RESEARCH ISSUES CONCERNING ALGORITHMS USED FOR OPTIMIZING THE DATA MINING PROCESS

Ion Lungu[1]
Alexandru Pîrjan[2]

Abstract

In this paper, we depict some of the most widely used data mining algorithms that have an overwhelming utility and influence in the research community. A data mining algorithm can be regarded as a tool that creates a data mining model. After analyzing a set of data, an algorithm searches for specific trends and patterns, then defines the parameters of the mining model based on the results of this analysis. The above defined parameters play a significant role in identifying and extracting actionable patterns and detailed statistics. The most important algorithms within this research refer to topics like clustering, classification, association analysis, statistical learning, link mining. In the following, after a brief description of each algorithm, we analyze its application potential and research issues concerning the optimization of the data mining process. After the presentation of the data mining algorithms, we will depict the most important data mining algorithms included in Microsoft and Oracle software products, useful suggestions and criteria in choosing the most recommended algorithm for solving a mentioned task, advantages offered by these software products.

Keywords: data mining optimization, data mining algorithms, software solutions.

Introduction

Data mining consists in techniques and algorithms used for discovering new patterns, clusters and for classifying different types of data from large datasets. Data mining can also be considered a research area based on computational methods designed and used to extract interesting or useful patterns (or knowledge) from real-word datasets. In the last couple of decades, the technology for generating and storing data greatly evolved and as a consequence the amount of data stored in databases become significant larger. Unfortunately, the process of knowledge and pattern discovery developed at a much smaller extent and so did the human capability of understanding and processing this huge amount of data.

The data mining process represents a viable solution to a variety of tasks such as regression, frequent pattern mining, clustering, classification and association discovery. In this purpose, there are hundreds of algorithms capable of performing these tasks. Data mining analysts, researchers, developers and programmers must be aware by the

[1] PhD, Economic Informatics Department, Academy of Economic Studies, 6, Romana Square, district 1, code 010374, Bucharest, Romania, e-mail: ion.lungu@ie.ase.ro
[2] PhD Candidate Faculty of Computer Science for Business Management, Romanian-American University, 1B Expozitiei Blvd., district 1, code 012101, Bucharest, Romania, e-mail: alex@pirjan.com

importance of understanding the way that algorithms work and the possibilities of applying them.

A data mining algorithm can be regarded as a tool that creates a data mining model. In order to achieve this, after analyzing a set of data, an algorithm first searches for specific trends and patterns. The algorithm defines the parameters of the mining model based on the results of this analysis. These parameters, applied across the entire data set, play a significant role in identifying and extracting actionable patterns and detailed statistics.

The optimization of specific algorithms deeply affects the data mining process and therefore developers have implemented a large number of data mining algorithms. Obviously, there is no appropriate algorithm for all applications, domains or databases. Data mining researchers and users must choose which is the most suitable algorithm for solving the problem at hand. Data mining incorporates a broad area of scientific fields' techniques like statistics, machine learning, artificial intelligence, pattern recognition.

In this paper we present a few of the most important data mining algorithms, whose utility and influence are overwhelming in the research community. After a brief description of each algorithm, we analyze its application potential and research issues concerning the optimization of the data mining process. The most important algorithms within this research refer to topics like clustering [1], classification [2], association analysis [3], statistical learning [4], link mining [5].

The k-means algorithm

This algorithm has been developed by researchers Lloyd [6], Forgey, Friedman, Rubin and McQueen. In 1967 James MacQueen used for the first time the term "k-means" but the idea was first suggested in 1956 by Hugo Steinhaus. Stuart Lloyd introduced for the first time the standard algorithm in 1957 as a technique for pulse-code modulation.

The k-means algorithm iteratively partitions a given dataset (containing n observations) into a users' specified number of clusters (denoted by k) in which each observation belongs to the cluster with the nearest mean.

Consider a set $D = \{x_i | i = 1, \ldots, N\}$ of d-dimensional vectors, where each $x_i \in \mathbf{R}^d$ and $d \in \mathbb{N}, N \in \mathbb{N}$. The initial k cluster representatives consists of k points in $\mathbf{R}^d$ and are used in order to initialize the algorithm. This are called centroids. After this stage, the algorithm keeps iterating two steps until convergence is reached:

- Step 1 – Data Assignment. In this step, a partitioning of the data occurs by assigning each data point to its closest centroid.
- Step 2 – Relocation of "means". For each representative cluster, relocation takes place in order to position each such cluster into the center of all data points that are assigned to it.

When no more changes in assignments take place, the algorithm is convergent. Each of the iterations needs $N \times k$ comparisons that determine the time complexity of one iteration.

