Improving the Predictive performance by integrating Decision tree based attribute selection in Clustering

Mrs.Anjani Yalamanchili[*], Mr.Jairam Kallepalli[1]

[*]*AssociateProfessor, VKR,VNB & AGK College of Engineering, Gudivada*

[1]*AssociateProfessor, VKR,VNB & AGK College of Engineering, Gudivada*

[1]*ramjai4@gmail.com*

ABSTRACT

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). The clustering problem has been addressed in many contexts and by researchers in many disciplines; this reflects its broad appeal and usefulness as one of the steps in exploratory data analysis. In this paper a new framework is proposed by integrating decision tree based attribute selection for data clustering. In this proposed system robust Modified Boosting algorithm is proposed to ID3,C45 based decision trees for clustering the data. Experimental results shows proposed algorithm improves results for data clustering results compare to existing clustering algorithms. In addition, this algorithm can improve the predictive performance especially for multi class datasets which can increase the accuracy.

Key Words *– Decision tree, Clusters, Patterns*

1.INTRODUCTION

Data mining applications, either in a design phase or as part of their on-line operations. Data analysis procedures can be dichotomized as either exploratory or confirmatory, based on the availability of appropriate models for the data source, but a key element in both types of procedures (whether for hypothesis formation or decision-making) is the grouping, or classification of measurements based on either (i) goodness-of-fit to a postulated model, or (ii) natural groupings (clustering) revealed through analysis. Cluster analysis is the organization of a collection of patterns (usually represented as a vector of measurements, or a point in a multidimensional space) into clusters based on similarity.

Intuitively, patterns within a valid cluster are more similar to each other than they are to a pattern belonging to a different cluster. An example of clustering is depicted in Figure 1. The input patterns are shown in Figure 1(a), and the desired clusters are shown in Figure 1(b). Here, points belonging to the same cluster are given the same label. The variety of techniques for representing data, measuring proximity (similarity) between data elements, and grouping data elements has produced a rich and often confusing assortment of clustering methods. It is important to understand the difference between clustering (unsupervised classification) and discriminate analysis (supervised classification). In supervised classification, we are provided with a collection of *labeled* (pre classified) patterns; the problem is to label a newly encountered, yet unlabeled, pattern. Typically, the given labeled (*training*) patterns are used to learn the descriptions of classes which in turn are used to label a new pattern. In the case of clustering, the problem is to group a given collection of unlabeled patterns into meaningful clusters. In a sense, labels are associated with clusters also, but these category labels are *data driven*; that is, they are obtained solely from the data. Clustering is useful in several exploratory pattern-analysis, grouping, decision- making, and machine-learning situations, including data mining, document retrieval, image segmentation, and pattern classification. However, in many such problems, there is little prior information (e.g., statistical models) available about the data, and the decision-maker must make as few assumptions about the data as possible. It is under these restrictions that clustering methodology is particularly appropriate for the exploration of interrelationships among the data points to make an assessment (perhaps preliminary) of their structure.

Clustering is an important method in data warehousing and data mining. It groups similar object together in a cluster (or clusters) and dissimilar object in other cluster (or clusters) or remove from the clustering process. That is, it is an unsupervised classification in data analysis that arises in many applications in different fields such as data mining[3], image processing, machine learning and bioinformatics. Since, it is an unsupervised learning

method, it does not need train datasets and pre-defined taxonomies. But there are some special requirements for search results clustering algorithms, two of which most important is, clustering performance and meaningful cluster description. Lots of clustering method is available, among those hierarchical clustering and Partition Clustering is the widely used clustering methods. A Partition-clustering algorithm in their outputs produce one clustering set that consists of disjoint clusters, i.e., the data description is flat. In other words, partitioned clustering is nothing but pre-defined number of partition range. Where the total number of partition (k) range should be less than number of object (n) in the dataset. Partition clustering always should satisfy the condition $k < n$.
A Hierarchical clustering is a nested of partitions technique depend on the business requirements. It produces not just one clustering set in their outputs but a hierarchy of clusters. This method work for both kind of approach either bottom up and top down approach. In this method all record object arranged with in a big cluster, then big cluster are continuously divided into small clusters.

