Exploiting the Data Warehouse: Data Mining Techniques

Tema 7

Data Mining

- The analysis of often large data sets to find unsuspected interesting relationships and summarize data in novel ways, understandable and useful to the users
- Single step in a larger process called **knowledge discovery in databases** (or **KDD**)
- **KDD** steps: data cleaning, selection, transformation, reduction, model selection, and exploitation
- Data mining borrows from several scientific fields like artificial intelligence, statistics, neural networks, and other ones
- Data mining requirements
  - Handling of heterogeneous data (e.g., textual, web, spatial, and temporal data, among others)
  - Efficient and scalable algorithms are required
  - Graphical user interfaces are necessary for KDD systems
  - Privacy-aware data mining algorithms must be developed
  - Mining at different abstraction levels is also needed
  - Since data in databases are constantly being modified, discovery methods should be incremental, to allow results to be updated as data change, without needing to rerun the algorithms from scratch.

Data Mining Tasks

- Aimed at discovering models and patterns.
- A **model** is a global summary of a data set
- A simple model can be represented by a linear equation like

$$Y = aX + b$$

where X and Y are variables and a and b are parameters

- **Patterns** make statements about restricted regions of space spanned by the variables.
- Example:

$$\textbf{if}\ \ X > x_1\ \ \textbf{then}\ \ \textbf{prob}(Y > y_1) = p_1.$$

Data Mining Tasks

- **Exploratory Data Analysis** (EDA) uses a variety of graphical techniques to get insight into a data set
  - Techniques are visual and interactive
  - EDA aims at exploring the data without a clear idea of what we are looking for
- **Descriptive modeling** describes the data or the process that generates such data
  - A typical descriptive technique is **clustering**
  - Clustering puts together similar records based on the values of their attributes
- **Predictive modeling** builds models that allow the analyst to predict the value of one variable from the values of other ones
  - Typical techniques are **classification** and **regression**
  - Classification: Predicted variable is categorical
  - Regression: Variable to be predicted is quantitative
- **Pattern discovery** of regular behavior in a data set or records that deviate from regular behavior
  - A typical example: Finding **sequential patterns** in a data set
  - In traffic analysis: Discover frequent routes of cars, trucks, pedestrians

Components of Data Mining Algorithms

- **Model** or **pattern**: For determining the underlying structure in the data
- **Score function**: To assess the quality of the model
- **Optimization** and search methods: To optimize score function and search models and patterns
- **Data management strategies**: To handle data access efficiently during search and optimization

Supervised Classification

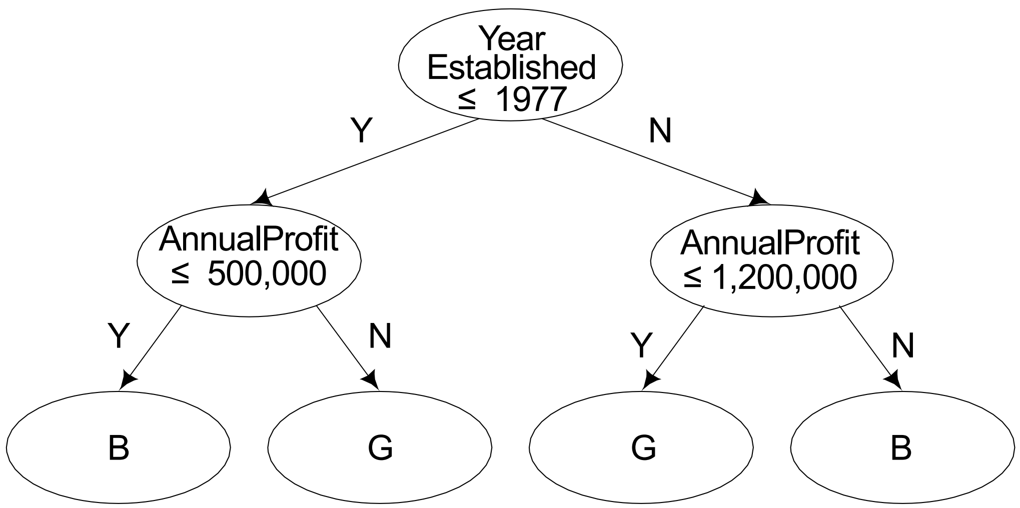
- Allocates a set of objects in a database to different predefined classes according to a model built on the attributes of these objects
- A database *DB* is split into a **training set** *E* and a **test set** *T*
- Tuples of *DB* and *T* have the same format; tuples in *E* have an additional field: the **class identity**, with the class of each tuple in *E*
- These classes are used to generate a model to be used for classifying new data
- Once the model is built using the training set, correctness evaluated using the test set
- Classification methods borrowed from statistics and machine learning
- Most popular methods based on **decision trees**
- **Decision tree**: a **root node** and **internal nodes**, and **leaf or terminal nodes**, with exactly one incoming edge and no outgoing edges
- Each leaf node is assigned a class label
- Nonterminal nodes contain attribute test conditions to split records