The term “closest” used in Step 1 refers to the Euclidean distance:

$$\sum_{i=1}^{N}\left(\underset{j}{\operatorname{argmin}}\left(\left\|x_i-c_j\right\|_2^2\right)\right)$$

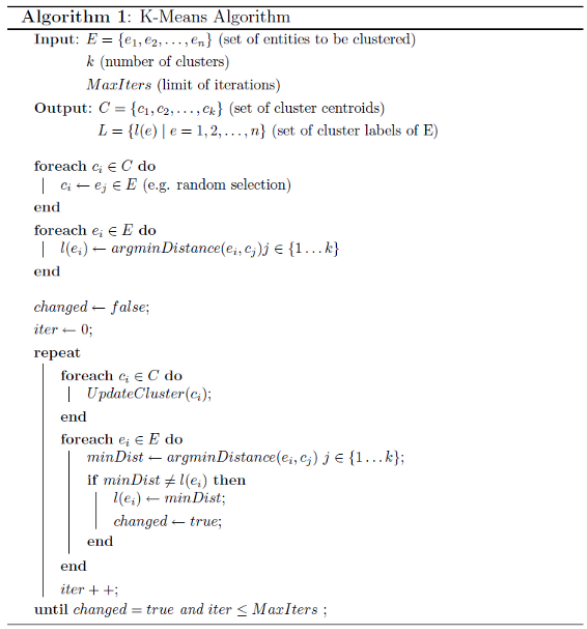
This function decreases whenever the assignment or the relocation steps changes [5] and so convergence is obtained in a finite number of iterations. The convergence occurs to a local optimum and is influenced by the initial chosen centroids. In order to overcome this inconveniences one performs a local limited search of the converged solution or he can run the algorithm multiple times, choosing each time different initial centroids.

Besides this difficulty, the k-means algorithm is prone to several other problems, one of them being the cluster model. The whole concept resides on spherical clusters that must be separable so that the mean value converges towards the center of the cluster. In order for the assignment to the nearest cluster to be correct, clusters must be of similar size. This problem can be overcome by subjecting the data to a rescaling process before clustering. The developer can also use a different distance measure that is more suitable for the dataset. Banerjee A. has shown that if, during the assignment step, distance is measured by selecting any member of a very large class of divergences (called Bregman divergences) without making other changes, the most important properties of k-means are retained (scalability, linear separation boundaries, convergence etc). If an appropriate divergence is used, the k-means algorithm is effective for a much larger class of datasets [6].

Another approach is to perceive the “means” as probabilistic models and not points in $\mathbf{R}^d$ In this case, the Step 1 assigns every data point to the most probable model that could have generated it. The Step 2 updates the model’s parameters so that they best fit their assigned datasets. Dhillon, Guan and Kulis suggested a “kernelization” of k-means in order to allow k-means to deal with clusters that are more complex [5]. Even if the clusters’ boundaries are still linear in an implicit high-dimensional space, they may become non-linear if projected back to the original space.

There is a close connection between kernel k-means and spectral clustering. The K-medoid algorithm and the Fuzzy c-means present several similitudes to the k-means, but in the K-medoid, centroids must belong to the data set that is clustered while in the Fuzzy
c-means the algorithm computes fuzzy membership functions for each of the clusters.

The k-means algorithm remains the most widely used and appreciated algorithms for partitioned clustering despite all of its limitations. The algorithm is scalable, easily understandable and can be optimized to process streaming data with little effort. KD-trees and triangular inequality have been employed in order to speed up k-means that it could process very large datasets. Researches optimized the basic algorithms during the years and the k-means gradually increased its effectiveness and relevance in the field of data mining. The pseudocode of the algorithm [5] is presented in **Fig.1.**



Algorithm 1: K-Means Algorithm

Input: $E = \{e_1, e_2, \ldots, e_n\}$ (set of entities to be clustered)
k (number of clusters)
$MaxIters$ (limit of iterations)

Output: $C = \{c_1, c_2, \ldots, c_k\}$ (set of cluster centroids)
$L = \{l(e) \mid e = 1, 2, \ldots, n\}$ (set of cluster labels of E)

```
foreach c_i ∈ C do
  | c_i ← e_j ∈ E (e.g. random selection)
end
foreach e_i ∈ E do
  | l(e_i) ← argminDistance(e_i, c_j) j ∈ {1...k}
end

changed ← false;
iter ← 0;
repeat
    foreach c_i ∈ C do
      | UpdateCluster(c_i);
    end
    foreach e_i ∈ E do
        minDist ← argminDistance(e_i, c_j) j ∈ {1...k};
        if minDist ≠ l(e_i) then
            l(e_i) ← minDist;
            changed ← true;
        end
    end
    iter++;
until changed = true and iter ≤ MaxIters ;
```

Fig.1. The pseudocode of the k-means algorithm.

The C4.5 and its successor See5/C5.0

Classifiers are of paramount importance in the data mining process. The most commonly used tools that construct them are the systems classifiers that take as input a collection of cases. Each case belongs to one of a small number of classes and is characterized by its values for a fixed set of attributes. The system outputs a classifier that can predict with a high degree of accuracy, which is the class the case belongs to.

We focus on C4.5 and its successor C5.0. The C4.5 is a descendant of CLS and ID3 [5]. C4.5 generates classifiers in the form of decision trees (like CLS and ID3) but it can also create classifiers in a rule set form.

