Incremental Clustering for the Classification of Concept-Drifting Data Streams

Ioannis Katakis, Grigorios Tsoumakas, Ioannis Vlahavas

Department of Informatics, Aristotle University, Thessaloniki, 54124, Greece.
{katak, greg, vlahavas}@csd.auth.gr

Abstract. Concept drift is a common phenomenon in streaming data environments and constitutes an interesting challenge for researchers in the machine learning and data mining community. This paper proposes a probabilistic representation model for data stream classification and investigates the use of incremental clustering algorithms in order to identify and adapt to concept drift. An experimental study is performed using three real-world datasets from the text domain, a basic implementation of the proposed framework and three baseline methods for dealing with drifting concepts. Results are promising and encourage further investigation.

1. Introduction

Recent advances in sensor, storage, processing and communication technologies have enabled the automated recording of data, leading to fast and continuous flows of information, referred to as data streams. The dynamic nature of data streams requires continuous or at least periodic updates of the current knowledge in order to ensure that it always includes the information content of the latest batch of data. This is important in applications where the concept of a target class and/or the data distribution changes over time. This phenomenon is known as concept drift.

Although traditional vector data representation is effective and widely used for stationary data classification tasks, it is not adequate for classification problems involving concept drift. The main reason is that in concept drifting scenarios, geometrically close items in the conventional vector space might belong to different classes. This is because of a concept change (drift) that occured at some time point.

After noticing this problem, we propose a new probabilistic representation for data streams suitable for problems with concept drift. More specifically, we map batches of data into what we name “Conceptual Vectors”. These vectors contain conceptual information for every batch of data. In this new probabilistic space, vectors that are geometrically close do always belong in the same conceptual theme.

Using the proposed representation, we apply incremental clustering in the stream of Conceptual Vectors. This way, we organize and summarize batches of instances into concepts. The final objective is to train an incremental classifier on every cluster / concept. When a new batch arrives we identify the concept (cluster) that this batch belongs to and apply the corresponding classifier for prediction. Experiments with a

basic incremental clusterer and the Naïve Bayes classifier produce interesting results and encourage further research.

The rest of the paper is organized as follows. In section 2 we present background information on mining data stream whereas in section 3 we summarize related work on concept drift detection and adaptation. Section 4 discusses the deficiency of the conventional data representation and introduces the proposed representation model. Section 5 suggests a framework for dealing with concept drift with the combination of incremental clustering and classification. In section 6, we develop a simple implementation of the proposed framework and evaluate it in real world datasets from the text domain. Section 7 concludes the paper by presenting our plans for future work.

2. Mining Data Streams

Recent advances in sensor, storage, processing and communication technologies have enabled the automated recording of data, leading to fast and continuous flows of data, referred to as data streams. Examples of data streams are the web logs and web page click streams recorded by web servers, transactions like credit card usage, data from network monitoring and sensor networks, video streams such as images from surveillance cameras, news articles in an RSS reader etc.

2.1 Stream Classification and Concept Drift

The task in data stream classification is the application of a machine learning algorithm that will learn incrementally to classify incoming items. For successful automatic classification of data streams we are not only looking for fast and accurate incremental algorithms, but also for complete methodologies that can detect and quickly adapt to time varying concepts. This problem is usually called “concept drift” and describes the change of concept of a target class with the passing of time.

There are two main kinds of concept drift: a) Instant (abrupt or sudden) and b) gradual. In the case of *abrupt concept drift* concepts alter instantly. Consider a machine learning based news reader. The user after the purchase of a car might stop, momentarily be interested in articles on the topic. The classifier should be able to instantly detect the drift and rapidly adapt. An example of *gradual concept drift* is the spam filtering problem. Spam messages change (or evolve, they become harder to identify by filters) with a certain rate. Algorithms should be able to emphasize in those changes and make more accurate future classifications.

An interesting situation that can occur in instant and gradual concept drift is the repetition of previous concepts. For example in the personalized news reader, user could re-gain interest in topics that was interested in the past. Furthermore, there are some special spam messages that re-appear in certain time periods (e.g. Christmas). This subtype of concept drift has been noted as “recurrent themes” [7].

As stated in [17], an ideal concept drift handling system should:

1. Quickly adapt to concept drift.
2. Be robust to noise and distinguish it from concept drift.
3. Recognize and reacts to reoccurring contexts.