2.RELATED WORK

A classification algorithm for data streams must meet several different requirements from the traditional setting (Bifet *et al*., 2009). The most significant are the following. First, process one example at a time, and inspect it at most once. The data examples flow in and out of a system one after another. Each example must be accepted in the order in which it arrives. Once inspected or ignored, the example is discarded with no way to retrieve it. Second, use a limited amount of memory. Memory will be easily exhausted without limiting its allocation since the amount of the data is potentially infinite. Third, work in a limited amount of time. Though most conventional algorithms are fast enough when classifying examples, the training processes are time consuming. For an algorithm to scale comfortably to any number of examples, its training complexity must be linear to the number of examples, such that online learning is possible. Fourth, be ready to perform classification at any time. This is the so-called any-time property, which indicates that the induction model is ready to be applied at any point between training examples[1].

Decision Tree Induction Algorithms on Data Streams

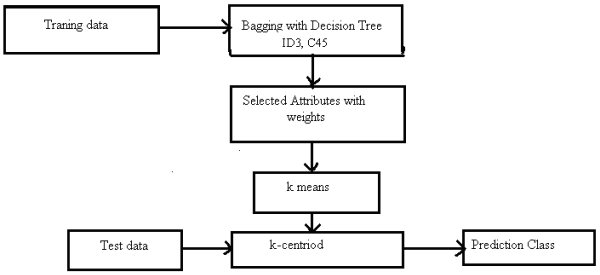
Decision tree is one of the most often used techniques in the data mining literature. Each node of a decision tree contains a test on an attribute. Each branch from a node corresponds to a possible outcome of the test and each leaf contains a class prediction. A decision tree is constructed by recursively
replacing leaves by test nodes, starting at the root. The attribute to test in a leaf is chosen by comparing all available attributes and choosing the best one[2]. according to some heuristic evaluation function. Classic decision tree learners like ID3, C4.5, and CART assume that all training examples can be stored simultaneously in memory, and thus are severely limited in the number of examples from which they can learn.

Predictive Clustering

In particular, the predictive modeling methods that partition the examples into subsets, e.g., decision trees and decision rules, can also be viewed as clustering methods .Namely, a decision tree can be regarded as a hierarchy of clusters, where each node is a cluster; such a tree is called a clustering tree. Likewise, a decision rule can represent a cluster of examples which it covers. The benefit of using these methods for clustering is that, in addition to the clusters themselves, we also get symbolic descriptions of the constructed clusters. Every cluster in a tree has a symbolic description in the form of a conjunction of conditions on the path from the root of the tree to the given node, and every cluster represented by a rule is described by the rule's condition. There is, however, a difference between 'tree' clusters and 'rule'[4] clusters. 'Tree' clusters are ordered in a hierarchy and do not overlap, while 'rule' clusters in general are not ordered in any way (they are flat) and can overlap (one example can belong to more than one cluster). We can say that clustering trees are a hierarchical clustering method, and clustering rules are a partitional (and possibly fuzzy) clustering method.
Brieman, Friedman, Olshen, and Stone developed the CART algorithm in 1984. It builds a binary tree. Observations are split at each node by a function on one attribute. The split is selected which divides the observations at a node into subgroups in which a single class most predominates. When no split can be found that increases the class specificity at a node the tree has reached a leaf node. When all observations are in leaf nodes

the tree has stopped growing. Each leaf can then be assigned a class and an error rate (not every observation in a leaf node is of the same class). Because the later splits have smaller and less representative samples to work with they may overfit the data. Therefore, the tree may be cut back to a size which allows effective generalization to new data. Branches of the tree that do not enhance predictive classification accuracy are eliminated in a process known as "pruning."

3.PROPOSED SYSTEM ARCHITECTURE



Dataset File Formats

In this project two types of file formats are used. They are

i) CSV
ii) ARFF

i. **CSV**: It stands for Comma Separated Value. This format is obtained using MS-Excel. KDD99 dataset is loaded into Excel and then it is saved with an extension of csv.
ii. **ARFF**: It stands for Attribute Relation File Format. An file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning software

ARFF files have two distinct sections. The first section is the Header information, which is followed the Data in The ARFF Header Section:

The ARFF Header section of the file contains the relation declaration and attribute declarations.

The @relation Declaration

The relation name is defined as the first line in the ARFF file. The format is:

```
@relation <relation-name>
```

where <relation-name> is a string. The string must be quoted if the name includes spaces.