The C4.5 first generates an initial tree by applying the divide-and-conquer algorithm, having a given set S of cases as an input. The tree is declared a leaf if all the cases in S belong to the same class or S is small. The leaf is labeled with the class that appears most frequently in S. If this does not happen, a test based on a single attribute is chosen. The test may have two or more outcomes and represents the root of the tree with one branch for each possible outcome of the test. The given set S of cases is partitioned into subsets $S_1, S_2, \ldots$ according to each case and afterwards the same procedure is applied recursively to each subset. The pseudocode of the algorithm is presented in **Fig. 2**.

Algorithm 2: C4.5 Algorithm

1. Check for **base cases**.
2. For each attribute a
   find the **normalized information gain** from splitting on a.
3. Let a_best be the attribute with the
   highest normalized information gain.
4. Create a **decision node** that splits on a_best.
5. Recur on the sublists obtained by splitting on a_best, and add those nodes as children of **node**.

Fig.2. The pseudocode of the C4.5 algorithm.

During the last step, many tests can be run in order to obtain the desired outcome. C4.5 ranks possible tests by using heuristic criteria such as:

- Information gain – used to minimize the total entropy for the subsets $\{S_i\}$, but it proves to be a biased criterion for tests with more possible outcomes.
- The default gain ratio – is used to divide the information gain mentioned above by the test outcomes' information.
  The attributes (numeric or nominal) determine the format of the test outcomes.

Sorting S on the values of a numeric attribute A and choosing the split between successive values that maximizes the criterion above, one can obtain the value of the threshold h. Then, for A there are $\{A \leq h, A > h\}$ [7]. If A is an attribute with discrete values, each value corresponds to one outcome. The values could be grouped into subsets each of them having a single corresponding outcome.

In order to avoid overfitting, the initial tree must be pruned. A pruning algorithm, based on the estimation of an error rate, is used. This error rate is associated with a set of N cases (chosen so that it does not belong to the most frequent class). This method will provide a pessimistic estimate of the error rate. The C4.5 algorithm does not perform calculation of $\frac{E}{N}$ but instead of it determines first the binomial probability that a number of E events to be observed in N trials and second the upper limit of this probability. For this purpose the user specifies a confidence factor.

The pruning process runs from the leaves to the root. For each leaf having N cases and E errors, the estimated error will be N times the pessimistic rate mentioned before. If we consider a subtree, the C4.5 algorithm compares the estimated error if the subtree is replaced by a leaf with the sum of estimated errors of the branches. If the sum of estimated errors of the branches is higher than the estimated error, the subtree will be pruned. In the same way, the algorithm checks the estimated error if one of its branches replaces the subtree, performs a comparation similar with the one in the previous case and after that, the tree is adjusted accordingly. After crossing the tree once, the pruning process is completed.

The information about each class propagates throughout the entire tree and this makes complex decision trees difficult to understand. In order to simplify the information processing, the C4.5 algorithm introduced a list of rules of the form "if A and B and C and ... then class X". This is an alternative formalism based on grouping together the rules

for each class. For classifying a case, it is enough to find the first rule satisfied in terms of meeting the necessary conditions. The case will be assigned to a default class if no rule is satisfied.

When the C4.5 algorithm builds rulesets, it uses the unprunned decision tree. When the tree is crossed from the root to a leaf, each path becomes a prototype rule. The conditions of this rule are all the outcomes along the path and the class of the rule represents the label of the leaf.

Then, the rule is simplified by dropping conditions one by one and determining the effect of this procedure. When a condition is dropped, two numbers may increase: the number of cases covered by the rule (denoted by *N*) and the number of cases that do not belong to the class nominated by the rule (denoted by *E*). In such condition, the pessimistic error rate may decrease. In the next stage, a hill-climbing algorithm is used for dropping conditions until the pessimistic error rate takes the lowest value.

For each class in turn a subset of simplified rules is selected and this completes the process depicted above. It is chosen a class which will be considered as default. The ruleset obtained contains a number of rules much smaller than that of the leaves from the decision tree after the pruning process. The algorithm's rulesets have as a main disadvantage the large amount of resources that they involve, if we refer to CPU processing time and memory.

In 1977 the C4.5 algorithm was replaced by his successor, a commercial system named See5/C5.0. In [7] it is depicted a new version that offers new capabilities and an improved efficiency such as: an ensemble of classifiers constructed by a variant of boosting, new data types, an improvement in what concerns the interpretability of rulesets, their predictive accuracy and the scalability of decision trees and rulesets.

There are still open issues in what concerns the decision trees, such as the obtaining of stable trees or decomposing complex trees into a small collection of simple trees that give the same result as the complex one [5].

The Apriori algorithm

The notion "frequent itemset" refers to those itemsets whose support is greater than some user-specified minimum support. Finding such frequent itemsets from a transaction dataset represents a very popular approach in data mining and the frequent itemset mining is also related to the obtaining of association rules.

In association rule mining, given a set of "itemsets" (for instance, sets of retail transactions, each listing individual items purchased), the algorithm attempts to find subsets which are common to at least a minimum number C (the cutoff, or confidence threshold) of the itemsets. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as "candidate generation") and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

The Apriori algorithm developed by Agrawal and Srikant in 1994 represents an innovative way to find association rules on a large scale, allowing implication outcomes that consist of more than one item and it is based on a minimum support threshold. The Apriori algorithm is used to find frequent itemsets and it uses candidate generation [8].

