DATA MINING PROBLEM SOLVING ALGORITHMS & THEIR COMPARATIVE STUDY

Jeevan Singh Bisht, Christian Eminent College, Christian Eminent College, Indore
Mayushi Chouhan, Student, Acropolis Institute, Indore

Abstract

This paper presents the comparison schemes between mining algos identified by the IEEE International Conference on Data Mining (ICDM) are : C4.5, k-Means, Apriori, FP growth, Page Rank, Ada Boost, kNN, and CART. These algos are among the most influential data mining algorithms in the research community. With each algorithm, we provide a description of the algorithm, comparison between these algos, discuss the impact of the algorithm, and review current and further research on the algos. These algorithms cover classification.

Keywords: KDD, Data mining, association rules, Preprocessing algos, k-means, kNN, CART, ICDM, CLS, C4.5 etc

1. Introduction: Data mining is a broad area that integrates techniques from several fields including machine learning, statistics, pattern recognition, artificial intelligence, and database systems, for the analysis of large volumes of data. There have been a large number of data mining algorithms rooted in these fields to perform different data analysis tasks. In an effort to identify some of the most influential algorithms that have been widely used in the data mining community, the IEEE International Conference on Data Mining (ICDM) identified the algorithms in data mining. The algorithms identified by the IEEE International Conference on Data Mining (ICDM) and presented in this article are among the most influential algorithms for classification [9], clustering [11,4], association analysis [13,7], and link mining. We hope this survey paper can inspire more researchers in data mining to further explore these algorithms, including their impact and new research issues.

1.1 C4.5 and beyond:

1.1.1 Introduction: Systems that construct classifiers are one of the commonly used tools in data mining. Such systems take as input a collection of cases, each belonging to one of a small number of classes and described by its values for a fixed set of attributes, and output a classifier that can accurately predict the class to which a new case belongs. These notes describe C4.5 [6], a descendant of CLS [4] and ID3 [6]. Like CLS and ID3, C4.5 generates classifiers expressed as decision trees, but it can also construct classifiers in more comprehensible rule set form.

1.1.2 Decision tress: Given a set S of cases, C4.5 first grows an initial tree using the divide-and-conquer algorithm as follows:

- If all the cases in S belong to the same class or S is small, the tree is a leaf labeled with the most frequent class in S.

* Otherwise, choose a test based on a single attribute with two or more outcomes. Make this test the root of the tree with one branch for each outcome of the test, partition S into corresponding subsets *S*1, *S*2, . . . according to the outcome for each case, and apply the same procedure recursively to each subset. C4.5 uses two heuristic criteria to rank possible tests: information gain, which minimizes the total entropy of the subsets {*Si* } (but is heavily biased towards tests with numerous outcomes), and the default gain ratio that divides information gain by the information provided by the test outcomes. Attributes can be either numeric or nominal and this determines the format of the test outcomes. The pruning process is completed in one pass through the tree. C4.5's tree-construction algorithm differs in several respects from CART [9], for instance:

- Tests in CART are always binary, but C4.5 allows two or more outcomes.
- CART uses the Gini diversity index to rank tests, whereas C4.5 uses information-based criteria.
- CART prunes trees using a cost-complexity model whose parameters are estimated by cross-validation; C4.5 uses a single-pass algorithm derived from binomial confidence limits.

1.1.3 Rule set classifiers: Complex decision trees can be difficult to understand, for instance because information about one class is usually distributed throughout the tree. C4.5 introduced an alternativeformalism consisting of a list of rules of the form "if A and B and C and ... then class X", where rules for each class are grouped together. A case is classified by finding the first rule whose conditions are satisfied by the case; if no rule is satisfied, the case is assigned to a default class. C4.5 rule sets are formed from the initial (un pruned) decision tree.

1.1.4 **Research issues:** We have frequently heard colleagues express the view that decision trees are a "solved problem." We do not agree with this proposition and will close with a couple of open research.

Problems:

1 Stable trees. It is well known that the error rate of a tree on the cases from which it was constructed is much lower than the error rate on unseen cases.