2.2 Stream Clustering

The task in the stream clustering problem is to discover groups of similar object in a sequence of items. The problem is far more interesting from the case of static data because we have no prior knowledge of the data and thus, groups must be created incrementally. As noted in [3] the main requirements for methods dealing with clustering data streams are: a) compactness of representation, b) fast, incremental processing of new data points c) clear and fast identification of "outliers". Well known representatives of streaming clusterers are COBWEB [6], BIRCH [21], STREAM [13] and CluStream [1]. In our approach, we use a combination of incremental classification and incremental clustering in order to tackle with the problem of concept drift

3. Related Work

As other researchers have noted [17], methodologies proposed for tackling with concept drift can be divided in three main groups.

Instance selection: In this category, proposed systems try to select the most appropriate set of past cases in order to make future classifications. Typical representatives of this group are time-window based methods, where the classifier is always trained from a fixed or adaptive window of instances. In window approaches, we make the assumption that older instances are useless for classification of new data and therefore, adapting to concept drift is synonym to successfully forgetting old instances / knowledge. Examples of this group can be found in [5, 10, 19].

Instance weighting: In this group, we assume that old knowledge becomes less important as time goes by. All instances are taken under consideration for building classification models, but this time, new instances have larger effect in the model than the older ones. To achieve this goal, a weighting scheme is defined (aging function) that assigns weights to instances [10]. A classifier to be used as a base learner in this approach must fulfill two requirements: a) must be incremental and b) must be able to consider weights. SVMs and Naïve Bayes Classifier are two examples that meet the aforementioned criteria.

Ensemble Methods: The main idea behind ensemble methods is to have a number of classifiers that are effective only on a certain concept. The important part is to identify correctly in which concept a future instance belongs to and apply the most appropriate classifier. Representatives of this group can be found in [18] and [11].

An additional group of methods could be the category of *Quickly adapting incremental* algorithms: This group consists of methods like CVFDT [8] that try to adapt rapidly to the newest batch of data in order to cope with the drift. Finally, a recent work that also exploits clustering for detection of concept but in different direction is [16].

4. Probabilistic Representation

In classification tasks, data is usually represented in an *n*-dimentional feature space. Every item (data point) is represented as a vector of size n $\vec{x} = (f_1, f_2, ... f_n)$. We generally assume that geometrically close items do belong in the same class. See for example Figure 1a. In Figure 1b we have the representation of the streaming classification data. Numbers in objects denote sequence of arrival. Unfortunately this representation in steaming classification problems is insufficient. As can be seen in Figure 1c, small distance between objects can not guarantee that they belong in the same class.

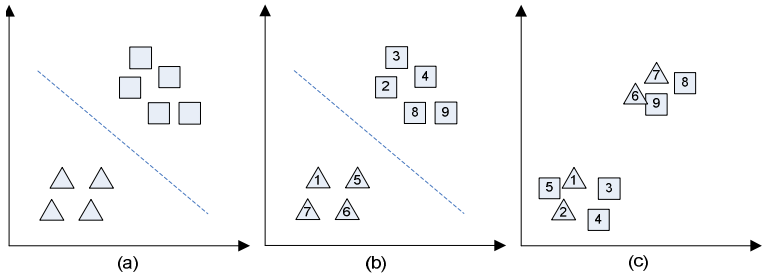

Fig. 1. (a) Static Classification (b) Streaming Classification (c) Streaming Classification with concept drift.

To overcome the problems of the aforementioned traditional representation, we propose a mapping function from conventional feature space into a probabilistic / conceptual space more suitable for detecting and adapting to concept-drift.

First, we separate data into a number of small batches of examples. When a batch has fully arrived, we calculate its mapping to the probabilistic space, which is a vector that we call "*Conceptual Vector*". A Conceptual Vector will be constructed out of a set of *conceptual feature-sets* (*CF*) which are transformations of the original features. Let's assume that an example of a dataset is represented as a simple vector:

$$\vec{x} = (f_1, f_2, ... f_n, c_j)$$

Where f_i is the value of the *ith* feature, nominal or numeric, and c_j is the belonging class of the instance. Number of attributes is *n* and number of classes is *m*. Each feature f_i is mapped / assigned into a conceptual feature-set as follows:

$$CF_i = \begin{cases} \{P_{i,j}^v : j = 1..m, v \in V_i\}, & \text{if } f_i \text{ is nominal} \\ \{\mu_{i,j}, \sigma_{i,j} : j = 1..m\}, & \text{if } f_i \text{ is numeric} \end{cases}$$