Apriori assumes that items within a transaction are ordered in lexicographic order. We will denote by F_k the set of frequent k-size itemsets and by C_k candidate itemsets for the same level. Taking into account the fact that items must satisfy the minimum support requirement, Apriori first scans the database for frequent 1-size itemsets and count those items satisfying the above mentioned condition. In the next stage, the algorithm iterates the tree steps that follow and extracts all the frequent itemsets. Briefly, the Apriori algorithm works as follows:

1. Using the frequent itemsets of size *k* Apriori generates C_{k+1}, candidates itemsets of size *k+1* .

2. Apriori then scans the database and calculates the support for each candidate.

3. The algorithm selects all itemsets whose support satisfies the minimum support requirement and add them to F_{k+1}.

The pseudocode of the algorithm is presented in **Fig. 3.**

Algorithm 3: Apriori algorithm

```
F1 = {frequent items of size 1};
for (k = 1; Fk !=ϕ; k++) do begin
    Ck+1 = apriori-gen (Fk); // New candidates generated from Fk
    for all transactions t in database do begin
        C't = subset (Ck+1,t); // Candidates contained in t
        for all candidate c ∈ C't do
            c.count + +; // Increment the count of all candidates
            in Ck+1 that are contained in t
        end
        Fk+1 = {C ∈ Ck+1| c.count ≥ minimum suport}
        //Candidates in Ck+1 with minimum support
    end
end
Answer ∪k Fk ;
```

Fig.3. The pseudocode of the Apriori algorithm.

The apriori-gen function presented in the above pseudocode generates new candidates C_{k+1} from the set of frequent k-size itemsets F_k in two steps: a join step and a prune one. In the join step, the function generates some initial frequent itemsets candidates of size $k+1$, denoted by R_{k+1}. If P_k and Q_k are two k-size frequent itemsets, having in common their first $k-1$ elements, R_{k+1} is the reunion of P_k and Q_k. Denoting by i_l the *l*-th item, than:

$$R_{k+1} = P_k \cup Q_k = \{i_1, i_2, \ldots, i_{k-1}, i_k, i_k'\}$$

$$P_k = \{i_1, i_2, \ldots, i_{k-1}, i_k\},\ Q_k = \{i_1, i_2, \ldots, i_{k-1}, i_k'\},$$

where $i_1 < i_2 < \cdots < i_{k-1} < i_k < i_k'$.

In the second step, the apriori-gen function selects those k-size itemsets in R_{k+1} that are frequent, removes those one that are not and this is how C_{k+1} is created. The procedure is

based on the fact that each *k*-size subset of C_{k+1} can be a subset of a *(k+1)*-size frequent itemset only if it is a frequent subset.

The C_t' function subset mentioned in the above pseudocode locates the transaction *t* and within it finds those candidates of the frequent itemsets. By scanning the database, the Apriori algorithm calculates the frequency just for candidates obtained before. The Apriori reduces the candidates set's size thus achieving an improved performance. The efforts of generating an increased number of candidates and repeatedly scaning the database can lead to a bottleneck in some situations, like a huge number of frequent itemsets or large itemsets or when the minimum support is very low. The Apriori algorithm is characterized by simplicity, ease of implementation and this is the reason why it is often embraced by data miners.

Apriori algorithm represented a starting point for designing more efficient algorithms in order to optimize the frequent itemset mining process. The generating candidates model implemented in Apriori was adopted by a lot of algorithms like those implemented in partitioning, sampling, vertical data format or hash base technique. The frequent pattern growth method (FP-growth) offered Apriori the highest degree of improvement because it has eliminated the candidate generation step [8]. The implemented strategy of FP-growth is "divide and conquer" and is achieved by:

- the construction of a frequent pattern tree (FP-tree) by using a compressed frequent itemsets database.
- obtaining a conditional databases set by dividing the compressed database. Each conditional database has an associated frequent itemset and it is mined separately.

The algorithm scans the database twice. During the first scan, the frequent items are obtained, their frequencies are computed and then the items are sorted in a descending order of their frequencies. The second scan merges each transaction into a prefix tree and counts common items in different transactions. Each of these common items, called node, is associated to an item. A pointer, called node-link, is used for linking nodes with the same label. All the necessary information is stored in a very compact representation, which follows from the fact that are sorted in the descending order of their frequencies and thus, nodes are shared by more transactions if they are closer to the root of the tree. The frequent pattern growth algorithm choses an item (in the increasing order of the frequency) and then calls itself reccursively on the conditional frequent pattern tree for extracting frequent itemsets containing the chosen itemset. Obviously, comparing the Apriori algorithm and its improved version, the frequent pattern growth method (FP-growth), one can observe that the second method is considerably faster than the original.

The researchers in data mining consider that the frequent pattern mining could also be improved and developed by taking into consideration other issues like: the usage of numeric valuable for item, another approach for measures rather than frequency, the incorporation of taxonomy in items, closed itemsets mining**,** the implementation of incremental mining and the usage of richer expressions than itemset [5].

The AdaBoost algorithm

Ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem [9]. Ensemble learning is particularly useful in improving a model's performance characteristics and in choosing the most suitable model for a certain problem. Usually ensemble learning involves multiple learners for solving a particular problem [5].