2 Decomposing complex trees. Ensemble classifiers, whether generated by boosting, bagging, weight randomization, or other techniques, usually offer improved predictive accuracy. Now, given a small number of decision trees, it is possible to generate a single (very complex) tree that is exactly equivalent to voting the original trees, but can we go the other way?

2. The *k*-means algorithm:

2.1 Algorithm: The k-means algorithm is a simple iterative method to partition a given dataset into a user specified number of clusters, *k*. This algorithm has been discovered by several researchers across different disciplines, most notably Lloyd (1982) [15], Forgey (1985), Friedman and Rubin (1997), and McQueen (1967). A detailed history of k-means along with descriptions of several variations is given in [16]. Techniques for selecting these initial seeds include sampling at random from the dataset, setting them as the solution of clustering a small subset of the data or perturbing the global mean of the data *k* times. Then the algorithm iterates between two steps till convergence.

Problems: In addition to being sensitive to initialization, the k-means algorithm suffers from several other problems. First, observe that k-means is a limiting case of fitting data by a mixture of *k* Gaussians with identical, isotropic covariance matrices (_ = *ó*2**I**), when the soft assignments of data points to mixture components are hardened to allocate each data point solely to the most likely component. So, it will falter whenever the data is not well described by reasonably separated spherical balls

3 The Apriori algorithm

3.1 Description of the algorithm: One of the most popular data mining approaches is to find frequent item sets from a transaction dataset and derive association rules. Finding frequent item sets (item sets with frequency larger than or equal to a user specified minimum support) is not trivial because of its combinatorial explosion. Once frequent item sets are obtained, it is straightforward to generate association rules with confidence larger than or equal to a user specified minimum confidence. Apriori is a seminal algorithm for finding frequent item sets using candidate generation [1]. It is characterized as a level-wise complete search algorithm using anti-monotonicity of item sets, "if an item set is not frequent, any of its superset is never frequent". By convention, Apriori assumes that items within a transaction or item set are sorted in lexicographic order. Let the set of frequent item sets of size *k* be *Fk* and their candidates be *Ck* . Apriori first scans the database and searches for frequent item sets of size 1 by accumulating the count for each item and collecting those that satisfy the minimum support requirement. It then iterates on the following three steps and extracts all the frequent item sets.

1. Generate *Ck*+1, candidates of frequent item sets of size *k* +1, from the frequent item sets of size *k*.
2. Scan the database and calculate the support of each candidate of frequent item sets.
3. Add those item sets that satisfies the minimum support requirement to *Fk*+1. The Apriori algorithm is shown in Fig.

Algorithm 1 *Apriori*

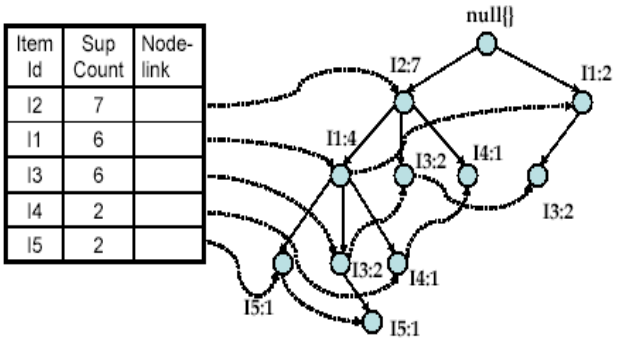
F_l=(Frequent itemsets of cardinality 1);
for($k = 1; F_k \neq \phi; k++$) do begin
  C_{k+1} = apriori-gen(F_k); //New candidates
  for all transactions $t \in$ Database do begin
    C'_t = subset(C_{k+1}, t); //Candidates contained in t
    for all candidate $c \in C'_t$ do
      $c.count++$;
    end
    $F_{k+1} = \{C \in C_{k+1} \quad | c.count \geq$ minimum support $\}$
  end
end
Answer $\cup_k F_k$;

3.2 The impact of the algorithm: Many of the pattern finding algorithms such as decision tree, classification rules and clustering techniques that are frequently used in data mining have been developed in machine learning research community. Frequent pattern and association rule mining is one of the few exceptions to this tradition.. The algorithm is quite simple and easy to implement. Experimenting with Apriori-like algorithm is the first thing that data miners try to do.

