Implementation of Methods of Stream Mining

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering/Information Technology

By

Abhimanyu Singh Gehlot (131319)
Rishav Kumar (131292)

Under the supervision of

Dr. Pardeep Kumar

to

Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled **"Implementation of Methods of Stream Mining"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2015 to May 2016 under the supervision of **Dr. Pardeep Kumar** Professor, Department of Computer Science And Engineering. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature) (Student Signature)
Abhimayu Singh Gehlot(131319) Rishav Kumar(131292)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Dr. Pardeep Kumar
Department of Computer Science And Engineering
Dated:15/12/2016

ACKNOWLEGEMENT

We are grateful and indebted to Dr. Pardeep Kumar, Department of Computer Science And Engineering for his help and advice in completion of this project report. We also express our deep sense of gratitude and appreciation to our guide for his constant supervision, inspiration and encouragement right from the beginning of this Seminar report. We also want to thank our parents and friends for their immense support and confidence upon us. We deem it a pleasant duty to place on record our sincere and heartfelt gratitude to our project guide for his long sightedness, wisdom and co-operation which helped us in tackling crucial aspects of the project in a very logical and practical way.

Abhimanyu Singh Gehlot
(131319)
Computer Science and Engineering

Rishav Kumar
(131292)

Table of Contents

List of Figures

List of Tables

List of Abbreviations

1. IP----------Internet Protocol
2. IEEE----------Institute of Electrical and Electronics Engineers
3. LLN----------Low power and lossy network
4. DIO----------DODAG Information Object
5. DAO----------Destination Advertisement Object
6. TCP----------Transmission Control Protocol

1. INTRODUCTION

1.1 Data Stream Mining

1.1.1 About Data Stream Mining

Data Stream Mining is the process of extracting knowledge structures from continuous, rapid data records. A data stream is an ordered sequence of instances that in many applications of data stream mining can be read only once or a small number of times using limited computing and storage capabilities. Examples of data streams include computer network traffic, phone conversations, ATM transactions, web searches, and sensor data. Data stream mining can be considered a subfield of data mining, machine learning, and knowledge discovery.

In many data stream mining applications, the goal is to predict the class or value of new instances in the data stream given some knowledge about the class membership or values of previous instances in the data stream. Machine learning techniques can be used to learn this prediction task from labeled examples in an automated fashion.

Often, concepts from the field of incremental learning, a generalization of Incremental heuristic search are applied to cope with structural changes, on-line learning and real-time demands. In many applications, especially operating within non-stationary environments, the distribution underlying the instances or the rules underlying their labeling may change over time, i.e. the goal of the prediction, the class to be predicted or the target value to be predicted, may change over time. This problem is referred to as concept drift.

1.2 Problem Statement

Every day, huge volumes of sensory, transactional, and web data are continuously generated as streams, which need to be analyzed online as they arrive. Streaming data can be considered as one of the main sources of what is called big data. While predictive modeling for data streams and big data have received a lot of attention over the last decade, many research approaches are typically designed for well-behaved controlled problem settings, overlooking important challenges imposed by real-world applications. Our goal is to identify gaps between current research and meaningful applications, highlight open problems, and define new application-relevant research directions for data stream mining.

1.3 Aims And Objectives

Our goal is to identify gaps between current research and meaningful applications, highlight open problems, and define new application-relevant research directions for data stream mining.

1.4 Methodology

In this project we simulate different methods of data stream mining using MOA a free open-source software specific for mining data streams. It has several machine learning algorithms (classification, regression, clustering, outlier detection and recommender systems).

1.5 Organisation

In Chapter 1 we have discussed about Data Stream Mining basics, the current growth in this field, the common challenges being faced in implementing the Stream mining algorithms.

In Chapter 2 we would be providing with the basic terminology about the different research paper read by us. We would be providing with facts and figures about different concepts we studied in those research papers.

In Chapter 3 we are going to provide a model of how the project is done on the basis of developments:-

• Analytical

• Experimental

• Statistical

In Chapter 4 we have given a proper analysis on Stream Mining algorithms on basis of which we will be implementing this project

2. LITERATURE SURVEY

2.1. Volatile Data

The volumes of automatically generated data are constantly increasing. According to the Digital Universe Study,over2.8ZB of data were created and processed in 2012, with a projected increase of 15 times by 2020. This growth in the production of digital data results from our surrounding environment being equipped with more and more sensors. People carrying smartphones produce data, database transactions are being counted and stored , streams of data are extracted from virtual environments in the form of logs or user generated content. A significant part of such data is volatile, which means it needs to be analyzed in real time as it arrives. Data stream mining is a research field that studies methods and algorithms for extracting knowledge from volatile streaming data.