Supervised Classification: Example

- Classify customers as good or bad ones: G and B
- We use two of the demographic characteristics: year the business was established, annual profit
- We have these data

YearEstablished	AnnualProfitCont	Class
1977	1,000,000	G
1961	500,000	B
1978	1,300,000	B
1985	1,200,000	G
1995	1,400,000	B
1975	1,100,000	G

- Possible decision tree



Supervised Classification: ID3 Algorithm

INPUT: A data set T OUTPUT: A classification tree

BEGIN

1. Build an initial tree from the training data set T
   IF all points in T belong to the same class THEN RETURN;
   Evaluate splits for every attribute;
   Use the best split for partition T into T1 and T2;
   ID3(T1);
   ID3(T2);
2. Prune this tree to increase test accuracy.
   This removes branches that are likely to induce errors.

END

Supervised Classification

- A key challenge: how to partition the tree nodes
- Attributes conveying more information are selected first
- Information conveyed by a piece of data measured using **Gini index** (entropy also used)
- The **Gini index** for a data set is: $Gini(T) = 1 - \sum_{i=1}^{C} p_i^2$

  where p_i is the relative frequency of class *i* in the data set T, whose elements are classified into *C* classes. T contains *n* samples
- If a split divides T into T1 and T2, with sizes n_1 and n_2:

$$Gini_{Split}(T) = \frac{n_1}{n} Gini(T1) + \frac{n_2}{n} Gini(T2)$$

- Example: Splitting the node using YearEstablished <= 1977 yields T1 containing one record in class B and 2 in G, and T2 with 2 records in class B and 1 record in class G

$$Gini(T1) = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.444$$

$$Gini(T2) = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 0.444$$

$$Gini_{YearEstablished \leq 1997}(T) = \frac{3}{6}(0.444) + \frac{3}{6}(0.444) = 0.444$$

Exercise: Compute the Gini index for YearEstablished <= 1977 and for AnnualProfit <= 1000000

Clustering

- A.k.a. **unsupervised classification**: Groups objects into classes of similar ones
- Classes defined as collections of objects with **high intraclass** similarity and **low interclass** similarity
- Example: group similar customers low interclass
- Most popular clustering methods based on similarity or distance between data points, typically Euclidean distance

Score Function:

- Call $d(x, y)$ the distance between two points x and y in a cluster C_k, with center r_k
- Cluster configuration $\mathbb{C} = C_1, \ldots, C_k$
- **Within cluster variation** measures the intraclass similarity
- First computed for each cluster, and then for the whole clustering configuration

$$wc(C_k) = \sum_{x \in C_k} d(x, r_k)^2 \qquad wc(\mathbb{C}) = \sum_{k=1}^{K} wc(r_k)$$

- **Between cluster variation**: distance between cluster centers

$$bc(\mathbb{C}) = \sum_{1 \le j \le k \le K} d(r_j, r_k)^2$$

Clustering

- Assigns a set c of points to clusters, minimizing/maximizing a score function
- **K-means**: Typical clustering algorithm, from which many different variants are built

K-Means Algorithm

INPUT: A data set T containing n data points $(x_1, \ldots, x_n)$
OUTPUT: A set of K clusters $C_1, \ldots, C_K$
BEGIN
    FOR $k = 1, \ldots, K$ let r_k be a randomly chosen point in T ;
    WHILE changes in clusters C_k happen DO
        /* Form clusters */ FOR
        $k = 1, \ldots, K$ DO
            $C_k = \{x_i \in T \mid d(r_k, x_i) \leq d(r_j, x_i)\ \forall\ j = 1, \ldots, K, j \mathrel{!=} k\}$;
        END;
        /* Compute new cluster centers */
        FOR $k = 1, \ldots, K$; DO
            r_k = the vector mean of the points in C_k ;
        END;
    END;
END;

Example: http://shabal.in/visuals/kmeans/1.html

Clustering

- Several enhancements and variations of the classic clustering method
- **Hierarchical clustering** reduces or increases iteratively the number of clusters of a given model. In the first case, we have
- Called **agglomerative** and **divisive** methods, respectively

Agglomerative Algorithm

INPUT: A data set T containing n data points $(x_1, \ldots, x_n)$.
    A function $d(C_i, C_j)$ to measure the distance between clusters.
OUTPUT: A set of K clusters $C_1, \ldots, C_K$
BEGIN
    FOR $k = 1, \ldots, n$ let $C_i = \{x_i\}$;
    WHILE there is more than one cluster left DO
        Let C_i and C_j be the clusters minimizing the distance between all pairs of clusters;
        $C_i = C_i \cup C_j$;
        Remove C_j;
    END;
END;