3.3 Current and further research: Since Apriori algorithm was first introduced and as experience was accumulated, there have been many attempts to devise more efficient algorithms of frequent itemset mining. Many of them share the same idea with Apriori in that they generate candidates. These include hash-based technique, partitioning, sampling and using vertical data format. Hash-based technique can reduce the size of candidate itemsets. Each itemset is hashed into a corresponding bucket by using an appropriate hash function. Since a bucket can contain different itemsets, if its count is less than a minimum support, these itemsets in the bucket can be removed from the candidate sets. A partitioning can be used to divide the entire mining problem into *n* smaller problems. The dataset is divided into *n* non-overlapping partitions such that each partition fits into main memory and each partition is mined separately. Since any itemset that is potentially frequent with respect to the entire dataset must occur as a frequent itemset in at least one of the partitions, all the frequent itemsets found this way are candidates, which can be checked by accessing the entire dataset only once.

4 The FP growth algorithm: The most outstanding improvement over Apriori would be a method called FP-growth (frequent pattern growth) that succeeded in eliminating candidate generation [6]. It adopts a divide and conquer strategy by (1) compressing the database representing frequent items into a structure called FP-tree (frequent pattern tree) that retains all the essential information and (2) dividing the compressed database into a set of conditional databases, each associated with one frequent itemset and mining each one separately.

FP-Growth Method: Construction of FP-Tree



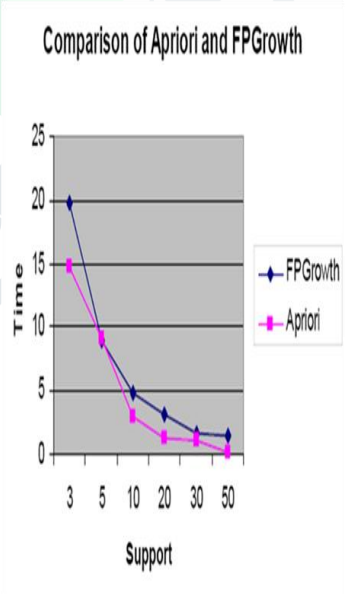
An FP-Tree that registers compressed, frequent pattern information

It scans the database only twice. In the first scan, all the frequent items and their support counts (frequencies) are derived and they are sorted in the order of descending support

count in each transaction. In the second scan, items in each transaction are merged into a prefix tree and items (nodes) that appear in common in different transactions are counted. Each node is associated with an item and its count. Nodes with the same label are linked by a pointer called node-link. Pattern growth algorithm works on FP-tree by choosing an item in the order of increasing frequency and extracting frequent itemsets that contain the chosen item by recursively calling itself on the conditional FP-tree. FP-growth is an order of magnitude faster than the original Apriori algorithm. There are several other dimensions regarding the extensions of frequent pattern mining. The major ones include the followings: (1) incorporating taxonomy in items [2]: Use of taxonomy makes it possible to extract frequent itemsets that are expressed by higher concepts even when use of the base level concepts produces only infrequent itemsets. (2) incremental mining: In this setting, it is assumed that the database is not stationary and a new instance of transaction keeps added. The algorithm in [12] updates the frequent itemsets without restarting from scratch. (3) using numeric valuable for item: When the item corresponds to a continuous numeric value, current frequent itemset mining algorithm is not applicable unless the values are discretized. A method of subspace clustering can be used to obtain an optimal value interval for each item in each itemset [5]. Thus, once the closed itemsets are found, all the frequent itemsets can be derived from them. LCM is the most efficient algorithm to find the closed itemsets [8].