Although data streams, online learning, big data, and adaptation to concept drift have become important research topics during
the last decade, truly autonomous, self-maintaining, adaptive data mining systems are rarely reported. This paper identifies real-world challenges for data stream research that are important but yet unsolved. Our objective is to present to the community a position paper that could inspire and guide future research in data streams. This article builds upon discussions at the International Workshop on Real-World Challenges for Data Stream Mining (Real Stream)1 in September 2013, in Prague, Czech Republic. Several related position papers are available. Dietterich[10]presents a discussion focused on predictive modeling techniques, that are applicable to streaming and non-streaming data. Fan and Bifet[12] concentrate on challenges presented by large volumes of data. Zlio baite et al. [48] focus on concept drift and adaptation of systems during online operation. Gaber et al. [13] discuss ubiquitous data mining with attention to collaborative data stream mining. In this paper, we focus on research challenges for streaming data inspired and required by real-world applications. In contrast to existing position papers, we raise issues connected not only with large volumes of data and concept drift, but also such practical problems as privacy constraints, availability of information, and dealing with legacy systems.

2.1.1 Categorisation

• Smoothness of concept transition:

Transitions between concepts can be sudden or gradual. The former is sometimes also denoted in literature as shift or abrupt drift.

• Singular or recurring contexts:

In the former case, a model becomes obsolete once and for all when its context is replaced by a novel context. In the latter case, a model's context might reoccur at a later moment in time ,for example due to a business cycle or seasonality, therefore, obsolete models might still regain value.

• Systematic or unsystematic:

In the former case, there are patterns in the way the distributions change that can be exploited to predict change and perform faster model adaptation. Examples are subpopulations that can be identified and show distinct, trackable evolutionary patterns. In the latter case ,no such patterns exist and drift occurs seemingly at random. An example for the latter is fickle concept drift.

• Real or virtual:

While the former requires model adaptation, the latter corresponds to observing outliers or noise, which should not be incorporated into a model. Stream mining approaches in general address the challenges posed by volume, velocity and volatility of data. However, in real-world applications these three challenges often coincide with other, to date insufficiently considered ones.

The next sections discuss eight identified challenges for data stream mining, providing illustrations with real world application examples, and formulating suggestions for forthcoming research.

2.1.2 Privacy Control

Data streams present new challenges and opportunities with respect to protecting privacy and confidentiality in data mining. Privacy preserving data mining has been studied for over a decade (see. e.g. [3]). The main objective is to develop such data mining techniques that would not uncover information or patterns which compromise confidentiality and privacy obligations. Modeling can be done on original or anonymized data, but when the model is released, it should not contain information that may violate privacy or confidentiality. This is typically achieved by controlled distortion of sensitive data by modifying the values or adding noise. Ensuring privacy and confidentiality is important for gaining trust of the users and the society in autonomous, stream data mining systems. While in offline data mining a human analyst working with the data can do a sanity check before releasing the model, in data stream mining privacy preservation needs to be done online. Several existing works relate to privacy preservation in publishing streaming data (e.g. [46]), but no systematic research in relation to broader data stream challenges exists. We identify two main challenges for privacy preservation in mining data streams. The first challenge is incompleteness of information. Data arrives in portions and the model is updated online. Therefore, the model is never final and it is difficult to judge privacy preservation before seeing all the data. For example, suppose GPS traces of individuals are being collected for modeling traffic situation. Suppose person A at current time travels from the campus to the airport. The privacy of a person will be compromised, if there are no similar trips by other persons in the very near future. However, near future trips are unknown at the current time, when the model needs to be updated. On the other hand, data stream mining algorithms may have some inherent privacy preservation properties due to the fact that they do not need to see all the modeling data at once, and can be incrementally updated with portions of data. Investigating privacy preservation properties of existing data stream algorithms makes another interesting direction for future research.

2.2 Timing and Availability of Information

Most algorithms developed for evolving data streams make simplifying assumptions on the timing and availability of information. In particular, they assume that information is complete, immediately available, and received passively and for free. These assumptions often do not hold in real-world applications, e.g., patient monitoring ,robot vision ,or marketing[43]. This section is dedicated to the discussion of these assumptions and the challenges resulting from their absence. For some of these challenges, corresponding situations in offline, static data mining have already been addressed in literature. We will briefly point out where a mapping of such known solutions to the online ,evolving stream setting is easily feasible, for example by applying windowing techniques. However, we will focus on problems for which no such simple mapping exists and which are therefore open challenges instream mining.

