ANÁLISE DE FLUXOS DE DADOS ECONOMICAMENTE EFICIENTES

ROBERTO LOURENÇO DE OLIVEIRA JÚNIOR

ANÁLISE DE FLUXOS DE DADOS ECONOMICAMENTE EFICIENTES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais - Departamento de Ciência da Computação como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Adriano Veloso
Coorientador: Wagner Meira Jr.

Belo Horizonte
Abril de 2014

ROBERTO LOURENÇO DE OLIVEIRA JÚNIOR

ECONOMICALLY-EFFICIENT DATA STREAM ANALYSIS

Dissertation presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais - Departamento de Ciência da Computação in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Adriano Veloso
Co-Advisor: Wagner Meira Jr.

Belo Horizonte
April 2014

Oliveira Júnior, Roberto Lourenço de

O48e Economically-Efficient Data Stream Analysis / Roberto Lourenço de Oliveira Júnior. — Belo Horizonte, 2014
xxiv, 45 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais - Departamento de Ciência da Computação
Orientador: Adriano Veloso Coorientador: Wagner Meira Jr.

1. Computação - Teses. 2. Aprendizado do computador. I. Título.

CDU 519.6*82(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Economically-efficient data stream analysis

ROBERTO LOURENÇO DE OLIVEIRA JÚNIOR

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. ADRIANO ALONSO VELOSO - Orientador
Departamento de Ciência da Computação - UFMG

PROF. WAGNER MEIRA JÚNIOR - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. ADRIANO CÉSAR MACHADO PEREIRA
Departamento de Ciência da Computação - UFMG

PROF. RENATO ANTÔNIO CELSO FERREIRA
Departamento de Ciência da Computação - UFMG

PROF. SRINIVASAN PARTHASARATHY
Department of Computer Science and Engineering - OSU

Belo Horizonte, 15 de abril de 2014.

Eu dedico este trabalho a minha Vó Tina pelo amor incontestável, imensurável e incondicional. Fique em paz Vó, te amo.

Acknowledgments

I had during my MSc the pleasure of live together with several persons, each one unique on their essence. All of them contributed in some way to completion of this work. They caused alteration in the space-time that created sequences of random events that conducted me to complete this work. This space is reserved to express my sincere thanks to of them.

I would like to first thank four women that are fundamental in all of my achievements. My Mom, Aline and Ameline, my sisters, and Isabella. They always supported me. They are an inexhaustible knowledge source that helps me maintain the critical look at the world and allows me go further. My Mom and Aline and Ameline, you are my example and my motivation. Isabella, my partner of all moments, you are responsible to push me to face the unknown and give me strengths to win. These 4 women are the pillars of this achievement, then I would like to sincerely express my gratitude for all the support.

An achievement like this is not complete without friendship. I would like to acknowledge my faithful comrades CambeLOL, for helped me see beyond algorithms. To my housemates, Cristiano and Daniel, thank you very much for the great friendship. Colleagues from MSc program and SPEED Lab, thank you so much for sharing their knowledge with me.

This work would not be possible without an advisement, due to this, I would like to express my sincerely gratitude to Dr. Adriano Veloso. His guidance was determining to the success of this work. I also would like to thank my co-advisor Dr. Wagner Meira Jr and this thesis committee: Dr. Adriano Cesar, Dr. Renato Ferreira e Dr. Srinivasan Parthasarathy.

I have to specially acknowledge Dr. Srinivasan Parthasarathy, Dr. Wagner Meira Jr and Dr. Adriano Veloso by the opportunity of visit the Data Mining and Research Lab. This visit was one of the most exciting experiences of my life. I extend my acknowledgments to all my friends of Data Mining and Research Lab, for the friendship and knowledge sharing.

During my staying at Columbus I had the opportunity to meet people from all parts of the globe. For all of them, Brazilians, Americans, Colombians, Chinese, Korean, Italian, Spanish, and Chileans thank you all for helped me face the language challenges, the extreme weather and homesickness.

Thank everyone that believes in my capacity to complete this work and offered their shoulders to allow me see further.

"What would be the importance in differentiate among the order of tea's ingredients?"
(David Salsburg, The Lady Tasting Tea)

Resumo

Processar dados na forma de fluxo tem se tornado um interessante modelo para extrair informação de grandes conjuntos de dados. Entretanto, tal modelo de processamento impõe restrições em termos de memória e tempo. No caso de algoritmos de aprendizado de máquina, tais como classificação e agrupamento, há outra restrição chamada Mudança de Conceito, em que consiste de mudanças nos dados causadas por falhas ou aparecimento de outras fontes de dados, evolução natural dos dados, entre outras razões. Neste trabalho nós atacamos os desafios de fluxos de dados propondo nosso método Amostragem Seletiva Economicamente Eficiente, que seleciona instâncias de treinamento relevantes a cada passo, mantendo assim o conjunto de treinamento pequeno enquanto provê ao modelo preditivo duas capacidades: Adaptação e Memorização. Adaptação é a capacidade do modelo preditivo adequar-se ao novo conceito, enquanto memorização é a capacidade do modelo preditivo recuperar-se da mudança de conceito. Prover ambas as capacidades simultaneamente ao modelo preditivo leva a um problema de conflito de objetivos, e nosso método aplica noções da economia para achar o melhor balanceamento entre adaptação e memorização. Nós realizamos análises do nosso método em várias aplicações contra algoritmos representativos do estado da arte. As avaliações revelam que nosso método superou os outros métodos em termos de redução de erro (acima de 14%) e redução de recursos de treinamento (ordens de magnitude). **Palavras-chave:** Fluxos de Dados Evolutivos, Aprendizado de Máquina, Amostragem Seletiva, Eficiência Econômica.

Abstract

Process data in streaming has becoming an interesting model to extract information from large data sets. However, such processing model poses restrictions in terms of memory and time. In case of learning algorithms, such as classification and clustering algorithms, there exists an another issue called Concept Drift, which consist of changes in the data caused by failures or appearance of new data sources, natural evolution of data, among others reasons. In this work we address data stream challenges by proposing our method Economically-Efficient Selective Sampling, which selects relevant training instances at each time step, so that training sets are kept small while providing to the predictive model two capabilities: adaptiveness and memorability. Adaptiveness is the capability to the predictive model suit itself to concept drift, while memorability is the capability to recover itself from concept drifts. Provide simultaneously both capabilities to the predictive model lead to a conflicting-objective problem, and our method employ notions of Economics in order to find a proper balance among adaptiveness and memorability. We performed the analysis of our method in several applications against representative state-of-the-art algorithms. Evaluation reveals improvements in terms of error reduction (up to 14%) and reduction of training resources (by orders of magnitude).