4.1 Comparison between Aprior & FP growth: Mining process of spatial association rule used Apriori based_and FP-Growth algorithms. The reason of using of both algorithms are widely used of those algorithms and using of two different approaches. Some interesting patterns got from mining process are:

IF Population_Density_Low
THEN DBD_Low (0.61) (0.75)
IF Health_Facility_No
AND Population_Density_Low
THEN DBD_Low (0.47) (0.8)
IF Health_Facility_No
AND Population_Density_Low
AND Close-to_Bog
THEN DBD_Low (0.86) (0.86) Etc



5 Page Rank:

5.1 Overview: Page Rank [1] was presented and published by Sergey Brin and Larry Page at the Seventh International World Wide Web Conference (WWW7) in April 1998. It is a search ranking.

5.2 The algorithm: We now introduce the Page Rank formula. Let us first state some main concepts in the Web context. In-links of page *i* : These are the hyperlinks that point to page *i* from other pages. Usually, hyperlinks from the same site are not considered.
Out-links of page *i*
i. A hyperlink from a page pointing to another page is an implicit conveyance of authorit to the target page. Thus, the more in-links that a page i receives, the more prestige the page *i* has.

ii. Pages that point to page *i* also have their own prestige scores. A page with a higher prestige score pointing to *i* is more important than a page with a lower prestige score pointing to *i* . In other words, a page is important if it is pointed to by other important pages.
PageRank-Iterate(*G*)
P0 ←□***e****/n k* ←□1
repeat
(1) ; *k* -1
T k ***P*** ←□−□*d* ***e*** □□*d****A*** ***P***
k ←□*k* + 1;
until ||***P****k* – ***P****k*-1||1 < □
return ***P****k*

The largest eigen value and the Page Rank vector *P* is the principal eigenvector. A well known mathematical technique called power iteration [3] can be used to find *P*. However, the problem is that Eq. (14) does not quite suffice because the Web graph does not meet the conditions. In fact, Eq. (14) can also be derived based on the Markov chain. Then some theoretical results from Markov chains can be applied. After augmenting the Web graph to satisfy the conditions, the following PageRank equation is produced:

P = *(*1 − *d)***e** + *d***AT****P**, (15)
where **e** is a column vector of all 1's. This gives us the Page Rank formula for each page *i*
P(i) = *(*1 − *d)* + *dn_ j*=1
Aji P(j), (16)
which is equivalent to the formula given in the original PageRank papers [10,6]:
P(i) = *(*1 − *d)* + *d* _ *(j,i)*∈*E P(j) Oj* . (17)
The parameter *d* is called the *damping factor* which can be set to a value between 0 and 1. *d* = 0.85 is used in [10,52]. The computation of PageRank values of the Web pages can be done using the power

6 *k*NN: *k*-nearest neighbor classification

6.1 Description of the algorithm: One of the simplest, and rather trivial classifiers is the Rote classifier, which memorizes the entire training data and performs classification only if the attributes of the test object match one of the training examples exactly. There are three key elements of this approach: a set of labeled objects, e.g., a set of stored records, a distance or similarity metric to compute distance between objects, and the value of *k*, the number of nearest neighbors.
Given a training set *D* and a test object *x* = *(***x**_, *y_)*, the algorithm computes the distance (or similarity) between *z* and all the training objects *(***x**, *y)* ∈ *D* to

determine its nearest-neighbor list, *Dz*. (**x** is the data of a training object, while *y* is its class. Likewise, **x**_ is the data of the test object and *y*_ is its class.)
Input: the set of training objects and test object **x**
Process: Compute **x** , **x** , the distance between& every object, **x**
Select , the set of closest training objects to .
Output: arg max **x**

7 CART: The 1984 monograph, "CART: Classification and Regression Trees," co-authored by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone, [9] represents a major milestone in the evolution of Artificial Intelligence, Machine Learning, non-parametric statistics, and data mining. The work is important for the comprehensiveness of its study of decision trees, the technical innovations it introduces, its sophisticated discussion of tree structured data analysis, and its authoritative treatment of large sample theory for trees.