Ensemble methods offer tremendous potential in solving a problem taking into account that an ensemble has a significantly better generalization ability that a single learner.
In [7] Yoav Freund and Robert Schapire proposed AdaBoost, a very important ensemble method based on a solid theoretical foundation offering very accurate prediction and great simplicity. In the following we will denote by *X* the instance space, by $Y = \{-1, +1\}$ the set of class labels, then consider a weak or base learning algorithm and a training set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) | x_i \in X, y_i \in Y, i = 1, 2, \ldots, m\}$. The AdaBoost algorithm consists of the following steps:

- For all the training examples (x_i, y_i), $i = 1, 2, \ldots, m$ are assigned equal weights. Using the training set *D* and the distribution of the weights at the *t*-th learning round (denoted by D_t), the algorithm generates (by calling the base learning algorithm) a weak or base learner $h_t: X \to Y$.
- Training examples are used in the next stage for testing h_t and increasing the weight of the incorrectly classified examples, obtaining thus the updated weight distribution D_{t+1}.
- The algorithm uses the updated weight distribution and the training set for generating (by calling again the base learning algorithm) another weak learner. The process is repeated T-times.
- By weighting the majority voting of the T weak learners (their weights are determined during the training process) and the final model is obtained.
  The pseudocode of the AdaBoost algorithm is presented in **Fig 4**.

In [7], Freund and Schapire introduced the Ada-Boost.M1 for dealing with multi-class problems. This improved version requires for the weak learners to be strong enough and this requirement is also mandatory on hard distributions that are generated by the algorithm. There is also a version based on decomposing a multi-class task to a series of binary ones, called AdaBoost.MH. In the statistic field there have been also developed versions of AdaBoost algorithm, used for dealing with regression problems.

Algorithm 4: AdaBoost algorithm

Input: Data set $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$;

Base learning algorithm $\mathcal{L}$;

Number of learning rounds T.

Process:

$D_1(i) = 1/m$. % Initialize the weight distribution

for $t = 1, \cdots, T$:

$h_t = \mathcal{L}(\mathcal{D}, D_t)$; % Train a weak learner h_t from $\mathcal{D}$ using distribution D_t

$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i)]$; % Measure the error of h_t

$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$; % Determine the weight of h_t

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(x_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(x_i) \neq y_i \end{cases}$$

$$= \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$ % Update the distribution, where Z_t is

% a normalization factor which enables D_{t+1} be a distribution

end.

Output: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Fig.4. The pseudocode of the AdaBoost algorithm.

In the literature many interesting topics related to AdaBoost have been researched. One of them is the test error which often tends to decrease even after the training error is zero. Schapire et al. argued that AdaBoost can increase the margins further even if the training error has reached zero. Breiman mentioned that a larger margin does not necessarily equal a better generalization. Reyzin and Schapire concluded that the margin-based explanation may still be valid because Breiman had analyzed the minimum margin instead of average or median margin. Based on Viola and Jones's work, X. Wu et al. [5] consider that AdaBoost could be very useful in feature selection, especially when considering that it has solid theoreticalfoundation. Even if current research mainly focuses on images, X. Wu et al. think general AdaBoost-based feature selection techniques are well worth studying.

Support vector machines

Support vector machines (SVM) offer robust and accurate methods in today's machine learning applications. The number of dimension does not influence the outcome and a reduced number of examples are enough for the training. When dealing with a two-class learning task, the SVM tries to identify the best classification function (using a geometrically approach) that distinguishes between the two classes' members.

The function separates linearly the two classes with a separating hyperplane $f(x)$ that passes through their middle. The new instance x_n is classified after the function has been determined by analyzing the sign of the function's value $f(x_n)$. If this value is greater than zero then x_n belongs to the positive class.

The support vector machines find the best function by maximizing the two classes' margin. The shortest distance between the closest data points to a point on the hyperplane corresponds to the margin, so even if there is infinity of hyperplanes, the solution to SVM could be provided only by a few of them. The maximum margin hyperplanes offer the best generalization and classification performance, so support vector machines try to discover it.

In order for the maximum margin hyperplanes to be discovered, a support vector machine classifier tries to maximize the following function with respect to the $\vec{w}$ vectors and the constant b :

$$L_p = \frac{1}{2}\|\vec{w}\| - \sum_{k=1}^{t} \alpha_k y_k (\vec{w} \cdot \vec{x_k} + b) + \sum_{k=1}^{t} \alpha_k$$

where t is the number of training examples, $\alpha_k, k = 1, \ldots, t$ are non-negative numbers and the derivatives of L_p with respect to α_k are zero. The numbers α_k are called the Lagrange multipliers and the function L_p is called the Lagrangian. The vectors $\vec{w}$ and the constant b define the hyperplane.

Based on some parameters α the support vector machines can be modeled as a function class. A parameter h, also known as the VC dimension represents the different capacity in learning of the different function classes. The VC dimension measures the cases when the obtained error rate is zero and so it provides the maximum number of training examples where the function class can be applied to learn flawlessly.

