are the major steps of NB text classification algorithm? The first phase is `Learn_NB_text()`, followed by `Classify_NB_text()`. See slides.

Limitations of the NB method are issues with its assumptions: the **independence assumption** states that the word probability contributes to the text classification has no association with its location in the next (words are independent of each other). This is clearly inaccurate. For non-topic oriented text, there is also a problem.

Sentiment classification requires the knowledge of individual word meanings and relationships between words — in this research, sentiment text features are divided into characteristic words, phrases, and pattern rules which are extracted from the training data. A classification scheme is designed to apply the evidenced sentiment features of input text for optimal decision on the overall sentiment orientation of the data. Example: spam filters. This involves process **Sentiment Knowledge Construction** (SKC).

Classification mining includes two main categories:

- **model based classification**, including DT and Artificial Neutral Networks, and
- **instance based classification**, including Bayesian classifier and nearest neighbor algorithm.

See slides for the concluding information for classification.

Cluster Analysis

Problem Overview

Data clusters are groups of data objects that represent different concepts. These objects are similar to one another within a single cluster and dissimilar to objects in other clusters. For example, this could be customer groups, news discussion groups, web page clusters, etc.

Clustering analysis discovers hidden clusters in a dataset by identifying data objects into different groups based on a **similarity measure** using generic criteria. It can be directly applied to an input dataset and has on need for training data.

What are the differences between clustering, classification, and association? These are three types of data mining problems. In the real world, in terms of data analysis and dealing with large datasets, each of these have a demand. But each of them approach the issue very differently. Classification is supervised (there is training data – samples with known answers of target concept are required) but clustering is not (no prior knowledge and may be viewed as unsupervised generic classification – the target is the data set itself and there are no predefined classes).

In cluster analysis, **intra-cluster distances** are minimized, and **inter-cluster distances** are maximized. Clustering is usually done when the DM objective is less precise. Clustering differs in that its objective is generally far less precise than other predictive and characterization mining. Therefore, it is a great stand-alone tool to get insight into data concepts or distribution. It can also serve as a preprocessing step for other algorithms.

Given a database of n tuples, $D = \{t_1, ..., t_n\}$ and an integer k, the **clustering problem** is to find a mapping $f : D \Longrightarrow \{1, ..., k\}$ where each t_i is assigned to one cluster K_j, $1 \leq j \leq k$. A **cluster** K_j contains precisely those tuples mapped

Problem Overview continued on next page...

to it (with the highest *similarity*). A general similarity method is **Euclidean distance** and unlike the classification problem, clusters are **not known a priori**. Another similarity measure is **Manhattan distance**.

A good clustering method will produce high quality clusters with high intra-class similarity and low intra-class similarity. General requirements of clustering include scalability, discovery of clusters with arbitrary shape, the ability to deal with different types, minimal requirements for domain knowledge, ability to deal with noise and outliers, insensitivity to order of input, high dimensionality, incorporation of user-specified constraints, and interpretability and usability.

The records in a data set need to be **mapped into points** (position vector) in a multidimensional space. How are non-numerical domains treated? Attributes can be non-numerical and must be mapped into points. With a dataset of n objects, two data structures can or will be formed: a **data matrix** ($n \times p$) and a **dissimilarity matrix** ($n \times n$). Many clustering algorithms operate on the dissimilarity matrix directly.

Data Types

The main data types intended for clustering include **interval-scaled variables**, **binary variables**, **categorical variables**, **ordinal variables**, and variables of mixed types. Variable data types can be divided into interval-scaled (numerical), nominal (non-numerical), and mixed.