Palavras-chave: Evolving Data Streams, Machine Learning, Selective Sampling, Economic Efficiency.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In the last years data generated has growing rapidly exceeding 2.8 zetabytes (2.8 trillion gigabytes) in 2012 as reported in the IDC surveyReinsel and Gantz [2012]. This growing data is a potential source of interesting and useful knowledge hidden in patterns not explicit, and discover such patterns represents a task that is impossible to be manually performed. In this context, data mining, machine learning and knowledge discovery methods have been proposed to automatically acquire interesting, non-trivial, previously unknown and ultimately understandable patterns from large data setsBramer [2007].

However, as the data grows rapidly, traditional methods of data analysis are unable to scale on huge amounts of data. Furthermore, store all generated data has becoming challenging, in particular for a class of applications that produces data in data streams Gaber et al. [2005]. Data streams can be seen as a sequence of data, possible unbound, that arrives continuously at time-varying. Applications that produces data as stream include network monitoring, security, telecommunication data management, Web applications, and sensor networks.

Traditional data mining methods focus on static environments, where patterns hidden in data are fixed and each register can be accessed more than once. In data mining, as well as machine learning, a popular task is classification, which is a process that automatically builds a classification model by learning from a set of previously labeled data with predefined classes (i.e., the training-set) the underlying characteristics that distinguish one class from another. The success of classification methods relies on their ability to relate non obvious patterns in data to each class. However, these methods frequently fail to successfully process data streams because of two factors: overwhelming data volume and changes of data's nature. Changes of data's nature receive a special name, **Concept Drift**, and consists of the changes caused by failures

on the data source, natural data's evolution, appearance of new data source, among others. This phenomena leads usual learning algorithms to a drastic drop in prediction accuracy. The data stream environment and its constraints motivated researchers to develop new approaches to face such challenges.

Approaches to face concept drift basically divide into forgetting and adaptation methods. Forgetting methods, such as sliding-windows, allows to limit the number of processed data and to react to changes. Adaptation methods include identification of concept drifts, model manipulation and prediction model ensembles. Such methods consist to adapt the prediction model as soon as a concept drift is identified.

Another possible way to cope with data stream challenges is to employ selective sampling approaches in order to focus only on the most relevant training examples. Such set of examples (e.g. training set) are kept as small as possible to ensure fast learning. Also, examples should be selected so that the resulting training set provides sufficient resources to enable the corresponding classifier, adaptiveness (capability to suit itself to concept drifts) and memorability (capability to recover itself from momentaneous drifts)

1.1 Goals

In this work we propose a wrapping for learning algorithms called **E**conomically-**E**fficient **S**elective **S**ampling (EESS). EESS aims to select training examples taking into account two important properties, which we define as **adaptiveness** and **memorability**. Informally, adaptiveness enables the classifier to adapt itself to concept drifts, and thus, improving adaptiveness involves incorporating fresh examples into the training-set, while discarding obsolete ones. Memorability, on the other hand, involves retaining examples belonging to pre-drift distributions, therefore enabling the classifier to recover itself from concept drifts.

We hypothesize that adaptiveness and memorability are both necessary to make classifiers robust to concept drifts. However, given their antagonistic natures, improving both properties may lead to a conflicting-objective problem, in which the attempt to improve memorability further may result in worsening adaptiveness. Thus, we tackle the problem by proposing selective sampling algorithms based on multi-objective optimization, that is, we propose to select training instances so that the resulting classifier achieves a proper balance between memorability and adaptiveness. Our algorithms are based on central concepts in Economics, namely *Pareto* and *Kaldor-Hicks* efficiency criteria [Palda, 2011; Kaldor, 1939; Hicks, 1939]. The Pareto Efficiency criterion infor-

mally states that "when some action could be done to make someone better off without hurting anyone else, then it should be done." This action is called Pareto improvement, and a system is said to be Pareto-Efficient if no such improvement is possible. The Kaldor-Hicks criterion is less stringent and states that "when some action could be done to make someone better off, and this could compensate those that are made worse off, then it should be done."

This work aims to present our approach of selective sampling EESS and evaluate it over a set of experiments against state-of-the-arts algorithms. In this experiments we validate our hypothesis about the requirement of adaptiveness and memorability as well as our model using economic efficiency criteria.

1.2 Contributions

The main contribution of this work is to exploit the intuition behind the aforementioned concepts for devising new algorithms for data stream analysis. In practice, we claim the following benefits and contributions:

- We formulate simple-to-compute yet effective utility measures that capture the notions of adaptiveness and memorability. For instance, the similarity between instances that are candidate to compose the current training set and the target message, as well as the freshness of the candidate instances, are measures that tend to privilege adaptiveness. In contrast, candidate instances are also randomly shuffled, thus privileging memorability. These utility measures result in a utility space, and the extent to which each candidate message contributes to adaptiveness and memorability depends on where it is placed in this space.

- We exploit the concept of Pareto Efficiency by separating instances (viewed as points in the utility space) that are not dominated by any other message. These instances compose the Pareto frontier [Palda, 2011], and instances lying in this frontier correspond to cases for which no Pareto improvement is possible. These instances privilege either adaptiveness or memorability, and thus they are selected to compose the current training set from which the classifier is built.

- We exploit the concept of Kaldor-Hicks Efficiency by selecting an additional set of instances that, although not lying in the Pareto frontier, correspond to a positive trade-off between adaptiveness and memorability. These instances are selected to compose the current training set from which the classifier is built.

- Our algorithms operate either on an instance-basis or in batch-mode, by employing classification models based on sentiment rules that are kept incrementally as the stream evolves and training sets are modified.
- We implemented our framework in MOA system and provide the implementation in Machine Learning Repository of UFMG [Veloso and Dafe, 2012] as well as all datasets;

The main contributions of this work were accepted to Symposium of Special Interest Group On Information Retrieval 2014 (SIGIR.) [Lourenco Jr. et al., 2014].