For nominal attributes $P_{i,j}^v = P(f_i = v \mid C_j)$ where $i \in [1,n], j \in [1,m], \ v \in V_i$, where V_i is the set of the potential values of the attribute *i*. Typically, $P_{i,j}^v$ is considered to

be equal to: $n_{v,j} / n_j$ where $n_{v,j}$ is the number of training samples of class C_j having the value *v* for attribute i and n_j is the number of training samples belonging to C_j.

For continuous (numeric) attributes we use the mean (μ_{i,C_j}) and standard deviation (σ_{i,C_j}) of attribute f_i for training samples of class C_j.

Thus, the obtained Conceptual Vector would be $CV = (CF_1,...,CF_n)$

The expected average dimensionality would be $x\bar{v}m + y2m$, where *x* and *y* are the total number of nominal and numeric attributes respectively and $\bar{v}$ is the average number of labels for nominal attributes.

While instances arrive, we incrementally calculate the above statistics in order to be able to construct the conceptual vector immediately after the batch is finished.

The notion behind this representation is that every element of the conceptual vectors expresses in what degree a feature characterizes a certain class. For example, if we notice that in two different batches *P("brakes"|interesting)*, *P("road"|interesting)*, *P("wheel"|interesting)* are similarly high, we could assume that these batches might belong to the same or similar concept (e.g. to a concept where car-related documents are considered interesting). The same principal applies in numeric attributes. Batches of instances that have similar mean values for many attributes, should be close conceptually.

Conceptual Euclidean Distance between two Conceptual Vectors can be defined as follows:

$$\text{Distance}(CV_1, CV_2) = \sqrt{dis(CF_{1,1}, CF_{2,1}) + ... + dis(CF_{1,n}, CF_{2,n})}$$

Where $dis(CF_1, CF_2) = \left(CF_1^1 - CF_2^1\right)^2 + ... + \left(CF_1^l - CF_2^l\right)^2$, and CF_i^j is the *jth* element of the *ith* conceptual feature-set, and *l* is the length of the feature set.

In Figure 2a, we see an example of abrupt concept drift in the probabilistic representation discussed here. Small circles represent conceptual vectors and numbers denote order of arrival. Note that from batch 4 and then on we suddenly go into concept 2 (instant concept drift). Then we return to concept 1. This also includes the recurrent theme type of concept drift discussed earlier. In Figure 2b, we see a gradually changing concept, like the spam example discussed earlier.

By observing Figure 2, we could set the requirements for the optimal solution for both types of concept drift. For the case of instant concept drift, we would like a single classifier for every different concept. Classifiers trained for older concepts should be maintained in case this concept re-appears in future. The classifier in this case will be "strengthened" with new examples instead of training from the beginning.

For the case of gradual concept drift, we should have more classifiers distributed in the probabilistic space. Less classifiers will produce better generalization whereas more classifiers will lead to creation of a great number of "experts". Again additional classifiers can be utilized in case of recurrent themes. In Figure 2, with "x" we represent the classifiers ideally distributed in the probabilistic space for both cases.

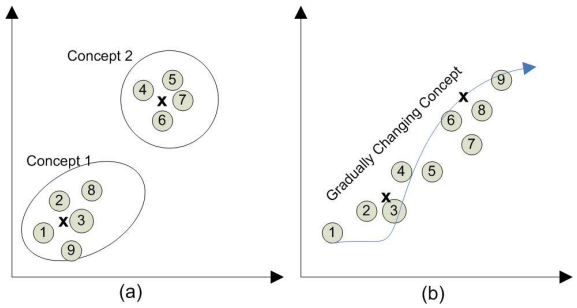

Fig. 2. Instant (a) and Gradual (b) Concept Drift in the new probabilistic representation. Circles represent Conceptual Vectors of Batches. Numbers denote sequence of arrival.

5. The CCP Framework

Considering a) the requirements for systems dealing with concept drift (see section 2.1) and b) the probabilistic representation introduced in the previous section, we propose a data stream classification framework that exploits incremental clustering in order to group the stream's conceptual vectors. The main aim of clustering is the online identification of groups of concepts and the use and maintenance of cluster-specific classifiers.