2.2.1 Dealing with Incomplete Information

Completeness of information assumes that the true values of all variables, that is of features and of the target, are revealed eventually to the mining algorithm. The problem of missing values, which corresponds to incompleteness of features, has been discussed extensively for the offline, static settings. A recent survey is given in [45]. However, only few works address data streams ,and in particular evolving data streams. Thus several open challenges remain, some are pointed out in the review by [29]: how to address the problem that the frequency in which missing values occur is unpredictable, but largely affects the quality of imputations? How to (automatically) select the best imputation technique? How to proceed in the trade-off between speed and statistical accuracy? Another problem is that of missing values of the target variable. It has been studied extensively in the static setting as semi-supervised learning (SSL, see [11]). A requirement for applying SSL techniques to streams is the availability of at least some labeled data from the most recent distribution. While first attempts to this problem have been made, e.g. the online manifold regularization approach in [19] and the ensembles-based approach suggested by [11],improvements in speed and the provision of performance guarantees remain open challenges. A special case of incomplete information is “censored data” in Event History Analysis(EHA),which is described in section 5.2. A related problem discussed below is active learning (AL, see [38]).

2.2.2 Handling Delayed Information

Latency means information becomes available with significant delay. For example, in the case of so-called verification latency, the value of the preceding instance's target variable is not available before the subsequent instance has to be predicted. On evolving data streams, this is more than a mere problem of streaming data integration between feature and target streams, as due to concept drift patterns show temporal locality [2]. It means that feedback on the current prediction is not available to improve the subsequent predictions, but only eventually will become available for much later predictions. Thus, there is no recent sample of labeled data at all that would correspond to he most-recent unlabeled data, and semi supervised learning approaches are not directly applicable. A related problem in static, offline data mining is that addressed by unsupervised rans ductive transfer learning(or un supervised domain adaptation): given labeled data from a source domain, a predictive model is sought for a related target domain in which no labeled data is available. In principle, ideas from transfer learning could be used to address latency in evolving data streams, for example by employing the minac hunk-based approach, as suggested in [43]. However, adapting them for use in evolving data streams has not been tried yet and constitutes a non-trivial, open task, as adaptation in streams must be fast and fully automated and thus cannot rely on iterated careful tuning by human experts. Furthermore, consecutive chunks constitute several domains, thus the transitions between several subsequent chunks might provide exploitable patterns of systematic drift. This idea has been introduced in [27], and a few so-called drift-mining algorithms that identify and exploit such patterns have been proposed since then. However, the existing approaches cover only a very limited set of possible drift patterns and scenarios.

2.3 Entity Stream Mining

Let T be a stream of entities, e.g. customers of a company or patients of a hospital. We observe entities over time, e.g. on a company's website or at a hospital admission vicinity: an entity appears and re-appears at discrete time points, new entities show up. At a time point t, an entity e $\in$ T is linked with different pieces of information- the purchases and ratings performed by a customer, the anamnesis, the medical tests and the diagnosis recorded for the patient. Each of these information pieces ij(t) is a structured record or an unstructured text from a stream Tj, linked toe via the foreign key relation.

Thus, the entities in T are in 1-to-1 or 1-to-n relation withentitiesfromfurtherstreamsT1,...,Tm (stream of purchases, stream of ratings, stream of complaints etc). The schema describing the streams T,T1,...,Tm can be perceived as a conventional relational schema, except that it describes streams instead of static sets. In this relational setting, the entity stream mining task corresponds to learning a model ζT over T, thereby incorporating information from the adjoint streams T1,...,Tm that ”feed” the entities in T. Albeit the members of each stream are entities, we use the term ”entity” only for stream T –the target of learning, while we denote the entities in the other streams as ”instances”. In the unsupervised setting ,entity stream clustering encompasses learning and adapting clusters over T, taking account the other streams that arrive at different speeds. In the supervised setting, entity stream classification involves learning and adapting a classifier ,not withstanding the fact that an entity’s label may change from one time point to the next, as new instances referencing it arrive.

2.4 Evaluating Data Stream mining Algorithms

All of the aforementioned challenges are milestones on the road to better algorithms for real-world data stream mining systems. To verify if these challenges are met, practitioners need tools capable of evaluating newly proposed solutions. Although in the field of static classification such tools exist, they are insufficient in data stream environments due to such problems as: concept drift, limited processing time, verification latency, multiple stream structures, evolving class skew, censored data, and changing misclassification costs. In fact, the myriad of additional complexities posed by data streams makes algorithm evaluation a highly multi-criterial task, in which optimal trade-offs may change over time. Recent developments in applied machine learning [6] emphasize the importance of understanding the data one is working with and using evaluation metrics which reflect its difficulties. As mentioned before, data streams set new requirements compared to traditional data mining and researchers are beginning to acknowledge the shortcomings of existing evaluation metrics. For example, Gama et al. [16] proposed a way of calculating classification accuracy using only the most recent stream examples, therefore allowing for time-oriented evaluation and aiding concept drift detection. Methods which test the classifier’s robustness to drifts and noise on a practical, experimental level are also starting to arise [34; 47]. However, all these evaluation techniques focus on single criteria such as prediction accuracy or robustness to drifts, even though data streams make evaluation a constant trade-off between several criteria [7]. Moreover, in data stream environments there is a need for more advanced tools for