1.3 Text Organization

The remains of this work are organized as follows: Chapter 2 presents the basics of data stream mining. In particular, definitions of data streams, concept drift, as well as types of stream learners and their applications are shown. In Chapter 3 we describe Economically-Efficient Selective Sampling (EESS) algorithm. Chapter 4 presents the experimental results of our approach compared to representative of the state-of-the-art in four different scenarios. Finally, Chapter 5 concludes this works with a discussion on completed work and potential future investigations.

Chapter 2

Data Streams Learning

In this chapter we define data stream learning concepts and present existent approaches to deal with concept drifts. In Section 2.1 we define the learning processes as a machine learning task, the data stream environment and its issues, focusing in concept drift phenomena. In Section 2.2 we present a summarization of learning techniques on data streams. Section 2.3 is reserved to present some relevant recent works in data stream learning.

2.1 Machine Learning, Data Stream and Concept Drifts

Machine learning aims to use computers to find patterns in data and infer over future data, even when the process that generated the data is a priori unknown [Alpaydin, 2004]. Applications of machine learning comes from classification of customers credit to sentiment analysis in textual data, from patterns recognition in images to identification of spams in email, etc.

There are exist different methodologies to learn from data. The two most known are supervised and unsupervised learning. However, another machine learning approaches are not limited to these methodologies, others approaches are active learning, semi-supervised learning, among others. In this work we focus in supervised learning for classification. The goal is predict a target categorical variable (a.k.a. class label) $y \in \mathcal{Y}$ given a set of input features $X \in \mathcal{R}^n$. A instance or example is one pair of (X, y) and a set of examples is the training set. By inspecting the training set, classification algorithms induce a classification model that can be represented by a function $f : X \mapsto \mathcal{Y}$, i.e., f maps input features to target classes $\mathcal{Y}$. This model find underlying

characteristic in the input features that are related to each class. Then the function f is used to predict the class label for all data in the test set, which is composed by instances in the form $(X, ?)$.

The most common use of supervised learning is offline setup. In that way, the classification algorithms requires the whole training set to induce the model. When the training set is not completely available the process of learning is called online learning. In this model the learning algorithm must update its model on-the-fly as the training instances arrives, while in the same time must be ready to perform predictions.

A particular case of online setup is the data stream model where data arrives in unbounded sequence of high-speed from one or more data sources. This characteristic pose the following constraints [Bifet, 2009]:

1. It is impossible to store all data from the data stream. Only small summaries of data stream can be computed and stored, and the rest of the information is thrown away;

2. The arrival rate of data stream tuples forces each particular element be processed essentially once, in real time, and then discarded;

3. The data nature or the distribution that generate the items may change over time. Thus, data from the past may become irrelevant or even harmful for the current procedure. This phenomena receives the name of *Concept Drift*.

Constraints (1) and (2) restricts the amount of memory available and the processing time. Constraint (3) require that learning algorithms react as quickly as possible to the changing of the data nature or distribution to avoid lose accuracy in the predictions. Data streams where Constraint (3) happens are called *Evolving Data Stream* and represents an important challenge to machine learning and data mining communities. Figure 2.1 shows the typical learning cycle of a learning algorithm in the data stream model. Each step has a related requirement [Bifet et al., 2010a]:

Requirement 1 The algorithm is passed the next available example from the stream;

Requirement 2 The algorithm processes the example, and then decides whether to upgrade or not its model. In the case where the update is needed it must be done without exceeding the memory bounds, and as quickly as possible;

Requirement 3 The algorithm is ready to accept the next example. On request it is able to supply a model that can be used to predict the class of unseen examples.

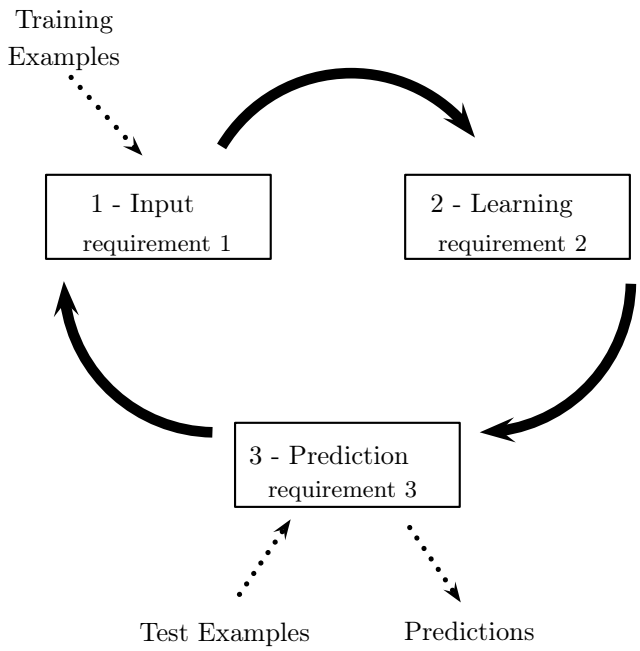


Figure 2.1. The data stream classification cycle [Bifet et al., 2010a].

In evolving data stream, **requirement 2** is more challenging due to the presence of concept drifts. A concept drift is an unforeseen substitution of one data source S_1 with another source S_2. Each data source has an underlying probability distribution that generate its data (generator process). The source change can be seen as an event in the current data source that causes change of the data's nature. The most popular example to present the problem of concept drift is that of detecting and filtering out spam e-mail. The distinction between unwanted and legitimate emails is user-specific and evolves over time. Another example is the monitoring of events on Twitter once the tweets are closely related to real world and [Tumasjan et al., 2010]. While real world events happen that may generate concept drifts in the twitter stream. Some events that have been studied are: health issues [Gomide et al., 2011; Paul and Dredze, 2011], traffic conditions [Ribeiro et al., 2012], political analysis [Guerra et al., 2011], among others.

The main assumption about concept drift is uncertainty about the future. It can be estimated or predicted, but there is not certainty [Žliobaitė, 2010a]. Kelly et al. [1999] presented three ways in which concept drift may occur:

- Proportion of classes, i.e., prior probabilities $P(c_1), ..., P(c_k)$, may change over time;
- class-conditional probability distribution, $P(X|c_i), i = 1, .., k$ might change;

- posterior probabilities $P(c_i|X), i = 1, ..., k$ might change as well.