The main components of the framework are:

a) *A mapping function (M)*, that transforms data into the probabilistic representation.

b) *An incremental clustering algorithm (R)*, that will group conceptual vectors into clusters.

c) *An incremental classifier (P)* for every cluster / concept discovered.

The framework (*CCP – C*onceptual *C*lustering & *P*rediction), in brief, works as follows. After a batch (B_{i-1}) is received, its conceptual vector is created (CV_{i-1}). The incremental clusterer *(R)* assigns CV_{i-1} into cluster *c* (or creates a new cluster if necessary). Knowing that this *CV* is within the cluster / concept *c*, the corresponding classifier P_c can therefore be updated with all instances of this batch. Instances of next batch (B_i) are classified using classifier P_c. By classifying the current batch according to the classifier built from the cluster of the previous batch we make a kind of a "Locality Assumption". We assume that successive batches most of the time will belong to the same concept. This assumption generally applies especially when small batches are used. Final step is to update the clusterer. The pseudocode of the framework can been seen in figure 3. It is important to know that there is no need to maintain past Batches or Conceptual Vectors in memory. What is maintained is the clusters centers and the appropriate classifiers for every cluster.

As basic components, any incremental clustering algorithm could be exploited like the ones mentioned in section 2.2 or the basic Leader-Follower clusterer [4] presented in Figure 3b. For the classification component, any incremental classifier could be use (e.g. Incremental SVMs, Incremental Naïve Bayes).

Commenting on the requirements stated in [17](see Section 2.1) and the proposed framework:

a) System based on this framework will quickly adapt to concept drift as long as the batch size is small. It will start using the most appropriate classifier if it notices that the concept vector of the batch belongs to a different cluster (concept).

b) In this approach, being able to distinguish noise from drift is equal to identifying outliers in the clusering problem. Much work has been done successfully to this direction by researchers working on clustering and incremental clustering algorithms (see section 2).

c) Reoccurring contexts will be recognized as clusters.

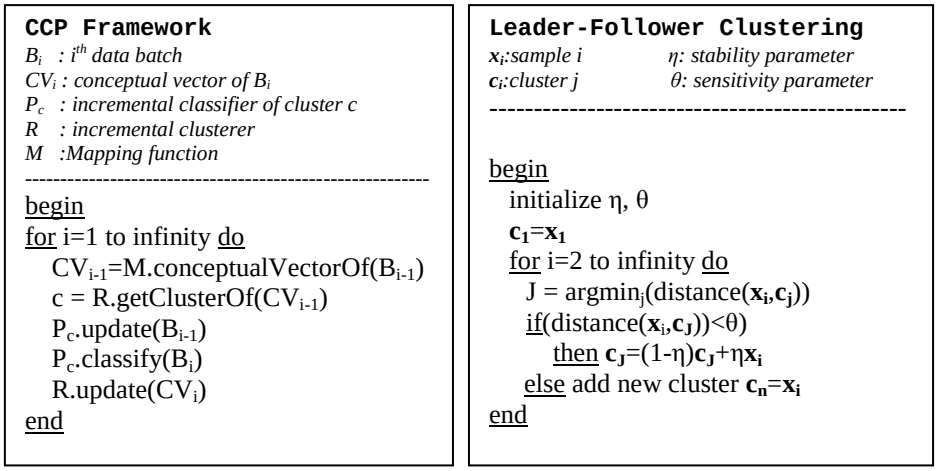

Fig. 3. (a) The main operation of *CCP* framework (b) The basic leader-follower clusterer

6. Evaluation

To evaluate the representation and solution discussed in this paper, we made a simple implementation of the proposed framework and compare it against three baseline methods that tackle concept drift in three real-world streaming textual datasets.