visualizing changes in algorithm predictions with time. The problem of creating complex evaluation methods for stream mining algorithms lies mainly in the size and evolving nature of data streams. It is much more difficult to estimate and visualize, for example, prediction accuracy if evaluation must be done online, using limited resources, and the classification task changes with time. In fact, the algorithm's ability to adapt is another aspect which needs to be evaluated, although information needed to perform such evaluation is not always available. Concept drifts are known in advance mainly when using synthetic or benchmark data, while in more practical scenarios occurrences and types of concepts are not directly known and only the label of each arriving instance is known. Moreover ,in many cases the task is more complicated ,as labeling information is not instantly available. Other difficulties in evaluation include processing complex relational streams and coping with class imbalance when class distributions evolve with time. Finally ,not only dowe need measures for evaluating single aspects of stream mining algorithms, but also ways of combining several of these aspects into global evaluation models, which would take into account expert knowledge and user preferences. Clearly, evaluation of data stream algorithms is a fertile ground for novel theoretical and algorithmic solutions. In terms of prediction measures, data stream mining still requires evaluation tools that would be immune to class imbalance and robust to noise. In our opinion, solutions to this problem should involve not only metrics based on relative performance to baseline (chance) classifiers, but also graphical measures similar to PR-curves or cost curves. Furthermore, there is a need for integrating information about concept drifts in the evaluation process. As mentioned earlier ,possible ways of considering concept drifts will depend on the information that is available. If true concepts are known, algorithms could be evaluated based on: how often they detect drift, how early they detect it, how they react to it, and how quickly they recover from it. Moreover, in this scenario, evaluation of an algorithm should be dependent on whether it takes place during drift or during times of concept stability. A possible way of tackling this problem would be the proposal of graphical methods, similar to ROC analysis, which would work online and visualize concept drift measures alongside prediction measures. Additionally, these graphical measures could take into account the state of the stream, for example, its speed, number of missing values, or class distribution. Similar methods could be proposed for scenarios where concepts are not known in advance, however, in these cases measures should be based on drift detectors or label-independent stream statistics. Above all, due to the number of aspects which need to be measured, we believe that the evaluation of data stream algorithms requires a multi-criterial view. This could be done by using inspirations from multiple criteria decision

analysis ,where trade-offs between criteria are achieved using user-feedback. In particular, a user could showcase his/her criteria preferences (for example, between memory consumption, accuracy, reactivity, self-tuning, and adaptability) by deciding between alternative algorithms for a given data stream. It is worth noticing that such a multi-criterial view on evaluation is difficult to encapsulate in a single number, as it is usually done in traditional offline learning. This might suggest that researchers in this area should turn towards semi-qualitative and semi-quantitative evaluation, for which systematic methodologies should be developed. Finally, a separate research direction involves rethinking the way we test data stream mining algorithms. The traditional train, cross validate ,test work flow in classification is not applicable for sequential data, which makes, for instance, parameter tuning much more difficult. Similarly ground truth verification in unsupervised learning is practically impossible in data stream environments. With these problems in mind, it is worth stating that there is still a shortage of real and synthetic benchmark datasets. Such a situation might be a result of non-uniform standards for testing algorithms on streaming data. As community, we should decide on such matters as: What characteristics should benchmark datasets have? Should they have prediction tasks attached? Should we move towards online evaluation tools rather than datasets? These questions should be answered in order to solve evaluation issues in controlled environments before we create measures for real-world scenarios.

3. SYSTEM DEVELOPMENT

There exists a plethora of work in the area of distributed data stream mining. The existing literature provides an excellent starting point for our main topic of discussion in this chapter. Not only have the distributed data mining and databases community contributed to the literature, a bulk of the work also comes from the wireless and sensor networks community. In this section we discuss some of the related papers with pointers for further reading. Computation of complex functions over the union of multiple of streams has been studied widely in the stream mining literature .

Our objective scope is to provide a working mechanism, which will take as an argument a phrase or a sub-sentence and it will return a sentiment score to this particular part of speech. A prerequisite for this to happen is the development of another mechanism that will take a piece of text (a tweet for example), and will break it into as many sub sentences as the different ontologies that are contained in it. This is something that is already the subject of research of another dissertation that is conducted at our university alongside this current dissertation.

3.1 Algorithms

3.1.1 C 4.5 Data Stream Mining Algorithm

C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan.C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier.

It became quite popular after ranking #1 in the *Top 10 Algorithms in Data Mining* pre-eminent paper published by Springer LNCS in 2008.