The change of $P(X|c_i)$ represents that the features related to a class can change, disappear or new ones may raise. However, that does not imply the concept itself. That is why this kind of drift is referred as a virtual drift and the change of $P(c_i|X)$ as real drift. The distinction between both drifts is important from a practical point of view, however we do not do that in this work.

This phenomena may appear in different patterns over time: sudden, incremental, gradual and recurrent. Figure 2.2 presents those patterns, where the horizontal axis is time and the vertical is the data mean. A drift may happen *suddenly/abruptly* by switching from one concept to another (e.g.,monitoring a soccer match on twitter when some team scores a goal), *incrementally* consisting on many intermediate concepts in between (e.g.,a sensor slowly wears off and becomes less accurate), *gradually* (e.g.,relevant news topics change from dwelling to holiday homes, while the user does not switch abruptly, but rather keeps going back to the previous interest for some time) or be *recurrent* (e.g.,someone without political polarization may post comments about two opposite candidates depending on the results of opinion surveys).

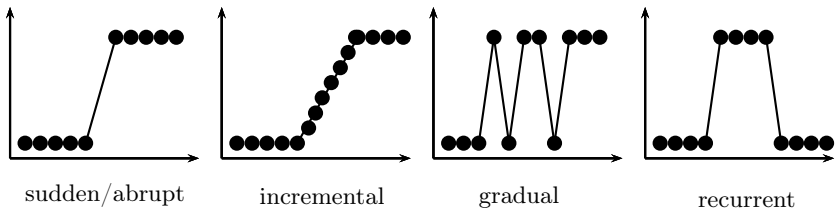


Figure 2.2. Patterns of changes over time [Gama et al., 2014].

According to Gama et al. [2014] most of adaptive learning techniques implicitly or explicitly assume and specialize in some subset of concept drifts. Many of them assume sudden non-reoccurring drifts. But in reality, mixtures of many types can be observed.

2.2 Adaptive Supervised Learning Methodologies

Many techniques have been proposed to allow classification of evolving data streams. Figure 2.3 is a taxonomy proposed by Žliobaitė [2010b], which groups the learning methods. This taxonomy is based on two dimensions "When" and "How" the algorithms learn and forget.

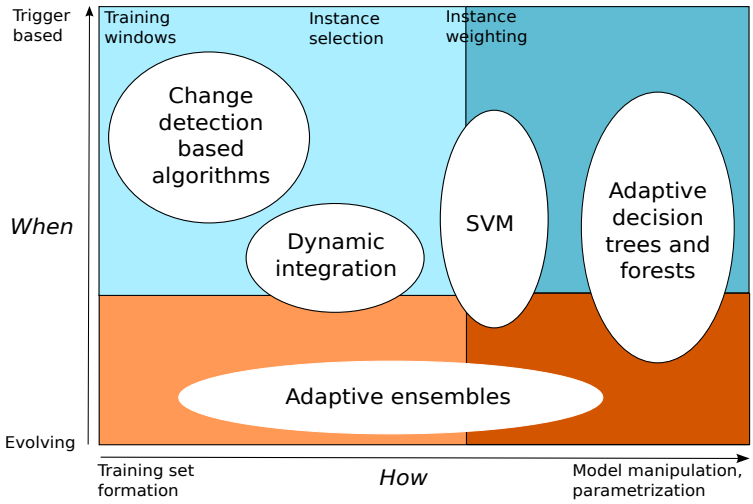


Figure 2.3. Adaptive supervised learning techniques[Žliobaitė, 2010b].

The "When" dimension ranges from gradually evolving to trigger based learners. Trigger based methods implements event detectors, usually drift detector, that indicates a need for model update. Such methods work well in data streams with sudden drift as the signal can be used to rebuild the whole model, instead of just updating it. On the other hand, evolving methods update the model gradually and usually react to changes without any drift detection mechanism. By manipulating ensemble weights or substituting models they try to adapt to changing environment without rebuilding the whole model.

The "How" dimension groups learners based on how they adapt. The adaptation mechanisms mainly involve example selection or parametrization of the base learner. Some mechanisms attempt to better forget outdated data. This strategy adjusts models to changing concepts. Next we highlight the usual adaptive strategies for evolving data streams.

Adaptive Ensemble: Consists of combine or select the classification output of several models. The combination or selection rules are often called fusion rules.

The ensemble techniques for concept drift may or not be dependent of base learners.In both cases, adaptivity is achieved by fusion rules, i.e., how weight each model prediction at each time step. The weight indicates the competence of the base learner expected to the current stream's moment.

Instance Weighting: Assigns weight to each example in training set in order to reduce the impact of examples from old concepts in the current model. This technique may be employed with a single learning algorithm or with an ensemble. The adaptivity

is achieved by systematic reformulation of the training set by assigning weights to each examples. Decay functions based on time or feature's space are used to weight examples.

Feature Space: Consists in manipulate the feature space to achieve adaptation. In evolving data streams features may disappear and new ones appear, thus, select the most relevant features to the current stream's status is mandatory.

Base Model Specific: Some traditional classification algorithms (Decision Trees, Support Vector Machines, etc.) may be modified to adapt in evolving scenarios. In case of decision trees, a simple approach is maintain a variable training window via adjusting its internal structures. In case of Support Vector Machines old support vectors are transfered and combined with recent training data.

Change Detectors: Change Detectors methods implements event detectors that indicates a need for model update. Such methods work well in data streams with sudden drift as the signal can be used to rebuild the whole model instead of just updating it. The detection methods usually cut the training window at change point.

Training Windows Consists to provide a limited amount of examples introduced to learner, thus eliminating those data points that come from an old concept. Each example updates the window and later the classifier is updated by that window. The key part of this algorithm lies in the definition of the window. In the simplest approach, sliding windows have fixed size and include only the most recent examples from the data stream. With each new data point the oldest example that does not fit in the window is thrown away. When using windows of fixed size, the user is caught in a trade-off. If it is chosen a small window size, the classifier will react quickly to changes, but may loose on accuracy in periods of stability. If it is chosen a large window size, may result in increasing accuracy in periods of stability, but to fail to adapt to rapidly changing concepts.