The @attribute Declarations:

Attribute declarations take the form of an orderd sequence of @attribute statements. Each attribute in the data set has its own @attribute statement which uniquely defines the name of that attribute and it's data type. The order the attributes are declared indicates the column position in the data section of the file. For example, if an attribute is the third one declared then Weka expects that all that attributes values will be found in the third comma delimited column.

The format for the @attribute statement is:

@attribute <attribute-name> <datatype>

where the <attribute-name> must start with an alphabetic character. If spaces are to be included in the name then the entire name must be quoted.

The <datatype> can be any of the four types currently (version 3.2.1) supported by Weka:

- numeric
- <nominal-specification>
- string
- date [<date-format>]

where <nominal-specification> and <date-format> are defined below. The keywords numeric, string and date are case insensitive.

Nominal attributes

Nominal values are defined by providing an <nominal-specification> listing the possible values: {<nominal-name1>, <nominal-name2>, <nominal-name3>, ...}

String attributes

String attributes allow us to create attributes containing arbitrary textual values. String attributes are declared as follows:

@ATTRIBUTE name string

Decision tree is a tree structure, where internal nodes denote a test on an attribute, each branch represents the outcomes of the test and the leaf node represents the class labels. Decision tree induction is the learning of decision trees from class-labeled training tuples. Construction of decision trees is simple and fast, and does not need any domain knowledge and hence appropriate for exploratory knowledge discovery. In general, decision tree classifiers have good accuracy, but successful use of it depends on the data at hand. Decision trees are used for classification and classification rules are easily generated from them. An unknown tuple X can be classified, given its attribute values by testing the attribute values against the decision tree. The general decision tree algorithm takes the training data set, attribute list and attribute selection method as input. The algorithm creates a node, and then applies attribute selection method to determine the best splitting criteria and the created node is named by that attribute. Subset of training tuples is formed using the splitting attribute. The algorithm is called recursively for each subset, till the subset contains tuples of same class. When the subset contains tuples from the same class a leaf is attached with a label of the majority class in the training set from the root. ID3, C4.5, and CART adopt a greedy, non-backtracking approach in which decision trees are constructed in a top-down recursive divide-and-conquer.

Decision Tree Induction

ID3, C4.5, and CART adopt a greedy (i.e., nonbacktracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner. Most algorithms for decision tree induction also follow such a top-down approach, which starts with a training set of tuples and their associated class labels. The training set I recursively partitioned into smaller subsets as the tree is being built. The leaf nodes of the decision tree contain the class name whereas a non-leaf node is a decision node. The decision node is an attribute test with each branch (to another decision tree) being a

possible value of the attribute. ID3 uses information gain to help it decide which attribute goes into a decision node. ID3 is a non-incremental algorithm, meaning it derives its classes from a fixed set of training instances. An incremental algorithm revises the current concept definition, if necessary, with a new sample. The classes created by ID3 are inductive, that is, given a small set of training instances, the specific classes created by ID3 are expected to work for all future instances. The distribution of the unknowns must be the same as the test cases. Induction classes cannot be proven to work in every case since they may classify an infinite number of instances. Note that ID3 (or any inductive algorithm) may misclassify data.

1) ID3 ALGORITHM:

Algorithm: Generate decision tree. Generate a decision tree from the training tuples of data partition D.
Input: Data partition, D, which is a set of training tuples and their associated class labels; attribute list, the set of candidate attributes; Attribute selection method, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a splitting attribute and, possibly, either a split point or splitting subset.
Output: A decision tree.
Method:
(1) create a node N;
(2) if tuples in D are all of the same class, C then
(3) return N as a leaf node labeled with the class C;
(4) if attribute list is empty then
(5) return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply Attribute selection method(D, attribute list) to find the "best" splitting criterion;
(7) label node N with splitting criterion;
(8) if splitting attribute is discrete-valued and
multiway splits allowed then // not restricted to binary trees
(9) attribute list attribute list □ splitting attribute; // remove splitting attribute
(10) for each outcome j of splitting criterion
// partition the tuples and grow subtrees for each partition
(11) let Dj be the set of data tuples in D satisfying outcome j; // a partition
(12) if Dj is empty then
(13) attach a leaf labeled with the majority class in D to node N;
(14) else attach the node returned by Generate decision tree(Dj, attribute list) to node N; endfor
(15) return N;

C4.5 is based on the ID3 algorithm developed by Ross Quinlan [6], with additional features to address problems that ID3 was unable to deal. In practice, C4.5 uses one successful method for finding high accuracy hypotheses, based on pruning the rules issued from the tree constructed during the learning phase. However, the principal disadvantage of C4.5 rule sets is the amount of CPU time and memory they require. Given a set S of cases, J48 first grows an initial tree using the divide-and-conquer algorithm as follows:
i) If all the cases in S belong to the same class or S is small, the tree is leaf labelled with the most frequent class in S.
ii) Otherwise, choose a test based on a single attribute with two or more outcomes. Make this test as the root of the tree with one branch for each outcome of the test, partition S into corresponding subsets S1,S2,… according to the outcome for each case, and apply the same procedure recursively to each subset.