3.1.2 The Algorithm

C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. The training data is a set of already classified samples. Each sample consists of a p-dimensional vector, where the represent attribute values or features of the sample, as well as the class in which falls.

At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists.

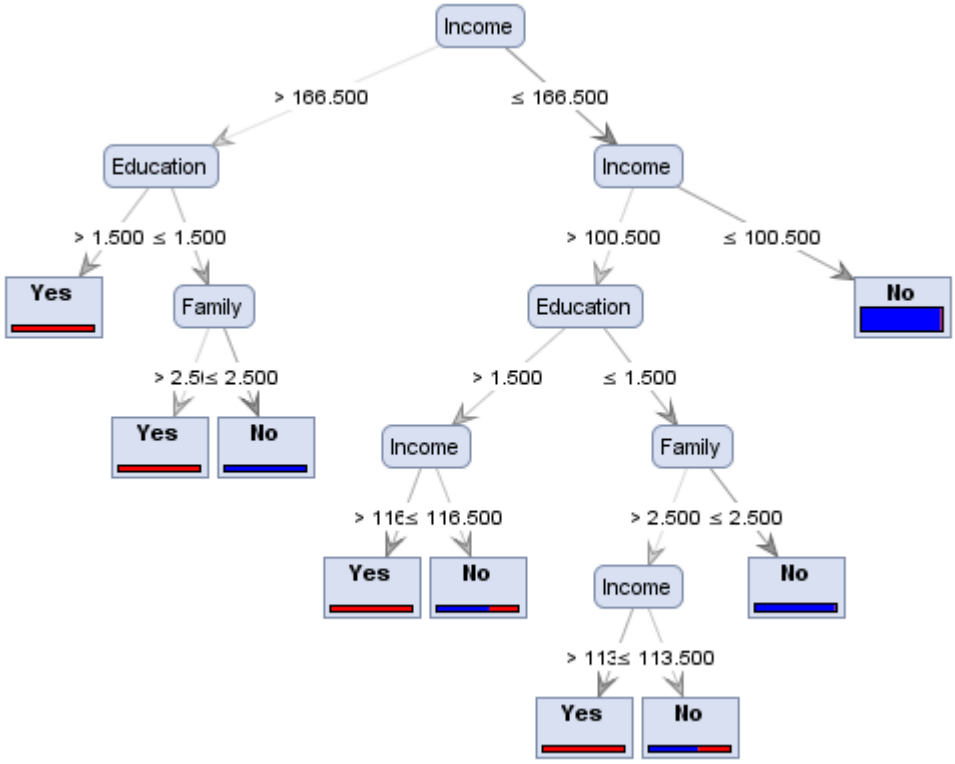
This algorithm has a few base cases.

- All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
- None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
- Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

3.1.3 Pseudocode

In pseudocode, the general algorithm for building decision trees is:

1. Check for the above base cases.
2. For each attribute *a*, find the normalized information gain ratio 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*.



C 4.5 Algorithm Decision Tree example

3.1.4 Improvements in C5.0 algorithm

Quinlan went on to create C5.0 and See5 (C5.0 for Unix/Linux, See5 for Windows) which he markets commercially. C5.0 offers a number of improvements on C4.5. Some of these are:

- Speed - C5.0 is significantly faster than C4.5 (several orders of magnitude)
- Memory usage - C5.0 is more memory efficient than C4.5
- Smaller decision trees - C5.0 gets similar results to C4.5 with considerably smaller decision trees.
- Support for boosting - Boosting improves the trees and gives them more accuracy.
- Weighting - C5.0 allows you to weight different cases and misclassification types.
- Winnowing - a C5.0 option automatically winnows the attributes to remove those that may be unhelpful.

3.2 K-means Data mining clustering Algorithm

K-Means is a simple learning algorithm for clustering analysis. The goal of K-Means algorithm is to find the best division of *n* entities in *k* groups, so that the total distance between the group's members and its corresponding centroid, representative of the group, is minimized. Formally, the goal is to partition the *n* entities into *k* sets S_i, *i=1, 2, ..., k* in order to minimize the within-cluster sum of squares (WCSS), defined as:

where term provides the distance between an entity point and the cluster's centroid.

The most common algorithm, described below, uses an iterative refinement approach, following these steps:

1. Define the initial groups' centroids. This step can be done using different strategies. A very common one is to assign random values for the centroids of all groups. Another approach is to use the values of *K* different entities as being the centroids.

2. Assign each entity to the cluster that has the closest centroid. In order to find the cluster with the most similar centroid, the algorithm must calculate the distance between all the entities and each centroid.

3. Recalculate the values of the centroids. The values of the centroid's fields are updated, taken as the average of the values of the entities' attributes that are part of the cluster.