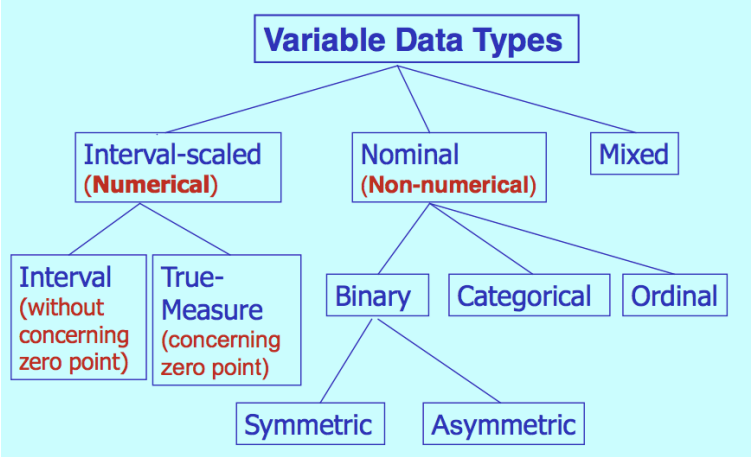

Figure 1: *Hierarchy of variable data types for clustering.*

For clustering DM approach, based on what measure are the objects in a dataset divided into clusters? Each object is represented as a **vector** and the measure is based on Euclidean distance (a similarity measure). Many dimensions can be represented.

How are data types categorized — describe the data type category tree; are they allowed for clustering data mining? A general picture of what types of features or attributes are present for data is important here. Variable data types are separated into numerical and non-numerical (see above) or mixed.

How are non-numerical and mixed data types handled for clustering data mining? The records in a set need to be mapped into points (into a position vector) in multidimensional space. For variables with non-numerical domains, attributes can be non-numerical (binary, nominal, ordinal) and some numeric variables such as **rankings** do not have the right behaviour to properly be treated as components of a position vector.

The dissimilarity $d(i,j)$ between two objects i and j, and for each legitimate variable f, the difference between the value pair x_{if} and x_{jf} is $d_{ij}^{f} = |x_{if} - x_{jf}|$. Giving all variables an equal weight is known as **standardizing** or **normalization**.

Data Types continued on next page...

Interval-valued variables describe the distance between two measure values without concerning a **zero point** (such as temperature, etc.). Ratio has no meaning for internal variables. On the other hand, **true measure variables** are interval variables with a zero point (the ratio of two values is meaningful).

Binary variables have dissimilarity measured between objects of their variable type by considering that two values of binary variables may not have an equal value or importance for an application. Having two variables of such being positive (1 or true), the outcome would be considered more signifiant than of two 0s.

There are two types of binary variables: **symmetric binary attributes** and **asymmetric binary attributes**. In symmetric binary attributes, two values carry the same weight (e.g., gender). In the latter, two of them are not equally important.

NOTE See slides for conversions and data treatment.

From here, dissimilarity matrices are formed.

K-means and K-medoid Methods

The notion of a cluster can be **ambiguous**. A different result can be made for how many clusters given a set of points in multidimensional space. There are two major types of clustering: **partitional clustering** and **hierarchical clustering**. The former divides a data set into non-overlapping subsets (clusters) such that each data object is in exactly one subset. The latter finds a set of nested clusters as a hierarchy tree with two strategies: agglomerative and divisible.

In a partitioning-based approach, this is a **single-level** clustering technique where data is partitioned without a hierarchy. It creates clusters in one stage as opposed to several stages. Since only one set of clusters is output, the user normally has to input the desired number of clusters k.

The **K-means** method randomly chooses an initial set of k-clusters. Iteratively, data objects are moved along clusters until the desired set of clusters is reached. A high degree of similarity among elements in a cluster is obtained (controlled by a stop condition). Given a cluster K_i, the **cluster mean** is $m_i = \frac{1}{p}(t_{i1} + ... + t_{ip})$.

The K-means method is relatively efficient: $O(tkn)$ — generally linear where n is the number of objects, k is the number of clusters, and t is the number of iterations. Normally, $k, t << n$. The weakness, though, is that you need to specify k mean values; unable to handle **outliers** well.