Example: https://www.youtube.com/watch?v=XJ3194AmH40

Min. 4:30

Exercises (I)

Consider the following training data about students:

StudID	Age	Country	FamilyIncome	Distance	Finish
s1	1	local	low	1	1
s2	0	local	medium	0	0
s3	0	local	high	1	0
s4	0	local	medium	1	0
s5	4	foreigner	medium	2	1
s6	3	foreigner	medium	1	1
s7	3	foreigner	low	1	2
s8	2	foreigner	low	1	3
s9	1	local	high	2	3
s10	0	local	high	1	2

where the classes are as follows:

- **Age** indicates the age at which the student started the studies. Possible values are as follows: 0 (between 17 and 21), 1 (between 22 and 26), 3 (between 27 and 32), and 4 (older than 32).
- **Country** can have two values: local and foreigner .
- **FamilyIncome** can be low , medium , and high .
- **Distance** indicates the distance that the student has to travel to go to university. It can take values 0 (less than 1 mile), 1 (between 1 and 3 miles), and 2 (more than 3 miles).
- **Finish** indicates whether the student finished her studies in the years planned for the corresponding career. It can take the values: 0 (the student finished her studies on time), 1 (the student finished at most with 1-year delay), 2 (the student finished with 2 or more years of delay), and 3 (the student abandoned her studies).

1. Manually run the ID3 algorithm to build a decision tree over the class Finish. Use the Gini index to partition the nodes.
2. Use the K-means algorithm to generate three clusters of students.

Association Rules

- Association analysis aims at discovering interesting relationships hidden in large data sets
- Very popular technique for market basket analysis, for example, in the retail industry
- $\mathrm{I} = \{i_1, i_2, \ldots, i_m\}$ a set of literals, called **items**. A set of items is called an **itemset**
- D a set of transactions; each transaction T is an itemset such that $T \subseteq \mathrm{I}$
- X an itemset. T contains X if and only if $X \subseteq T$
- An **association rule** is an implication of the form $X \Rightarrow Y$, where $X \subset \mathrm{I}$, $Y \subset \mathrm{I}$, and $X \cap Y = \emptyset$
- The rule $X \Rightarrow Y$ holds in D with **confidence** c, if c% of the transactions in D that contain X also contain Y
- The rule $X \Rightarrow Y$ has **support** s in D if s% of the transactions in D contain $X \cup Y$
- Example: a table of transactions

TransactionId	Items
1000	{1,2,3}
2000	{1,3}
3000	{1,4}
4000	{2,5,6}

- From this data set we obtain:
  - $1 \Rightarrow 3$ with support 50% and confidence 66% ($c = \frac{2}{4}$ and $s = \frac{2}{3}$)
  - $3 \Rightarrow 1$ with support 50% and confidence 100%

Association Rules

- Algorithms based on two steps:
  (1) Generate the **frequent itemsets**, which finds all the itemsets that satisfy a minimum support (*minsup*) threshold.
  (2) Generate the association rules, which extracts all the high-confidence rules from the frequent item- sets found in the previous step. These are called **strong rules**
- The most well-known algorithm: **Apriori** algorithm
- Generates, in the i-th iteration, the set of **candidate itemsets** of length i (C_i), and prunes the ones that do not satisfy the minimum required support
- From C_i creates the **large itemsets** L_i, used for finding candidate itemsets with length $i + 1$
- To **prune** these sets, the **Apriori principle** is applied: **if an itemset is frequent, all of its subsets must also be frequent**
- Example: $\{A, B\}$ cannot be a frequent itemset if either A or B are not frequent

Apriori Algorithm: Example

- Assume minimum support required is $minsup = 50\%$

Item	Count
1	3
2	2
3	2
4	1
5	1
6	1

- Initially, every item is a candidate 1-itemset C_1
- Only items 1, 2, and 3 have support at least equal to *minsup* $\rightarrow$ we delete the other items to obtain the set of large 1-itemsets L_1
- With this set we generate the new candidate itemset table C_2

Item	Count
{1,2}	1
{1,3}	2
{2,3}	1

- The 2-itemset that satisfies *minsup* is {1,3}
- We cannot generate 3-itemsets $\rightarrow$ we stop here
- Two rules generated: $1 \Rightarrow 3$ and $3 \Rightarrow 1$

Hierarchical Association Rules

- Assume now that in the transaction database above, items 1 and 2 belong to category A, items 3 and 4 to category B, and items 5 and 6 to category C.