7.1 Overview: The CART decision tree is a binary recursive partitioning procedure capable of processing continuous and nominal attributes both as targets and predictors. Data are handled in their raw form; no binning is required or recommended. Trees are grown to a maximal size without the use of a stopping rule and then pruned back (essentially split by split) to the root via cost-complexity pruning. The next split to be pruned is the one contributing least to the overall performance of the tree on training data (and more than one split may be removed at a time). The final reports include a novel attribute importance ranking.

7.2 Prior probabilities and class balancing: In its default classification mode CART always calculates class frequencies in any node relative to the class frequencies in the root. This is equivalent to automatically reweighting the data to balance the classes, and ensures that the tree selected as optimal minimizes balanced class error. The reweighting is implicit in the calculation of all probabilities and improvements and requires no user intervention; the reported sample counts in each node thus reflect the unweighted data. For a binary (0/1) target any node is classified as class 1 if, and only if, $N1(node)/N1(root) > N0(node)/N0(root)$. (18) This default mode is referred to as "priors equal" in the monograph. It has allowed CART users to work readily with any unbalanced data, requiring no special measures regarding.

8 Concluding remarks: Data mining is a broad area that integrates techniques from several fields including machine learning, statistics, pattern recognition, artificial intelligence, and database systems, for the analysis of large volumes of data. There have been a large number of data mining algorithms rooted in these fields to perform different data analysis tasks. We hope this survey paper can inspire more researchers in data mining to further explore these algorithms, including their impact and new research issues.

9. References:

1. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th VLDB conference, pp 487–499
2. Ahmed S, Coenen F, Leng PH (2006) Tree-based partitioning of date for association rule mining. Knowl Inf Syst 10(3):315–331
3. Gray RM, Neuhoff DL (1998) Quantization. IEEE Trans Inform Theory 44(6):2325–2384.
4. Hart P (1988) The condensed nearest neighbor rule. IEEE Trans Inform Theory 14:515–516
5. Banerjee A, Merugu S, Dhillon I, Ghosh J (2005) Clustering with Bregman divergences. J Mach Learn Res 6:1705–1749
6. Bezdek JC, Chuah SK, Leep D (2002) Generalized k-nearest neighbor rules. Fuzzy Sets Syst 18(3):237–256. http://dx.doi.org/10.1016/0165-0114(86)90004-7
7. Bloch DA, Olshen RA, Walker MG (2002) Risk estimation for classification trees. J Comput Graph Stat 11:263–288
8. Bonchi F, Lucchese C (2006) On condensed representations of constrained frequent patterns. Knowl Inf Syst 9(2):180–201
9. Breiman L (1968) Probability theory. Addison-Wesley, Reading. Republished (1991) in Classics of mathematics. SIAM, Philadelphia
10. Dasarathy BV (ed) (1991) Nearest neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press
11. Breiman L (2005) Prediction games and arcing classifiers. Neural Computer 11 (7):1493–1517
12. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees. Wadsworth, Belmont
13. Chi Y, Wang H, Yu PS, Muntz RR (2006) Catch the moment: maintaining closed frequent itemsets over a data stream sliding window. Knowl Inf Syst 10(3):265–294
14. Cost S, Salzberg S (1993) A weighted nearest neighbor algorithm for learning with symbolic features. Mach Learn 10:57.78 (PEBLS: Parallel Examplar-Based Learning System)
15. Cover T, Hart P (1987) Nearest neighbor pattern classification. IEEE Trans Inform Theory 13(1):21–27
16. Dasarathy BV (ed) (1991) Nearest neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society Press
17. Data Pre-processing & Mining Algorithm, Knowledge & Data Mining & Preprocessing, 3rd edition, *Han & Kamber*.
18. Mohd Helmy Abd Wahab, Mohd Norzali Haji Mohd, Hafizul Fahri Hanafi, Mohamad Farhan(1998) Mohamad Mohsin
19. Apriori: Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In VLDB '94.
20. FP-Tree: Han, J., Pei, J., and Yin, Y. 2000. Mining Frequent patterns without candidate generation.In SIGMOD '00