The actual error on the future data depends by the sum of two elements: the training error and the parameter h (the VC dimension). If *h* is minimized, so is the future error, as long as the training error is also minimized.

In [5] it is presented a "soft margin" idea that tries to extend he SVM algorithm. A slack variable ξ_k is introduced for accounting the amount of classification errors by the function $f(x_k)$. The ξ_k variable is interpreted as the distance from the data that has been classified prone to error to the hyperplane $f(x)$. The original objective minimization function can be revised using the total cost that was created by the slack variables.

In order for the support vector machines to work when the training data is not linearly separable a kernel function can be applied if we extend the dot product $\overrightarrow{x_m} \cdot \overrightarrow{x_n}$ through a functional mapping $\Phi(\overrightarrow{x_m})$ of each $\overrightarrow{x_m}$ to a different space $\mathcal{H}$ of larger or infinite dimensions, case in which the equations still hold. Everywhere where appears the dot product $\overrightarrow{x_m} \cdot \overrightarrow{x_n}$, this is replaced with a kernel function defined by the dot product of the vectors transformed by applying Φ, meaning $\Phi(\overrightarrow{x_m}) \cdot \Phi(\overrightarrow{x_n})$. A large class of nonlinear relationship between inputs can be defined by the kernel function.
Support vector machines can also be used to perform numerical calculations such as regression analysis and to learn to rank elements rather than producing a classification for individual elements.
A significant drawback for this method is that support vector machines require a tremendous computational power. In order to solve this problem, one can divide a large optimization problem into smaller ones, chosen in such a way that every small problem depends only on a couple of variables and this optimizes the entire process. The decomposed optimization problems are solved through an iteration process.

The main data mining algorithms implemented in Microsoft and Oracle software products

In the following we will depict the main data mining algorithms implemented in Microsoft and Oracle software products. As mentioned before, a data mining algorithm analyzes a set of data and looks for specific patterns and trends in order to create a model. Based on the results of this analysis, the algorithm defines the parameters of the mining model which are then applied across the entire data set to extract patterns and statistics. A few examples of the multiple forms that data mining models (created by a data mining algorithm) can take are: rules describing how to conduct the transaction in terms of grouping products, a prediction method based on decision trees and used for evaluating whether a customer will buy a product, a forecasting sales model based on mathematical issues, a description of related cases in a dataset based on a set of clusters and others.
A useful and practical tool in data mining is Microsoft SQL Server Analysis Services, which provides several algorithms for data mining solutions [10]. A few of the most important existing algorithm types are included in this tool and described in **Fig. 5**.

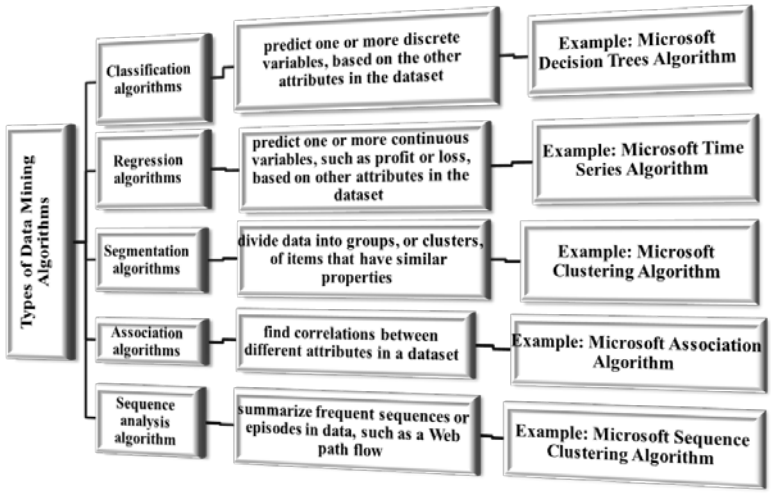


Fig. 5. The most important existing algorithm types included in Microsoft SQL Server Analysis Services.

One of the most important challenges in data mining is the choosing of the most suitable algorithm for solving a specific task and this should be made taking into account some important facts as:

- a business task can be performed using different algorithms
- the result produced by each algorithm is different
- there are some algorithms which can produce more types of results. For example, Microsoft Decision Trees algorithm could be used for prediction and also for reducing the number of columns in a dataset based on the fact that the decision tree identifies columns that do not affect the final mining model.

- for obtaining a single data mining solution, algorithms could be used together, not independently. For example, some algorithms could be used for exploring data and after that, some other algorithms could be used in order to predict a specific outcome based on that data.
- separate tasks could be performed by multiple algorithms within one solution. For example, in order to obtain financial forecasting information one can use a regression tree algorithm first and then a rule-based algorithm that is best suitable to perform a market basket analysis.
- using mining models one can facilitate the prediction of values, the production of data summaries, the finding of some hidden correlations.

Below are presented some suggestions and criteria useful for choosing the most recommended algorithm for solving the mentioned task (**Table 1**).