Adaptive Sampling The listed trigger based methods were using training windows. Another group of trigger based methods use instance selection. The incoming testing instances (unlabeled) are inspected and then, based on the relation between the testing instance and predefined prototypes or historical training instances, a set of training instances is selected to be the new training set. This work lies on this category of adaptive methodologies for evolving data streams.

In the next section we present some relevant and recent works in data stream learning.

2.3 Related Work

Núñez et al. [2007] proposed a method for keeping a variable training window by adjusting internal structures of decision trees. An ensemble of Hoeffding trees have been proposed in [Bifet et al., 2012], each tree is limited to a small subset of attributes. Gama et al.Gama et al. [2009] proposed a mechanism to discard old information based on a sliding window over time. Bifet and Gavaldà [2007]; Bifet and Gavaldà [2009] present an adaptive sliding window algorithm, called ADWIN, suitable for data streams with sudden drift. The main idea of ADWIN is whenever two sub-windows of W exhibit "distinct enough" averages, one can conclude that the corresponding expected values are different, and the older portion of the window is dropped out. The approach presented in [Koychev, 2000] suggests that the introduction of a time-based forgetting function, which makes the last observations more significant for the learning algorithms than the old ones, thus providing adaptiveness to the classifier. Klinkenberg [2004] compares example selection, often used in windowing approaches, versus example weights. Experiments with simulated concept drift scenarios show that both approaches can effectively select appropriate training-sets. In [Santana et al., 2011] the authors proposed an approach based on a training augmentation procedure, which takes as input a small training seed and then automatically incorporates new relevant training messages to the training-set. Classification models are produced on-the-fly using association rules, which are kept up-to-date in an incremental fashion, so that at any given time the model properly reflects the sentiments in the event being analyzed.

Some works have focused on feature similarity, such as Torres et al. [2011] that studied different methods for data stream classification and proposed a new way of keeping the representative data models based on similarity. Feng et al. [2013] extracted the concept from each data block using conditional and feature similarity probabilities. Žliobaitė [2011] proposed a family of algorithms called FISH, which uses time and space similarities between training examples as a way of dynamically fitting the training-set to the stream. It is proposed an unified view of the training-set formation, which is flexible with respect to the actual changes.

Masud et al. [2008] proposed a novel technique to overcome the lack of labeled examples by building a classification model from a training-set having both unlabeled and a small amount of labeled instances. Zhu et al. [2010] employed active learning

to produce a classifier ensemble that selects labeled instances from data streams to build an accurate classifier. In [Žliobaitė et al., 2013] and [Žliobaitė et al., 2011] active learning strategies are presented for data streaming that explicitly handle concept drift. They are based on uncertainty, dynamic allocation of labeling efforts over time, and randomization of the search space. Žliobaitė et al. [2010] propose a software system that implements active learning strategies, extending the MOA framework [Bifet et al., 2010a].

FISH algorithms proposed in [Žliobaitė, 2011] are the most close to our proposed approaches. However, FISH algorithms focus just on adaptiveness, by combing distance on time and space dimensions. A ranking function is employed to weight each example and select the ones with weight greater than an arbitrary or dynamic threshold. Another issue about FISH algorithms is that finding the optimal combination factor is an open problem and is solved by cross validation, which is a expensive process. On other hand, our proposed approach provides both memorability and adaptiveness to the classifier by selecting instances that improve both properties in the training-window. Also, we combine both aspects in a dynamic, multi-objective way, which makes our approach suitable for different types of concept drift: gradual, sudden, incremental and recurrent.

Chapter 3

Economically-Efficient Selective Sampling

In this chapter we present our selective sampling algorithm named **E**conomically-**E**fficient **S**elective **S**ampling (EESS) which, aims to deal with different types of concept drifts. However, as discussed above, there exist other challenges in data stream environment. Addressing to such challenges, we added EESS inside a framework containing components that face those issues. An overview of this framework can be seen in Figure 3.1.

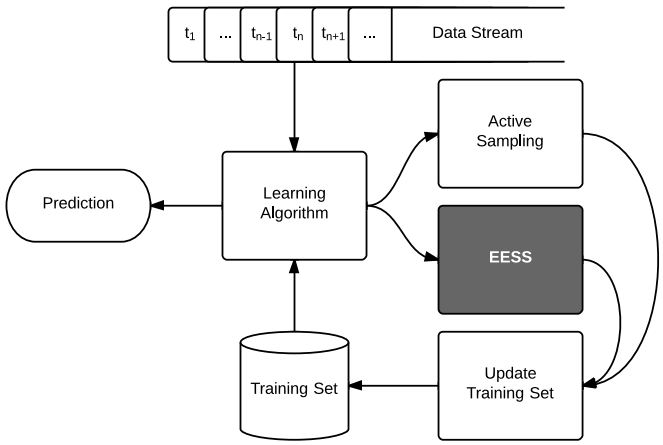


Figure 3.1. Data Stream Framework that wraps EESS.

The framework is composed by three main parts: A learning algorithm, our selective sampling approach EESS and, an active sampling strategy. As the learning task generally requires considerable computational resources, we chose an efficient classification model, capable to operate under time and memory restrictions. Also addressing to reduce computational resources, EESS is able to provide small training sets to the

learning algorithm reducing memory consumption as well as processing time. Training sets provided by EESS also provides two basic features: Adaptiveness and Memorability. We hypothesized that both features are essential to a learning model to be robust to concept drifts. The active sampling strategy aims to reduce labeling efforts as the data stream evolves. This is a challenging task, considered by Žliobaitė et al. [2012], one of the next challenges that machine learning researchers will focus for adaptive systems. We used a simple algorithm to choose streaming instances to be labeled with a budget control.

Thus, we treat limited resources with an efficient learning algorithm plus small training sets. Labeling efforts are reduced applying an active sampling algorithm and, concept drifts are deal by our Economically-Efficient Selective Sampling algorithm.

The remaining of this chapter we describe each part of the framework and the Economically-Efficient Selective Sampling algorithm. In Section 3.1 we present the learning algorithm chosen. In Section 3.2 we show the active sampling approach. Finally, the last two Sections are reserved to describe our approach. Section 3.3 is destined to present the concepts of adaptiveness and memorability. In Section 3.4 we show the Utility Space built from adaptiveness and memorability measures and introduce economic efficient criteria used to balance adaptiveness and memorability.