4. Repeat steps 2 and 3 iteratively until entities can no longer change groups.

The K-Means is a greedy, computationally efficient technique, being the most popular representative-based clustering algorithm.

3.2.1 Pseudocode

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 ;
```

Note that the last line of the pseudocode should be either:

while c h a n g e d = t r u e and i t e r ≤ M a x I t e r s

; or

until c h a n g e d = f a l s e or i t e r > M a x I t e r s.

3.2.2 Why Use k means algorithm

The key selling point of k-means is its simplicity. Its simplicity means it's generally faster and more efficient than other algorithms, especially over large datasets.

It gets better:

k-means can be used to pre-cluster a massive dataset followed by a more expensive cluster analysis on the sub-clusters. k-means can also be used to rapidly "play" with k and explore whether there are overlooked patterns or relationships in the dataset.

It's not all smooth sailing:

Two key weaknesses of k-means are its sensitivity to outliers, and its sensitivity to the initial choice of centroids. One final thing to keep in mind is k-means is designed to operate on continuous data — you'll need to do some tricks to get it to work on discrete data.

3.2.3 K-means example illustration

As a simple illustration of a k-means algorithm, consider the following data set consisting of the scores of two variables on each of seven individuals:

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

This data set is to be grouped into two clusters. As a first step in finding a sensible initial partition, let the A & B values of the two individuals furthest apart (using the Euclidean distance measure), define the initial cluster means, giving:

	Individual	Mean Vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

The remaining individuals are now examined in sequence and allocated to the cluster to which they are closest, in terms of Euclidean distance to the cluster mean. The mean vector is recalculated each time a new member is added. This leads to the following series of steps:

	Cluster 1		Cluster 2	
Step	Individual	Mean Vector (centroid)	Individual	Mean Vector (centroid)
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

Now the initial partition has changed, and the two clusters at this stage having the following characteristics:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2, 3	(1.8, 2.3)
Cluster 2	4, 5, 6, 7	(4.1, 5.4)

But we cannot yet be sure that each individual has been assigned to the right cluster. So, we compare each individual's distance to its own cluster mean and to that of the opposite cluster. And we find:

Individual	Distance to mean (centroid) of Cluster 1	Distance to mean (centroid) of Cluster 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

Only individual 3 is nearer to the mean of the opposite cluster (Cluster 2) than its own (Cluster 1). In other words, each individual's distance to its own cluster mean should be smaller that the distance to the other cluster's mean (which is not the case with individual 3). Thus, individual 3 is relocated to Cluster 2 resulting in the new partition:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

The iterative relocation would now continue from this new partition until no more relocations occur. However, in this example each individual is now nearer its own cluster mean than that of the other cluster and the iteration stops, choosing the latest partitioning as the final cluster solution.

Also, it is possible that the k-means algorithm won't find a final solution. In this case it would be a good idea to consider stopping the algorithm after a pre-chosen maximum of iterations.

4. PERFORMANCE ANALYSIS

K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function know as squared error function.

4.1 Advantages (k-means algorithm)

1) Fast, robust and easier to understand.

2) Relatively efficient: O(tknd), where n is # objects, k is # clusters, d is # dimension of each object, and t is # iterations. Normally, k, t, d << n.

3) Gives best result when data set are distinct or well separated from each other.

4.1.2 Disadvantages (k-means algorithm)

1) The learning algorithm requires apriori specification of the number of cluster centers.

2) The use of Exclusive Assignment - If there are two highly overlapping data then k-means will not be able to resolve that there are two clusters.

3) The learning algorithm is not invariant to non-linear transformations i.e. with different representation of data we get

different results (data represented in form of cartesian co-ordinates and polar co-ordinates will give different results).

4) Euclidean distance measures can unequally weight underlying factors.

5) The learning algorithm provides the local optima of the squared error function.

6) Randomly choosing of the cluster center cannot lead us to the fruitful result.

7) Applicable only when mean is defined i.e. fails for categorical data.

8) Unable to handle noisy data and outliers.

9) Algorithm fails for non-linear data set.

4.2 Advantages (C4.5 algorithm)

The advantages of the C4.5 are:

- Builds models that can be easily interpreted
- Easy to implement
- Can use both categorical and continuous values
- Deals with noise

4.2.1 Disadvantages (C4.5 Algorithm)

The disadvantages are:

- Small variation in data can lead to different decision trees (especially when the variables are close to each other in value)

- Does not work very well on a small training set

5. CONCLUSIONS AND FUTURE WORK

C4.5 is used in classification problems and it is the most used algorithm for builing DT.

It is suitable for real world problems as it deals with numeric attributes and missing values. The algorithm can be used for building smaller or larger, more accurate decision trees and the algorithm is quite time efficient.