Robust Weighted Agglomerative Algorithm:

Based on these facts, we briefly present a simple and computationally efficient clustering algorithm for base models which is a combination and modification of two existing algorithms, namely *Leaders* and *Agglomerative* methods. Since this algorithm just uses a one-time scanning of the data samples inorder to accomplish the partitioning of the dataspace ,it is computationally very efficient .However because of the random and simple nature of the method, its results are usually considered very poor compared to the performance of other algorithms. Before proceeding into details of the proposed methods, we present our basemodel in Algorithms(1) and(2). The clustering approach for *LeadersforWeightedSamples* is a very straightforward center-based partitioning method :for any input sample xi, it finds the *first* leader (cluster center) which is close enough to the sample. Note that the algorithm even does not search for best matching leader, just the first encounter is selected(based on the leaders order in the set P).

Algorithm: Leaders for Weighted Samples

1. Input dataset and sample weights: X = $\{x_n\}$, W=$\{w_n\}$, n= 1,2,…,N
2. Initialize the leaders (clusters): P ={ }
3. Initialize the leader weights : C ={ }
4. Select the distance threshold: τ
5. **for** n = 1 to N **do**
6. Randomly select one sample, x_i, using the weights as probability distribution, and remove it from the list of samples to be chosen later
7. **if** P is empty **then**
8. Add x_i as the first leader: P = { x_i } and C = { w_i }
9. **else**
10. Find the first leader which $||x_i - \mathbf{p_j}|| \leq \tau$, $\mathbf{p_j} \in P$
11. **if** there was no p_j satisfying the distance criterion **then**
12. Add x_i as the next leader: P← P U $\{x_j\}$ and C← C U $\{w_i\}$
13. **else**
14. Update: $p_j \leftarrow \frac{c_j p_j + w_i x_i}{c_j + w_i}$ and $c_j \leftarrow c_j + w_i$
15. **end if**
16. **end if**
17. **end for**
18. Output the Final Partitions: (P, C)

Algorithm: Agglomerative Clustering of Leaders

1. Input the *Leaders* model (P, C)
2. Input the desired number of final clusters, K
3. **for** k = 1 to K_p - K **do**
4. Choose the closest leaders: (i, j) = $\arg_{i,j} \min ||p_i - p_j||$
5. Merge these leaders:
$$p_{new} = \frac{c_i p_i + c_j p_j}{c_i + c_i} \text{ and } c_{new} \leftarrow c_i + c_j$$
6. Remove ith and jth leaders from P and C, and add (P_{new}, C_{new}) to the list of leaders
7. **end for**
8. Output the Final Partitions (Leaders): P

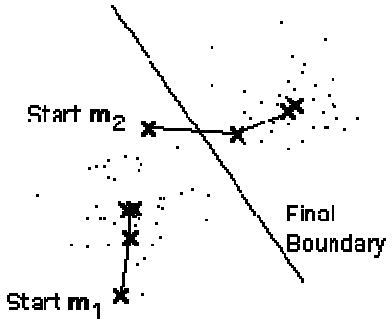
K-MEANS algorithm:

Suppose that we have n sample feature vectors $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ all from the same class, and we know that they fall into k compact clusters, k < n. Let $\mathbf{m}_i$ be the mean of the vectors in cluster i. If the clusters are well separated, we can

use a minimum-distance classifier to separate them. That is, we can say that **x** is in cluster i if || **x** - $\mathbf{m}_i$ || is the minimum of all the k distances. This suggests the following procedure for finding the k means:

- Make initial guesses for the means $\mathbf{m}_1$, $\mathbf{m}_2$, ..., $\mathbf{m}_k$
- Until there are no changes in any mean
  - Use the estimated means to classify the samples into clusters
  - For i from 1 to k
    - Replace $\mathbf{m}_i$ with the mean of all of the samples for cluster i
  - end_for
- end_until

Here is an example showing how the means $\mathbf{m}_1$ and $\mathbf{m}_2$ move into the centers of two clusters.