6.1 Datasets

The first two datasets (usenet1 and usenet2) were created from the well known 20 newsgroups collection from the UCI repository [12]. We kept messages from three discussion groups to simulate a stream of news articles. These datasets represent the news filtering problem. In news filtering, we have a system that accepts user feedback and is able to learn user interests in order to reduce irrelevant news articles shown and

reduce information overload. Hence, this is a two class problem, news articles are marked *interesting* or *junk.* Both datasets contain articles from the science/medicine, science/space, and recreation/sports/hockey groups. Each dataset is a stream of 1500 instances, split into five time periods. Each period contains 300 instances. After the end of each period concept drift occurs. Table I and II present which messages are considered interesting (+) or junk (-) in each period. We notice that the first (usenet1) dataset is a far more diverse dataset with all categories changing class in each period. The second dataset (usenet2) is more moderate in concept changes (2 out of 3 classes change concept every time). Both datasets represent sudden concept drift and contain recurrent themes. In preprocessing, we removed the headers from all messages and applied a Boolean bag-of-words [14] representation.

Table 1. Dataset Usenet 1

Usenet1 **From**	**0**	**301**	**600**	**900**	**1200**
group To	**300**	**600**	**900**	**1200**	**1500**
med	+	-	+	-	+
space	-	+	-	+	-
baseball	-	+	-	+	-

Table 2. Dataset Usenet2

Usenet2 **From**	**0**	**301**	**600**	**900**	**1200**
group To	**300**	**600**	**900**	**1200**	**1500**
med	+	-	-	-	+
space	-	+	-	+	-
baseball	-	-	+	-	-

The Spam Assassin collection comes in four parts (folders): “spam”, “spam2”, “ham”, and “easy ham” which is a collection of more easily recognized legitimate messages. In order to convert this collection into a longitudinal data set we extracted the date and time that the mail was sent. Then we converted the time into GMT time. Date was also changed where needed. We stamped each mail with its date and time. If an email occurred more than once in the corpus we kept all copies (because many times users get the same spam messages over and over). All attachments were removed. The Boolean bag-of-words approach was used for representing the emails. We chose the SpamAssasin (http://spamassasin.apache.org/) data collection because a) Every mail of the collection is available with the headers, thus we were able to extract the exact date and time that the mail was sent or received, and b) It contains both spam and legitimate (ham) messages with a decent spam ratio (about 20 %). This dataset consists of 9324 instances and represents the gradual concept drift.

All datasets are available in Weka (ARFF) format at following URL: http://mlkd.csd.auth.gr/datasets.html

6.2 Methods

To compare the framework and representation proposed we made a comparison with three baseline stream learning methodologies.

Simple Incremental Classifier(SIC): In this simple method we maintain a classifier that has the ability to update its knowledge for every incoming instance. Such classifiers are the Naïve Bays classifier, the *k*NN classifier etc.

Time Window (TW): In this case, we have a machine learning algorithm that classifies incoming instances based on the knowledge of the latest *N* instances. Main advantage of this method is that the classifier is always focused on the latest batch of

data. The main disadvatanges are that it lacks of generalization due to smaller set of examples used and the fact that it disregards older knowledge that might be useful for future classification in case of the recurring themes.

Weighted Examples (WE): This methodology consists of a base classifier that is incremental but can also comprehend weighted learning. Bigger weights are assigned to most recent instances in order to force the classifier focus on new concepts and forget old ones.

6.3 Experimental Setup

We evaluated the four methodologies in the three datasets discussed above. All methodologies and algorithms are implemented with the aid of the Weka [20] API.

For Time Windows we tested two classifiers (Naïve Bayes and Support Vector Machines – both widely used classifiers in Text Classification [15]), and three different time window sizes (*w=100, w=150, w=300*). SVMs are used with default Weka parameter settings (SMO, Linear Kernel, *C*=1, *L*=0,001). For Weighted examples, with preliminary experimentation we found out that a decent way to update weights is according to $w(n) = w(n-1) + n^2$. Where *w(n)* is the weight of the *n-th* instance.

Our framework was implemented with the Mapping function discussed in section 4, the basic Follower-Leader Clustering described in [4] as the clustering component and incremental Naive Bayes classifier. By experimentation we observed that a batch size around 50 instances is sufficient enough. With larger batches, the aforementioned locality assumption loses its validity, whereas smaller batches are not sufficient for calculating the summary probabilistic statistics.

The pseudocode of the Follower-Leader clustering is presented in Fig. 3b. Parameter θ actually determines how sensitive the clustering will be in creating new clusters. Small values of θ tend to create many new clusters. Parameter η sets the stability of the cluster centers. It determines how rapidly old centers will move into the direction of new points arriving. Note that studying parameter variation and how it affects the clustering performance is out of the scope of this paper.