Compared to ID3, C4.5 performs by default a tree pruning process, which leads to smaller trees, more simple rules and more intutive interpretations.

K-means algorithm is also used in classification problems as it is Fast, robust and easier to understand.

Relatively efficient: O(tknd), where n is # objects, k is # clusters, d is # dimension of each object, and t is # iterations. Normally, k, t, d << n.

Gives best result when data set are distinct or well separated from each other.

6. REFRENCES.

B.V., B. M. (2013). Open Dover. (Byelex Multimedia Products B.V.) Retrieved Oktober 2013, from http://www.opendover.nl/

Antoniou, G. C. (2012). Creation of a Twitter web-hint system that propose tweets based on the user preferences. Thessaloniki, Greece: Dept. of Electrical and Computer Engineering, Aristotle University of Thessaloniki .

Barbosa, L., & Feng, J. (2010, August). Robust Sentiment Detection on Twitter from Biased and Noisy Data. Coling 2010: Poster Volume, pp. 36-44.

Brody, S., & Elhadad, N. (2010). An Unsupervised Aspect-Sentiment Model for Online Reviews. Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL (pp. 804-812). Los Angeles: Association for Computational Linguistics.

Cavnar, W. B., & John, T. M. (1994). N-Gram-Based Text Categorization. Environmental Research Institute of Michigan.

Cerini, S., Compagnoni, V., Demontis, A., Formentelli, M., & Gandini, G. (2007). MicroWNOp: A gold standard for the evaluation of automatically compiled lexical resources for opinion mining. Andrea Sans , ed., Language Resources and Linguistic Theory: Typology, Second Language Acquisition, English Linguistics.

Cianciullo, J. (n.d.). SocialMention. Retrieved Oktober 2013, from http://www.socialmention.com/

Bakliwal, A., Foster, J., van der Puil, J., O'Brien, R., Tounsi, L., & Hughes, M. (2013). Sentiment Analysis of Political Tweets: Towards an Accurate Classifier. Proceedings of the Workshop on Language in Social Media (LASM 2013), (pp. 49-58). Atlanta, Georgia.

Davidov, D., Tsur, O., & Rappoport, A. (2010, August). Enhanced Sentiment Learning Using Twitter Hashtags and Smileys. Coling 2010: Poster Volume, pp. 241-249.

Dergiades, T. (2012). Do investors' sentiment dynamics affect stock returns? Evidence from the US economy. Economics Letters, 116(3), pp. 404-407.

Esuli, A., & Sebastiani, F. (2006). SENTIWORDNET: A Publicly Available Lexical Resource.

Ganter, B. B., & Wille, R. (1999). Formal concept analysis, mathematical foundation. Berlin: Springer Verlag.

Gruber, T. R. (1993, June). A translation approach to portable ontology specifications. Knowledge Acquisition 5 (2), pp. 199-220.

Hall, M., & Frank, E. (2001). A Simple Approach to Ordinal Classification. 12th European Conference on Machine Learning, (pp. 145-156).

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (pp. 168–177).

John G. Cleary, L. E. (1995). K*: An Instance-based Learner Using an Entropic Distance Measure. 12th International Conference on Machine Learning, (pp. 108-114).

Kontopoulos, E., Berberidis, C., Dergiades, T., & Bassiliades, N. (2013). Ontologybased sentiment analysis of twitter posts. Elsevier(Expert Systems with Applications 40 (2013)), pp. 4065–4074.

Kouloumpis, E., Wilson, T., & Moore, J. (2011). Twitter Sentiment Analysis: The Good the Bad and the OMG! Association for the Advancement of Artificial.

Kumar, S., Morstatter, F., & Liu, H. (2013). Twitter Data Analytics. Springer.

Liu, B. (2012). Sentiment Analysis and Opinion Mining. AAAI-2011, EACL-2012, and Sentiment Analysis Symposium.

Liu, Y., Huang, X., An, A., & Yu, X. (2007). ARSA: A Sentiment-Aware Model for Predicting Sales (Vol. SIGIR'07). Amsterdam.

Miller, G. A. (1995). WordNet: A lexical database for English. Communications of the ACM 38(11), pp. 39–41.

Naes, T., Isaksson, T., Fearn, T., & Davies, T. (2002). A User Friendly Guide to Multivariate Calibration and Classification. NIR Publications.

Nebhi, K. (2012). Ontology-Based Information Extraction from Twitter. Proceedings of the Workshop on Information Extraction and Entity Analytics on Social Media Data (pp. 17-22). Mumbai: COLING.

Pang, B., & Lee, L. (2004). A Sentimental Education: Sentiment Analysis Using Subjectivity. ACL '04, (pp. 271–278).

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning. Proceedings of EMNLP, (pp. pp. 79–86).

Park, S., Ko, M., Kim, J., Liu, Y., & Song, J. (2011). The Politics of Comments:

Predicting Political Orientation. ACM 978-1-4503-0556-3/11/03. Hangzhou. Stanford University. (2013). The Stanford Natural Language Processing Group. Retrieved from http://nlp.stanford.edu/software/corenlp.shtml

Stone, P. J., Dunphry, D., Smith, M., & Ogilvie, D. (1966). The General Inquirer: A Computer Approach to Content Analysis. Cambridge: MIT Press. Tsoumakas, G., & Vlahavas, I. (n.d.). Effective Stacking of Distributed Classifiers.

Thessaloniki: Dept. of Informatics, Aristotle University of Thessaloniki.

Tweet Sentiment Visualization. (n.d.). Retrieved Oktober 2013, from Sentiment Viz: http://www.csc.ncsu.edu/faculty/healey/tweet_viz/tweet_app/ Waikato, D. o. (2013, Oktober). Weka 3: Data Mining Software in Java. Retrieved from http://www.cs.waikato.ac.nz/~ml/weka/

Witten, I. H., Frank, E., & Hall, M. A. (2011). Data Mining, Practical Machine Learning Tools and Techniques (3rd ed.). U.S.A.: Morgan Kaufmann Publishers.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., . . . Steinberg, D. (2007). Top 10 algorithms in data mining. Springer-Verlag.

Zhang, L., Ghosh, R., Dekhil, M., Hsu, M., & Liu, B. (2011). Combining Lexiconbased and Learning-based Methods for Twitter. HP Laboratories(HPL-2011-89).

Zhang, L., Xu, W., & Li, S. (2012). Aspect identification and sentiment analysis based on NLP. Network Infrastructure and Digital Content (IC-NIDC), 2012 3rd IEEE International Conference on, (pp. 660 - 664). Beijing.

7. APPENDIX

7.1 Code Snippets:-