Task	Example	Microsoft algorithms to use
Predicting a discrete attribute	Predict whether the recipient of a targeted mailing campaign will buy a product	Microsoft Decision Trees Algorithm
		Microsoft Naive Bayes Algorithm
		Microsoft Clustering Algorithm
		Microsoft Neural Network Algorithm
Predicting a continuous attribute	Forecast next year's sales	Microsoft Decision Trees Algorithm
		Microsoft Time Series Algorithm
Predicting a sequence	Perform a clickstream analysis of a company's Web site	Microsoft Sequence Clustering Algorithm
Finding groups of common items in transactions	Use market basket analysis to suggest additional products to a customer for purchase	Microsoft Association Algorithm
		Microsoft Decision Trees Algorithm
Finding groups of similar items	Segment demographic data into groups to better understand the relationships between attributes	Microsoft Clustering Algorithm
		Microsoft Sequence Clustering Algorithm

Table 1. Suggestions and criteria useful for choosing the most recommended algorithm.

One of the options included in Oracle Corporation's Relational Database Management System Enterprise Edition is the **Oracle Data Mining** (ODM) [11], which:

- contains several data mining and data analysis algorithms designed for classification, clustering, prediction, associations, regression, anomaly detection, feature selection, feature extraction and specialized analytics.
- offers facilities for the creation of data mining models, for their management and deployment.
- implements inside the Oracle relational database a variety of data mining algorithms, integrates these implementations into the database's kernel and as a consequence eliminates some operations: the extraction and transfer of data through mining servers.

The secure managing of models and the efficient execution of SQL queries on large volumes of data are facilitated by the database platform. A general unified interface for data mining functions is provided by a few generic operations (around which the system is organized) including functions designed to create data mining models, to apply, test and manipulate them. Created and stored as database objects, these models are managed within the database as all the other contained objects.

After a data mining program generates a model, it could be used by the same data mining program to derive predictions or descriptions of behavior. In order to obtain predictions, the new generated model can be applied to all the new information. The technique implemented in the data mining model and used to make predictions about the studied behavior is called "scoring" and the prediction offered by the model is called "the score". ODM offers some Oracle SQL functions for scoring data stored in the database and therefore one can efficiently use the features offered by Oracle SQL: the ability to pipeline and manipulate the results over several levels and also the parallelizing and partitioning of data access for performance.

The data never leaves the database: data, data preparation, model building and model scoring results remain in the database. This enables Oracle to provide an infrastructure for application developers to integrate data mining seamlessly with database applications.

Oracle Data Mining (ODM) provides a broad suite of data mining techniques and algorithms to solve many types of business problems [11]. These data mining techniques and algorithms are presented below (**Table 2**).

Techniques	**Applicability**	**Oracle Data Mining Mining Algorithms**
Classification	Most commonly used technique for predicting a specific outcome such as response/no-response, high /medium/low-value customer, likely to buy/not buy.	Logistic Regression
		Naive Bayes
		Support Vector Machine
		Decision Tree
Regression	Technique for predicting a continuous numerical outcome such as customer lifetime value, house value, process yield rates.	Multiple Regression
		Support Vector Machine

Attribute Importance	Ranks attributes according to strength of relationship with target attribute. Use cases include finding factors most associated with customers who respond to an offer, factors most associated with healthy patients.	Minimum Description Length
Anomaly Detection	Identifies unusual or suspicious cases based on deviation from the norm. Common examples include health care fraud, expense report fraud, and tax compliance.	One-Class Support Vector Machine
Clustering	Useful for exploring data and finding natural groupings. Members of a cluster are more like each other than they are like members of a different cluster. Common examples include finding new customer segments, and life sciences discovery.	Enhanced K-Means
		Orthogonal Partitioning Clustering
Association	Finds rules associated with frequently co-occuring items, used for market basket analysis, cross-sell, root cause analysis. Useful for product bundling, in-store placement, and defect analysis.	Apriori
Feature Extraction	Produces new attributes as linear combination of existing attributes. Applicable for text data, latent semantic analysis, data compression, data decomposition and projection, and pattern recognition.	Non-negative Matrix Factorization

Table 2. Data mining techniques and algorithms provided by Oracle Data Mining.

Implemented in the Oracle Database kernel, Oracle Data Mining models are first class database objects, while Oracle Data Mining processes maximize scalability and make efficient use of system resources through the usage of built-in features of Oracle Database. Data mining within Oracle Database offers many advantages, as shown in **Table 3.**

The Advantage	**Details**

No Data Movement	• Some data mining products require that the data be exported from a corporate database and converted to a specialized format for mining. • With Oracle Data Mining, no data movement or conversion is needed. • This makes the entire mining process less complex, time-consuming, and error-prone.
Security	• Your data is protected by the extensive security mechanisms of Oracle Database. • Moreover, specific database privileges are needed for different data mining activities. • Only users with the appropriate privileges can score (apply) mining models.
Data Preparation and Administration	• Most data must be cleansed, filtered, normalized, sampled, and transformed in various ways before it can be mined. • Up to 80% of the effort in a data mining project is often devoted to data preparation. • Oracle Data Mining can automatically manage key steps in the data preparation process. • Additionally, Oracle Database provides extensive administrative tools for preparing and managing data.
Ease of Data Refresh	• Mining processes within Oracle Database have ready access to refreshed data. • Oracle Data Mining can easily deliver mining results based on current data, thereby maximizing its timeliness and relevance.
Oracle Database Analytics.	• Oracle Database offers many features for advanced analytics and business intelligence. • Oracle Data Mining can easily be integrated with other analytical features of the database, such as statistical analysis and OLAP. • See "Oracle Data Mining and Oracle Database Analytics".
Oracle Technology Stack	• You can take advantage of all aspects of Oracle's technology stack to integrate data mining within a larger framework for business intelligence or scientific inquiry.
Domain Environment	• Data mining models have to be built, tested, validated, managed, and deployed in their appropriate application domain environments. • Data mining results may need to be post-processed as part of domain specific computations (for example, calculating estimated risks and response probabilities) and then stored into permanent repositories or data warehouses. • With Oracle Data Mining, the pre- and post-mining activities can all be accomplished within the same environment.