TransactionId	Items
1000	{A,A,B}
2000	{A,B}
3000	{A,B}
4000	{A,C,C}

- If *minsup* = 75% on the original database, no rules produced
- Categories A and B have support 1 and 0.75, respectively
- Then, aggregating items over categories results in the rules A $\Rightarrow$ B and B $\Rightarrow$ A

Association Analysis: Enhancements

- Efficiency of the association analysis process can be enhanced by:
  - Database scan reduction: storing candidate itemsets in main memory
  - Sampling: If mining is required frequently, sampling can improve performance, with reasonable accuracy cost
    - The reduction factor must be considered when computing confidence and support
    - A **relaxation factor** is calculated according to the size of the sample
  - Parallel data mining. Several algorithms supporting parallelism have been developed to take advantage of parallel architectures
  - Incremental updating of association rules, to avoid repeating the whole mining process

Association Analysis: Fast Update

- First proposed for incremental mining of association rules
- Handles insertions, not deletions, although enhanced in sequel versions
- Database *DB*, frequent itemsets $L = \{L_1, \ldots, L_k\}$
- Incremental database *db* with the new records
- The goal of FUP: Reuse information to efficiently obtain the new frequent itemsets $L^t = \{L^t_1, \ldots, L^t_k\}$ over the database $DB^t = DB \cup db$
- D, d: Number of transactions of *DB* and *db*
- $X.s_{DB}$: Support of itemset X over *DB*
- Based on the following rules:
  - A 1-itemset X frequent in *DB* (that is, $X \in L_1$) becomes infrequent in DB^t (that is, $X < L^t_1$) if and only if $X.s_{DB^t} < minsup \times (D + d)$.
  - A 1-itemset X infrequent in *DB* (that is, $X < L_1$) may become frequent in DB^t (that is, $X \in L^t_1$) if and only if $X.s_{db} < minsup \times d$.
  - A k-itemset X whose $(k - 1)$-subsets become infrequent (that is, the subsets are in L_{k-1} but not in L^t_{k-1}) must be infrequent in *db*.

Association Analysis: Fast Update

- Like in Apriori, the k-th iteration, db is scanned exactly once
- The original frequent itemsets $\{X | X \in L_k\}$ only have to be checked against db
- The set of candidate itemsets C_k first extracted from db, then pruned using the rules in previous slide
- We show example for 1-itemsets
- To compute L_1^t in DB^t:
  - Scan db for all itemsets $X \in L_1$, and update their support count $X.s_{DB^t}$
  - If $X.s_{DB^t} < minsup \times (D + d)$, X will not be in L_1^t (a **loser**).
  - In the same scan compute C_1 with all the items X in db but not in L_1
  - If $X.s_{db} < minsup \times d$, X cannot be a frequent itemset in the updated database.
  - Scan the original database DB to update the support count for each $X \in C_1$
  - Then, we can generate L_1^t

Fast Update: Example

- Use the previous *DB*, and the following *db*, and *minsup* = 50%:

DB

Item	Count
1	3
2	2
3	2
4	1
5	1
6	1

db

TransactionId	Items
5000	{1,2,4}
6000	{4}

- The count of each item in *db*:

Item	Count
1	1
2	1
4	2

+ Items in $L_1 = I_1 = 1, I_2 = 2, I_3 = 3$

- FUP scans *db* for all itemsets in L_1, and compute their support w.r.t. DB^t

$$I_1.s_{DB^t} = 4 > 0.5 \times 6$$

$$I_2.s_{DB^t} = 3 = 0.5 \times 6$$

$$I_3.s_{DB^t} = 2 < 0.5 \times 6 \text{ (\textit{a loser, dropped})}$$

Association Analysis: Fast Update

- I_3 a loser, I_1 and I_2 included in L^t_1
- The second step computes the candidate set C_1 with all the 1-itemsets in *db* not in L_1, i.e., $I_4 = 4$ in this situation.
- I_4 is in both transactions in *db* $\rightarrow$ $I_4.s_{db} = 2 > 0.5 \times 2$, I_4 added to L^t_1
- Updated support count (in light gray the items *I* with support less than *minsup* $\times$ 6):

Item	Count
1	4
2	3
3	2
4	3
5	1
6	1

Exercises (II)

Consider the following transaction database:

TID	Items
T1	{A,B,C}
T2	{A,B,D}
T3	{B,C}
T4	{D,E,F}
T5	{E,F,G}
T6	{A,C,E}
T7	{A,B,D}
T8	{A,B,C,F}
T9	{A,D,E,F}
T10	{B,C,D,E}

1. Manually run the Apriori algorithm to find out the frequent itemsets and rules with minimum support and confidence of 40%.
2. Use the FUP algorithm to insert the following transactions:

TID	Items
T1	{A,K}
T2	{C,E,K}
T3	{F,G}
T4	{K,L}

Explain the algorithm step by step.