```
#include <iostream>
```

```
#include <conio.h>
#include <stdlib.h>

using namespace std;

int main()
{
 int numbers, k, kvals[25], prevKvals[25], steps = 1, addition[25][100], count = 0, groups[25]
[100], min, groupnum, value, sum, ok = 1, nums[100];
 cout << "How many numbers you want to enter: ";
 cin >> numbers;

 cout << "Enter value of k: ";
 cin >> k;

 //get numbers
 for(int i = 0; i < numbers; i++)
 {
  cout << "Enter Number " << i+1 << ": ";
  cin >> nums[i];
 }

 // set values of C's
 for(int i = 0; i < 3; i++)
 {
  kvals[i] = nums[i];
 }
 //show values of user
 cout << "You have entered: ";
 for(int i = 0; i < numbers; i++)
 {
  cout << nums[i] << ", ";
 }
```

```
//while(steps < 10)
while(ok == 1)
{
 cout << endl << "Itration Number: " << steps;
 //make calculations (C - bla bla bla)
 for(int i = 0; i < k; i++)
 {
          for(int j = 0; j < numbers; j++)
          {
  addition[i][j] = abs(kvals[i] - nums[j]);
 }
 }

 //make groups of number(C)
 for(int i = 0; i < numbers; i++)
 {
  min = 100000;
          for(int j = 0; j < k; j++)
          {
  if(addition[j][i] < min)
  {
     min = addition[j][i];
     value = nums[i];
     groupnum = j;
  }
 }
 groups[groupnum][i] = value;
}

 //show results of calculations (C - bla bla bla)
   cout << endl << "Calculations" << endl;
 for(int i = 0; i < numbers; i++)
 {
          for(int j = 0; j < k; j++)
```

```
        {
 cout << addition[j][i] << "\t";
 }
 cout << endl;
}
// show groups and get new C's
cout << endl << "Gruops" << endl;
   for(int i = 0; i < k; i++)
{
 sum = 0;
 count = 0;
 cout << "Group " << i+1 << ": ";
        for(int j = 0; j < numbers; j++)
        {
  if(groups[i][j] != NULL)
  {
   cout << groups[i][j] << "\t";
   sum += groups[i][j];
   count++;
  }
 }
 prevKvals[i] = kvals[i];
 kvals[i] = sum/count;
 cout << "\t=\t" << kvals[i] << endl;
}

//make empty array of groups
for(int i = 0; i < 25; i++)
{
 for(int j = 0; j < 100; j++)
 {
  groups[i][j] = NULL;
 }
}
```

```
 //check condition of termination
 ok = 0;
 for(int i = 0; i < k; i++)
 {
  if(prevKvals[i] != kvals[i])
  {
      ok = 1;
  }
 }

  if(ok != 1)
  {
       getch();
}

    steps++;
} // end while loop

getch();
return 0;
}
```