3.1 Association Rules and Classification Model

Next we describe classifiers composed of specific association rules, and how these rules are used for label-scoring. Such classifiers are built on-the-fly in an incremental fashion, being thus well-suited for data stream classification, as shown in [Santana et al., 2011].

Definition 1. *A association rule is $\mathcal{X} \rightarrow s_i$, where the antecedent $\mathcal{X}$ is a set of features, and the consequent s_i is the predicted label. The domain for $\mathcal{X}$ is the features of the training set $\mathcal{D}_n$. The cardinality of rule $\mathcal{X} \rightarrow s_i$ is given by the number of features in the antecedent, that is $|\mathcal{X}|$. The support of $\mathcal{X}$ is denoted as $\sigma(\mathcal{X})$, and is the number of instances in $\mathcal{D}_n$ having $\mathcal{X}$ as a subset. The confidence of rule $\mathcal{X} \rightarrow s_i$ is denoted as $\theta(\mathcal{X} \rightarrow s_i)$, and is the conditional probability of class s_i given the features in $\mathcal{X}$, that is, $\theta(\mathcal{X} \rightarrow s_i) = \frac{\sigma(\mathcal{X} \cup s_i)}{\sigma(\mathcal{X})}$.*

We denote as $\mathcal{R}(t_n)$ the classifier obtained at time step n, which is composed of a set of rules $\{\mathcal{X} \rightarrow s_i\}$ extracted from $\mathcal{D}_n$. Rules in $\mathcal{R}(t_n)$ are collectively used to score class in instance $t_n \in \mathcal{T}$. Basically, the classifier is interpreted as a poll, in which each

rule $\{\mathcal{X} \rightarrow s_i\} \in \mathcal{R}(t_n)$ is a vote given by $\mathcal{X}$ for class s_i. Given instance $t_n \in \mathcal{T}$, a rule $\mathcal{X} \rightarrow s_i$ is only considered as a valid vote if this rule is applicable to t_n.

Definition 2. *A rule $\{\mathcal{X} \rightarrow s_i\} \in \mathcal{R}(t_n)$ is said to be applicable to instance $t_n \in \mathcal{T}$ if $\mathcal{X} \subseteq t_n$. That is, if all features in $\mathcal{X}$ are present in t_n.*

We denote as $\mathcal{R}_a(t_n)$ the set of all rules in $\mathcal{R}(t_n)$ that are applicable to t_n. Thus, only and all the rules in $\mathcal{R}_a(t_n)$ are considered as valid votes when scoring class labels in instance t_n. Further, we denote as $\mathcal{R}_a^{s_i}(t_n)$ the subset of $\mathcal{R}(t_n)$ containing only rules predicting class s_i. Votes in $\mathcal{R}_a^{s_i}(t_n)$ have different weights, depending on the confidence of the corresponding rules. The weighted votes for class s_i are averaged, giving the score for class s_i with regard to instance t_n, as shown in Equation 3.1:

$$s(t_n, s_i) = \sum \frac{\theta(\mathcal{X} \rightarrow s_i)}{|\mathcal{R}_a^{s_i}(t_n)|} \tag{3.1}$$

Finally, the scores are normalized, as expressed by the scoring function $\hat{p}(s_i|t_n)$, shown in Equation 3.2. The scoring function estimates the likelihood of class s_i being the implicit the underlying features in instance t_n.

$$\hat{p}(s_i|t_n) = \frac{s(t_n, s_i)}{\sum_{j=0}^{k} s(t_n, s_j)} \tag{3.2}$$

3.2 Random Active Sampling

Žliobaitė et al. [2011] presents strategies to evaluate whether a streaming instance t_n must be labeled. Among those strategies we chose the simplest, Random Sampling. The Random strategy is naive in the sense that it labels the incoming instances at random instead of actively deciding which label would be more useful. The labeling decision does not depend on the actual incoming instance t_n. For any instance the true label is requested with a probability $\mathcal{B}$, where $\mathcal{B}$ is the given budget that represents how many resources are available to labeling. Algorithm 1 gives a formal description.

3.3 Utility Measures

Our approach to data stream classification is based on selecting the most appropriate instances to compose the training set at each time step. Our hypothesis is that

Algorithm 1 Random Active Sampling

Input: Labeling budget $\mathcal{B}$

Output: $labeling \in \{True, False\}$ indicates whether to request the true label y_n for t_n

1: **if** $z < \mathcal{B}$, $z \sim U(0,1)$ **then**
2: **return** $labeling \leftarrow True$
3: **else**
4: **return** $labeling \leftarrow False$

the training set must provide adaptiveness and memorability to the resulting classifier, and thus instances should be selected properly in order to provide both properties to the resulting classifier. This is challenging, however, because improving adaptiveness and memorability simultaneously may lead to a conflicting-objective problem. Instead, our selective sampling approaches form training sets that provide a proper balance between adaptiveness and memorability. Specifically, at each time step, candidate instances are placed into a n-dimensional space, in which each dimension corresponds to a utility measure which is either related to adaptiveness or memorability. Next we discuss the utility measures we investigate.

At each time step, the classifier must score classes in a particular target instance. Some of the utility measures we are going to discuss next are based on the distance to the target instance. By minimizing such distance we are essentially maximizing adaptiveness, since the selected instances are more likely to belong to the same distribution of the target instance (i.e., the selected instances are more similar to the target instance), as shown in [Žliobaitė, 2010a].