Application Programming Interfaces	• PL/SQL and Java APIs and SQL language operators provide direct access to Oracle Data Mining functionality in Oracle Database.

Table 3. Advantages of data mining within Oracle Database.

Conclusions and further research

Data mining comprises techniques and algorithms for determining interesting patterns from large datasets and has applications in a large set of domains including business and science, such as web analysis, targeted marketing, disease diagnosis and outcome prediction, weather forecasting, credit risk and loan approval, customer relationship modeling, fraud detection and many others. The above mentioned techniques cover a wide area of fields integrating statistics, pattern recognition, database systems, artificial intelligence and machine learning in the purpose of analyzing huge amounts of data. In order to perform different data analysis tasks such as clustering, classification, frequent pattern mining and others, researchers have created, developed and implemented hundreds (or even more) data mining algorithms. The understanding of the usage and improvement of those algorithms are real challenges for scientists, analysts, researchers, practitioners.

In this paper we have depicted some of the most widely used data mining algorithms. The main criteria used for choosing them was their utility and influence in the research community. They refer to topics such as: clustering, classification, association analysis, statistical learning, link mining. Each algorithm's study starts with its brief description followed by its application potential and research issues concerning the optimization of the data mining process.

We have also depicted the most important existing data mining algorithms included in Microsoft and Oracle software products, suggestions and criteria useful for choosing the most recommended algorithm for solving a mentioned task, advantages offered by these software products. Both products can be used to build stable, efficient system. The stability, effectiveness of applications and databases depend rather on the experience of the database developers and database administrators than on the database's provider.

An important and interesting issue regarding the data mining process is real time data mining, which will enable scientists to develop researches on a scale that seemed unimaginable until recently [8]. In order to improve data analysis, both hardware architectures and data mining algorithms must properly manage and process huge volumes of data. This improvement could be achieved through the development and implementation of new parallel processing algorithms and novel hardware architectures. In recent years there have been developed new techniques and data mining methods useful for the discovering of new patterns, clusters and for the classifying of different types of data. The optimization of a data mining algorithm must take into account some targets, as the improvement of the data extraction process quality and also the reducing of the response time.

Further research in the field of data mining algorithms used for optimizing the data mining process must refer to both improving existing algorithms and developing new ones, based on modern and powerful hardware architecture. As an example, the Compute

Unified Device Architecture, a revolutionary software and hardware parallel computing architecture from NVIDIA, allows the graphics processor to execute programs written in C, C++, FORTRAN, OpenCL, Direct Compute and other languages. Some algorithms based on the Compute Unified Device Architecture programming model, implementing some kind of parallelism and based all on the MapReduce programming model are depicted in [8], [12].

References

[1] Chen JR (2007) Making clustering in delay-vector space meaningful. Knowl Inf Syst 11(3):369–385

[2] Kukar M (2006) Quality assessment of individual classifications in machine learning and data mining, Knowl Inf Syst 9(3):364–384

[3] Ahmed S, Coenen F, Leng PH (2006) Tree-based partitioning of date for association rule mining. Knowl Inf Syst 10(3):315–331

[4] Fung G, Stoeckel J (2007) SVM feature selection for classification of SPECT images of Alzheimer's disease using spatial information. Knowl Inf Syst 11(2):243–258

[5] Xindong Wu, Vipin Kumar, The Top Ten Algorithms in Data Mining, Chapman and Hall/CRC 2009 ISBN: 978-1-4200-8964-6

[6] Lloyd SP (1957) Least squares quantization in PCM, IEEE, Trans Inform Theory (Special Issue on Quantization), vol IT-28, pp 129–137, March 1982

[7] Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139

[8] Pirjan A., The optimization of algorithms in the process of temporal data mining using the Compute Unified Device Architecture, Database Systems Journal, nr.1/2010 , http://www.dbjournal.ro/

[9] Polikar, R., "Ensemble learning," *Scholarpedia*, vol. 4, no. 1, pp. 2776, 2009

[10] http://technet.microsoft.com/en-us/library/ms175595.aspx

[11] http://download.oracle.com/docs/cd/B28359_01/datamine.111/b28129/intro_concepts.htm

[12] Pirjan A., Algorithms for extracting frequent episodes in the process of temporal data mining, Informatica Economică Journal, Vol. 14 No. 3/2010, 165-169, http://www.revistaie.ase.ro/