4.EXPERIMENTAL RESULTS

In the experiment Internet advertisement and Libras Movement are used as shown below.

Libras Movement:

Number of Instances: 360 (24 in each of fifteen classes)
Number of Attributes: 90 numeric (double) and 1 for the class (integer)
Attribute Information:

1. coordinate abcissa
2. coordinate ordinate
3. coordinate abcissa
4. coordinate ordinate

...

89. coordinate abcissa
90. coordinate ordinate
91. class:
    -- 1: curved swing
    -- 2: horizontal swing
    -- 3: vertical swing
    -- 4: anti-clockwise arc
    -- 5: clockwise arc
    -- 6: circle
    -- 7: horizontalstraight-line
    -- 8: vertical straight-line
    -- 9: horizontal zigzag
    -- 10: vertical zigzag
    -- 11: horizontal wavy
    -- 12: vertical wavy

-- 13: face-up curve
-- 14: face-down curve
-- 15: tremble

Internet Advertisement:

Number of Instances: 3279 (2821 nonads, 458 ads)
Number of Attributes: 1558 (3 continous; others binary;)
height: continuous. | possibly missing
width: continuous. | possibly missing
aratio: continuous. | possibly missing
local: 0,1.
| 457 features from url terms, each of the form "url*term1+term2...";
| for example:
url*images+buttons: 0,1.
| 495 features from origurl terms, in same form; for example:
origurl*labyrinth: 0,1.
| 472 features from ancurl terms, in same form; for example:
ancurl*search+direct: 0,1.
| 111 features from alt terms, in same form; for example:
alt*your: 0,1.
| 19 features from caption terms
caption*and: 0,1.

Results:

```
Registered_to_Vote = Not_Say
|  Education_Attainment = College: 1-3_yr (2.0)
|  Education_Attainment != College: 6-12_mo (6.0/1.0)
Registered_to_Vote != Not_Say
|  Major_Occupation = Other
|  |  Falsification_of_Information = Over_75
|  |  |  Actual_Time = Other: Under_6_mo (2.0/1.0)
|  |  |  Actual_Time != Other: 4-6_yr (2.0)
|  |  Falsification_of_Information != Over_75
|  |  |  Willingness_to_Pay_Fees = Payment_mechanism: 4-6_yr (2.0)
|  |  |  Willingness_to_Pay_Fees != Payment_mechanism
|  |  |  |  Country = Washington: 1-3_yr (2.0/1.0)
|  |  |  |  Country != Washington
|  |  |  |  |  Age = 49.0: Under_6_mo (2.0/1.0)
|  |  |  |  |  Age != 49.0
|  |  |  |  |  |  Age = 43.0: Under_6_mo (2.0/1.0)
|  |  |  |  |  |  Age != 43.0
|  |  |  |  |  |  |  Sexual_Preference = Bisexual: Under_6_mo (2.0/1.0)
|  |  |  |  |  |  |  Sexual_Preference != Bisexual
|  |  |  |  |  |  |  |  Actual_Time = Looking: 1-3_yr (2.0/1.0)
|  |  |  |  |  |  |  |  Actual_Time != Looking
|  |  |  |  |  |  |  |  |  Age = 63.0: 1-3_yr (2.0)
|  |  |  |  |  |  |  |  |  Age != 63.0
|  |  |  |  |  |  |  |  |  |  Education_Attainment = Grammar: 1-3_yr (2.0)
|  |  |  |  |  |  |  |  |  |  Education_Attainment != Grammar
|  |  |  |  |  |  |  |  |  |  |  Age = 23.0: 6-12_mo (3.0)
|  |  |  |  |  |  |  |  |  |  |  Age != 23.0
|  |  |  |  |  |  |  |  |  |  |  |  Country = Indiana
|  |  |  |  |  |  |  |  |  |  |  |  |  Actual_Time = Other: 6-12_mo (2.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  Actual_Time != Other: 1-3_yr (3.0)
|  |  |  |  |  |  |  |  |  |  |  |  Country != Indiana
|  |  |  |  |  |  |  |  |  |  |  |  |  Actual_Time = Artist/Musician: 1-3_yr (3.0/1.0)
|  |  |  |  |  |  |  |  |  |  |  |  |  Actual_Time != Artist/Musician
|  |  |  |  |  |  |  |  |  |  |  |  |  |  Race = Black: 1-3_yr (3.0/1.0)
```

```
Correctly Classified Instances         157               31.4629 %
Incorrectly Classified Instances       342               68.5371 %
=== Confusion Matrix ===

  a   b   c   d   e    <-- classified as
 67  28  44  29   4 |  a = 1-3_yr
 34  32   8  23   1 |  b = Under_6_mo
 48   7  27  12   8 |  c = 4-6_yr
 32  26   7  30   2 |  d = 6-12_mo
 11   2  15   1   1 |  e = Over_7_yr
```