We additionally included in the experiments a benchmark version of our framework (dubbed Oracle) where we manually provide the perfect clustering assignments to the system. This aims at approximating the maximum performance that can be achieved using the CCP framework.

6.4 Results and Discussion

Table one depicts the results of the experiments in the three concept-drifting textual datasets discussed in the paper. The first observation is that even a basic implementation of the CCP framework achieves better performance than all other methods in all datasets.

The second best performance in general is achieved by weighted examples (WE). The main reason that WE outperforms the rest of the methods is a) because it is trained from more data than time windows (TW) and therefore has better generaliza-

tion capabilities and b) because it focuses on the most recent documents and thus adapts faster than simple incremental classifier (SIC) to concept drift.

Table 3. Accuracy of the four methods in the three datasets

		Usenet1	Usenet2	spam
Simple Incremental	**NB**	0.59	0.73	0.75
TimeWindow (w=100)	**NB**	0.56	0.60	0.60
	SVM	0.60	0.63	0.67
TimeWindow (w=150)	**NB**	0.59	0.62	0.64
	SVM	0.63	0.66	0.69
TimeWindow (w=300)	**NB**	0.58	0.70	0.62
	SVM	0.59	0.72	0.70
CCP (Oracle)	**NB**	**0.81**	**0.80**	-
CCP (Leader-Follower)	**NB**	0.75	0.77	0.93
Weighted Examples	**NB**	0.67	0.75	0.91

On the other hand, CCP outperforms WE mainly because WE lacks of real concept drift detection mechanism and consequently can not adapt fast enough when drift occurs. In Figure 4 we see the average accuracy over fifty instances for the Clustering and WE methods for the Usenet1 dataset. Note the sudden collapses of WE's accuracy in drift time-point (300, 600, 900, 1200). In every case, CCP manages to recover much faster from the drift and maintain a nearly stable performance. Most notably, at the last drift point, CCP recognizes the recurrent theme and does remains accurate in classifications.

Usenet1 is the most demanding dataset due to many changes in concepts. Thus, algorithms have difficulties in coping with the drift, especially the ones that lack of detection mechanism. This fact explains the noticeable advantage of CCP over WE.

Usenet2 dataset is especially complicated for the CCP because after drift time-points there are some sub-concepts that remain unchanged and therefore the clustering problem here is more difficult. This explains the small differences between WE and CCP, although CCP still manages to outperform all methods.

The spam corpus is obviously the easiest dataset mainly because it contains no sudden changes in concepts. The Leader Follower clusterer created three clusters in this dataset one significantly larger and two smaller ones maybe discovering some recurrent messages.

Support Vector Machines naturally performed better than the Naïve Bayes classifier in the TW approach. Furthermore, results show that windows of bigger size do not guaranteed better performance obviously because of data becoming outdated fast.

Finally, the performance of Oracle, strongly underlines the fact that there is further room for improvement by using more advanced incremental clustering algorithms.

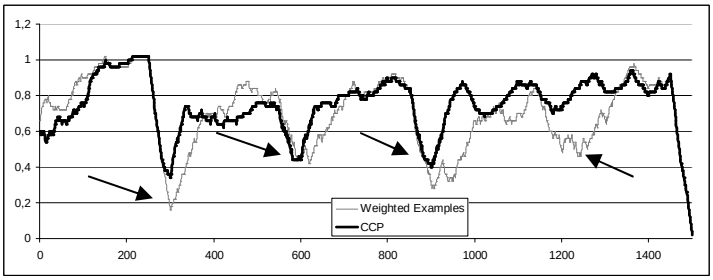

Fig. 4. Average accuracy over 50 instances for WE and CCP. With arrows are pointed out the drift time points

7. Conclusions and Future Work

In this paper we discussed the deficiency of conventional data representation and propose a new probabilistic representation for problems involving concept drift. Additionally we propose a basic framework that can exploit this representation to identify and adapt to drift. We experiment with a basic implementation and three textual datasets retrieving interesting results encouraging further experimentation.

It is in our immediate plans to test the framework more extensively by using more advanced incremental clustering algorithms and classifiers and supplementary datasets from different domains. At present we work into integrating this framework into our adaptive RSS aggregator PersoNews [2] which is currently using the framework described in [9], in order to evaluate it in a real application and test scalability.