Distance in space The similarity between the target instance t_n and an arbitrary instance t_j is given by the number of rules in the classifier $\mathcal{R}^a(t_n)$ that are also applicable to t_j. Differently from traditional measures such as cosine and Jaccard, the rule-based similarity considers not only isolated terms, but also combination of terms. Thus, the utility of instance t_j is given as:

$$U_s(t_j) = \frac{|\{r \in \mathcal{R}^a(t_n) \text{ such that } r \text{ is applicable to } t_j\}|}{|\{\mathcal{R}^a(t_n)\}|} \tag{3.3}$$

Distance in time Let $\gamma(t_j)$ be a function that returns the time in which instance t_j arrived. The utility of instance t_j is given as:

$$U_t(t_j) = \frac{\gamma(t_j)}{\gamma(t_n)} \tag{3.4}$$

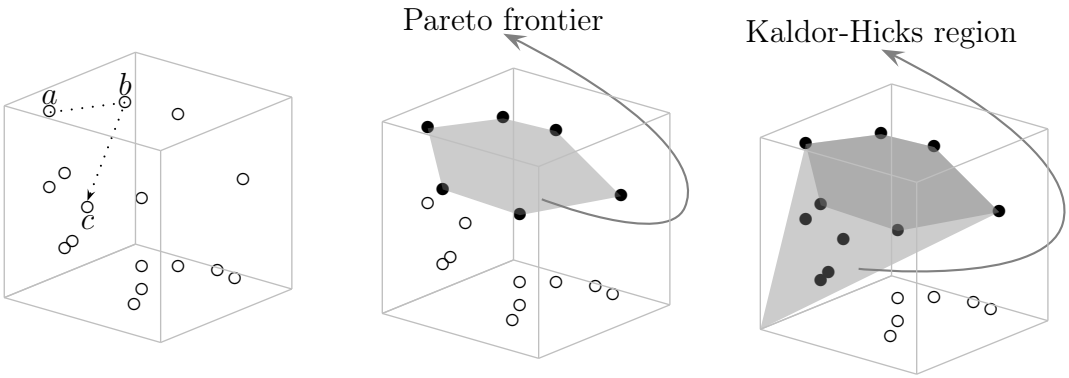


Figure 3.2. Illustrative example. (Left) The dominance operator: neither a or b dominates each other, but b dominates c. (Middle) Points lying in the Pareto frontier. (Right) Points inside the Kaldor-Hicks region.

As for memorability, we are going to discuss a utility measure based on randomly shuffling candidate instances:

Memorability In order to provide memorability, the training set must contain instance posted in different time periods. A simple way to force this is to generate a random permutation of the candidate instances, that is, randomly shuffling the candidate instances [Durstenfeld, 1964]. Let $\alpha(t_j)$ be a function that returns the position of instance t_j in the shuffle. The utility of instance t_j is given as:

$$U_r(t_j) = \frac{\alpha(t_j)}{|\mathcal{D}_n|} \tag{3.5}$$

3.4 Economic Efficiency

In economics, the term *economic efficiency* refers to the use of resources so as to maximize the production of goods and services to the society [Marshall, 2001]. An economic system is said to be more efficient than another if it can provide more goods and services for society without using more resources.

The economic efficiency criteria has been used in several fields beyond economics, such as marketing and optimization. One of the most known economic efficiency criteria used is Pareto Efficiency. This criteria follow the proper concept of Economic Efficiency.

The same intuition could be exploited for the sake of selecting instances to compose the training set at each time step. In this case, a training set is economically efficient if it is only possible to improve memorability at the cost of adaptiveness, and vice-versa.

There is an alternative, less stringent notion of efficiency, which is based on the principle of compensation [Chipman, 2008]. Under new arrangements in the society, some may be better off while others may be worse off. Compensation holds if those made better off under the new set of conditions could compensate those made worse off. Next we discuss algorithms that exploit these two notions of economic efficiency in order to select instances to compose the training sets.

Pareto Frontier

Instances that are candidate to compose the training set at time step $n$are placed in a 3-dimensional space, according to their utility instances, as shown in Figure 3.2. Thus, each instance a is a point in such utility space, and is given as $< U_s(a), U_t(a), U_r(a) >$.

Definition 3. *Instance a is said to dominate instance b iff both of the following conditions are hold:*

- $U_s(a) \geq U_s(b)$ *and* $U_t(a) \geq U_t(b)$ *and* $U_r(a) \geq U_r(b)$
- $U_s(a) > U_s(b)$ *or* $U_t(a) > U_t(b)$ *or* $U_r(a) > U_r(b)$

Therefore, the dominance operator relates two instances so that the result of the operation has two possibilities as shown in Figure 3.2 (Left): (i) one instance dominates another or (ii) the two instances do not dominate each other.

Definition 4. *Window $\mathcal{P}_n = \{d_1, d_2, \ldots, d_m\}$ is said to be Pareto-efficient at time step n, if $\mathcal{P}_n \subseteq \mathcal{D}_n$ and there is no pair of instances $(d_i, d_j) \in \mathcal{P}_n$ for which d_i dominates d_j.*

Instances that are not dominated by any other instances, lie on the Pareto frontier [Palda, 2011]. Therefore, by definition, the Pareto-efficient training set at time step n, $\mathcal{P}_n$, is composed by all the instances lying in the Pareto frontier that is built from $\mathcal{D}_n$. There are efficient algorithms for building and maintaining the Pareto frontier, and we employed the algorithm proposed in [Börzsönyi et al., 2001] which ensures $O(|\mathcal{D}_n|)$ complexity. We denote the process of exploiting Pareto-efficient training sets as Pareto-Efficient Selective Sampling, or simply PESS. Figure 3.2 (Middle) shows an illustrative example of a Pareto frontier built from arbitrary points in the utility space. Algorithm 2 shows the Pareto-Efficient Selective Sampling procedure.

Algorithm 2 Pareto-Efficient Selective Sampling

Input: Instances in $\mathcal{D}_n$

```
1: P_n ← ∅
2: for all instances d ∈ D_n do
3:     notDominated ← True
4:     for all instances p ∈ P_n do
5:         if d dominates p then
6:             Remove p from P_n
7:         else if p dominates d then
8:             insert p into P_n
9:             notDominated ← False
10:            break
11:    if notDominated then
12:        insert d into P_n
13: D_n ← P_n
14: return P_n
```

Kaldor-Hicks Region

The PESS strategy follows a stringent criterion, which tends to select only few instances to compose the training sets. As a result, the training sets may become excessively small and prone to noise. The Kaldor-Hicks criterion, on the other hand, follows a benefit-cost analysis and circumvents the small-training set problem by stating that efficiency is achieved if those that are made better off could in theory compensate those that are made worse off. Thus, under the Kaldor-Hicks criterion an utility measure can compensate other utility measures, and therefore, this criterion selects instances that are located inside a region which is below the Pareto frontier. To define this region we must first define the overall utility of a instance.