5.CONCLUSION AND FUTURE WORK

The framework builds the patterns of the internet advertisement behavior over datasets labelled by the services. With the built patterns, the framework detects usage in the datasets using the classification algorithms. These results were useful in focusing research and highlighting the current capabilities and recent advances of existing algorithms. Experimental results shows that internet advertisement data set results poor accuracy when c45 algorithm is applied

Data mining algorithms require an offline training phase, but the testing phase requires much less time and future work could investigate how well it can be adapted to performing more accuracy by using combined decision tree as well as clustering algorithm.

6.ACKNOWLEDGEMENTS

Mrs.Y.Anjani has 10 years of teaching experience. She is presently working in VKR,VNB& AGK College of Engineering, Gudivada in the Department of Computer Science & Engineering. Her areas of interest are Data Mining, Mobile Computing. The author like to thank the Management of VKR,VNB& AGK College of Engineering for the support.

Mr.K.Jairam has 9 years of teaching experience. He is presently working in VKR,VNB& AGK College of Engineering, Gudivada in the Department of Computer Science & Engineering. Her areas of interest are Data Mining, Mobile Computing. The author like to thank the Management of VKR,VNB& AGK College of Engineering for the support.

7.REFERENCES

[1] ***Clustering feature decision trees for semi-supervised classification from high-speed data streams**** *Wen-hua XU†1, Zheng QIN†‡2, Yang CHANG2 (1Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China) (2School of Software, Tsinghua University, Beijing 100084, China)*

[2] ***Generalization and Decision Tree Induction: Efficient Classification in Data Mining*** *Micheline Kamber Lara Winstone Wan Gong Shan Cheng Jiawei Han Database Systems Research Laboratory School of Computing Science Simon Fraser University, B.C., Canada V5A 1S6 kamber, winstone, wgong, shanc, han @cs.sfu.ca*

[3] ***An Efficient Way for Clustering Using Alternative Decision Tree*** *Gothai, E. and P. Balasubramanie Department of Computer Science and Engineering, Kongu Engineering College, Perundurai, Erode, Tamilnadu, India*

[4] Learning Predictive Clustering Rules Bernard ˇZenko1, Saˇso Dˇzeroski1, and Jan Struyf2 [6] Yao H., Hamilton H.J., and Butz C.J, (2004) : A foundational approach to mining itemset utilities from databases, Proceedings of the 3rd SIAM International Conference on Data Mining, pp. 482-486.

[5]R. E. Schapire, 1999. A Brief Introduction to Boosting, in Proc. 16th Int. Joint Conference on Artificial Intelligence.

[6]R. E. Schapire, 2003. "The boosting approach to machine learning: An overview," in D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, B. Yu, editors, Nonlinear Estimation and Classification, Springer.

[7]Brown, R. G. and Hwang P. Y. C., 1983, Introduction to Random Signals and Applied Kalman Filtering, (New York:Wiley).

[8]De Loupy, C., Bellot, P., El-Bèze, M., Marteau, P.-F. (1999). Query Expansion and Automatic Classification. In Proceedings of Seventh Text REtrieval Conference TREC-7 (pp. 443–450), Gaithersburg, MD, USA: NIST special publication 500-242.

[9]Diday, E., Lemaire, J., Pouget, J., Testu, F. (1982). Eléments d'Analyse des Données. Dunod Informatique.

[10]Evans, D.A., Huettner, A., Tong, X., Jansen, P., Bennett, J. Effectiveness of Clustering in Ad-Hoc Retrieval. In Proceedings of the Seventh Text Retrieval Conference (TREC-7), NIST Special publication 500-242 (pp. 143–148).