8. References

1. Aggarwal, C., Han, J., Wang, J., and Yu, P. *A framework for clustering evolving data streams*. in *29th VLDB Conference*. 2003
2. Banos, E., Katakis, I., Bassiliades, N., Tsoumakas, G., and Vlahavas, I. *PersoNews: A Personalized News Reader Enhanced by Machine Learning and Semantic Filtering*. in *5th International Conference on Ontologies, DataBases and Applications of Semantics (ODBASE 2006)*. 2006. Montpellier, France: Springer-Verlag: p. 975-982.
3. Barbar, D., *Requirements for clustering data streams.* SIGKDD Explor. Newsl., 2002. **3**(2): p. 23-27.
4. Duda, R.O., Hart, P.E., and Stork, D.G., *Pattern Classification*. 2000: Wiley-Interscience.
5. Fan, W. *Systematic data selection to mine concept-drifting data streams*. in *Tenth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. 2004. Seattle, WA, USA: ACM Press: p. 128-137.
6. Fisher, D., *Iterative Optimization and Simplification of Hierarchical Clusterings.* Journal of Artificial Intelligence Research, 1996. **4**: p. 147-180.

7. Forman, G. *Tackling Concept Drift by Temporal Inductive Transfer*. in *29th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2006. Washington, USA: ACM Press: p. 252-259.
8. Hulten, G., Spencer, L., and Domingos, P. *Mining time-changing data streams*. in *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2001. San Francisco, California: ACM Press: p. 97-106.
9. Katakis, I., Tsoumakas, G., and Vlahavas, I. *Dynamic Feature Space and Incremental Feature Selection for the Classification of Textual Data Streams*. in *ECML/PKDD-2006 International Workshop on Knowledge Discovery from Data Streams*. 2006. Berlin, Germany: Springer Verlag: p. 107-116.
10. Klinkenberg, R., *Learning Drifting Concepts: Example Selection vs. Example Weighting* Intelligent Data Analysis, Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift, 2004. **8**(3): p. 281-200.
11. Klinkenberg, R., *Boosting Classifiers for Drifting Concepts.* Intelligent Data Analysis, Special Issue on Knowledge Discovery from Data Streams (accepted for publication), 2006.
12. Newman, D.J., Hettich, S., Blake, C.L., and Merz, C.J., *UCI Reprository of ML databases [http://ics.uci.edu/~mlearn/MLRepository.html]*. 1998, University of California, Department of Information and Computer Science: Irvine, CA.
13. O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., and Motwani, R. *High-Performance Clustering of Streams and Large Data Sets*. in *ICDE 2002*. 2002
14. Salton, G., Wong, A., and Yang, C.S., *A Vector Space Model for Automatic Indexing*. Communications of the ACM, 1975. **18**(11): p. 613-620.
15. Sebastiani, F., *Machine Learning in automated text categorization.* ACM Computing Surveys, 2002. **34**(1): p. 1-47.
16. Spinosa, E.J., Carvahlo, A.P.L.F.d., and Gama, J. *OLINDDA: A cluster-based approach for detecting novelty and concept drift in data streams*. in *22nd Annual ACM Symposium on Applied Computing*. 2007: ACM Press: p. 448-452.
17. Tsymbal, A., *The problem of concept drift: definitions and related work, Technical Report TCD-CS-2004-15*. 2004, Department of Computer Science, Trinity College: Dublin, Ireland.
18. Wang, H., Fan, W., Yu, P.S., and Han, J. *Mining concept-drifting data streams using ensembles classifiers*. in *Ninth ACM SIGKDD International conference on Knowledge Discovery and Data Mining*. 2003. Washington, D.C.: ACM Press: p. 226-235.
19. Widmer, G. and Kubat, M., *Learning in the Presense of Concept Drift and Hidden Contexts.* Machine Learning, 1996. **23**(1): p. 69-101.
20. Witten, I. and Frank, E., *Data Mining: Practical Machine Learning tools and techniques", 2nd Edition*. 2005, San Francisco.
21. Zhang, T., Ramakrishan, R., and Livny, M. *BIRCH: An efficient data clustering method for very large databases*. in *ACM SIGMOD International Conference on Management of Data*. 1996. Montreal, Canada: p. 103-104.