Definition 5. *Assuming that all measures are equally important, the overall utility of an arbitrary instance $d_i \in \mathcal{D}_n$ is:*

$$U(d_i) = U_s(d_i) + U_t(d_i) + U_r(d_i) \tag{3.6}$$

That is, the overall utility of a instance is given as the sum of its utility measures. Also, the baseline instance, which is denoted as d^*, is defined as:

$$d^* = \{d_i \in \mathcal{P}_n | \forall d_j \in \mathcal{P}_n : U(d_i) \leq U(d_j)\} \tag{3.7}$$

That is, the baseline is the instance lying in the frontier for which the overall utility assumes its lowest value.

The Kaldor-Hicks region is composed of instances for which the overall utility is not smaller than the baseline overall utility. Such baseline utility is the utility associated with the instance lying in the Pareto frontier for which the overall utility is the lowest.

Algorithm 3 Kaldor-Hicks-Efficient Selective Sampling

Input: Instances in $\mathcal{D}_n$ and instances in $\mathcal{P}_n$

1: $\mathcal{K}_n \leftarrow \emptyset$
2: $d^* \leftarrow \{d_i \in \mathcal{P}_n | \forall d_j \in \mathcal{P}_n : U(d_i) \leq U(d_j)\}$
3: **for all** instances $d \in \{\mathcal{D}_n - \mathcal{P}_n\}$ **do**
4: **if** $U(d) \geq U(d^*)$ **then**
5: insert d into $\mathcal{K}_n$
6: $\mathcal{D}_n \leftarrow \mathcal{K}_n$
7: **return** $\mathcal{K}_n$

Definition 6. *Window $\mathcal{K}_n = \{d_1, d_2, \ldots, d_m\}$ is said to be Kaldor-Hicks-efficient at time step n, if $\mathcal{P}_n \subseteq \mathcal{K}_n \subseteq \mathcal{D}_n$, and there is no instance $d_i \in \mathcal{K}_n$ such that $U(d^*) > U(d_i)$.*

We denote the process of exploiting Kaldor-Hicks-efficient training sets as Kaldor-Hicks-Efficient Selective Sampling, or simply KHSS. Figure 3.2 (Right) shows an illustrative example of a Kaldor-Hicks region built from arbitrary points in the utility space. Algorithm 3 shows the Kaldor-Hicks-Efficient Selective Sampling procedure.

Chapter 4

Data Stream Classification

In this Chapter we empirically evaluate our EESS algorithms. In Section 4.1 we describe the experimental evaluation: evaluation metrics, baselines and experiment setup. The remaining sections are case of study in different applications. In Section 4.2 we show the results on sentiment analysis datasets collected from Twitter. In Section 4.3 contains results of experiments performed in forest cover types dataset. In Section 4.4 we show the results in a email spam datasets. In Section 4.5 we show the results in a poker game dataset. Finally, in Section 4.6 we discuss about the results.

4.1 Experimental Evaluation

In this section we detail experimental evaluation. First, we show metrics used to evaluate our approach and compare it against stat-of-the-art data stream learning algorithm (Section 4.1.1). Then, we shortly discuss about each baseline (Section 4.1.2), followed by parameters setup for each algorithm (Section 4.1.3).

4.1.1 Evaluation Metrics

We select a set of metrics to evaluate the performance of our algorithm against state-of-the-art algorithms. The experiments were conducted to evaluate the performance in terms of prediction performance, computational resources and labeling efforts.

We used Mean Squared Error (MSE) to evaluate the prediction performance through class scoring. Equation 3.2 formalize MSE concept, where s_i is the correct class associated with instance $t_i \in \mathcal{T}$, and $\hat{p}(s_i|t_i)$ is the class score assigned by the classifier to instance $t_i \in \mathcal{T}$.

$$MSE = \frac{1}{|\mathcal{T}|} \sum_{\forall t_i \in \mathcal{T}} (1 - \hat{p}(s_i|t_i))^2 \tag{4.1}$$

In terms of computing resources we employ RAM-Hours measure [Bifet et al., 2010b], where every RAM-Hour is equals a GB of RAM deployed for 1 hour of execution. Also, we evaluate the amount of training resources used over time, as the number of instances labeled during the process.

Finally, to evaluate labeling efforts we ranged the labeling budget control and correlated it with the overall MSE.

4.1.2 Baselines

As baselines, we used Hoeffding Adaptive Trees [Bifet and Frank, 2010; Bifet et al., 2011] (abbreviated as HAT), Active Classifier [Žliobaitė et al., 2013, 2011] (abbreviated as AC), and Incremental Lazy Associative Classifier [Santana et al., 2011] (abbreviated as ILAC). Next we briefly describe the main characteristics of each of these algorithms.

Hoeffding Adaptive Trees is the mixture between Hoeffding Trees (a.k.a. Very Fast Decision Tree) and ADWIN algorithms. Hoeffding Trees is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams, assuming that the distribution generating examples does not change over time. Hoeffding trees exploit the fact that a small sample can often be enough to choose an optimal splitting attribute. This idea is supported mathematically by the Hoeffding bound, which quantifies the number of observations needed to estimate some statistics within a prescribed precision. The Hoeffding Adaptive Trees uses ADWIN to monitor the performance of branches on the tree and to replace them with new branches when their accuracy decreases if the new branches are more accurate. ADWIN is a change detector and estimator that keeps a variable-length window of recently seen items, with the property that the window has the maximal length statistically consistent with the hypothesis "there has been no change in the average value inside the window".

Active Classifier is a theoretically supported framework for active learning from drifting data streams and develops three active learning strategies for streaming data that explicitly handle concept drift. They are based on uncertainty, dynamic allocation of labeling efforts over time and randomization of the search space.

Incremental Lazy Associative Classifier is an incremental streaming algorithm, which produces classification models using association rules. The classification model is kept up-to-date in an incremental fashion, so that at any given time the model properly reflects the current concept. In order to track concept drift, ILAC projects training examples on a demand driven basis, according to the content of the instance being classified. Projecting the training data offers a series of advantages, including the ability to quickly detect trending information emerging in the stream.

4.1.3 Experiments Setup

All experiments were performed on a 1.93 GHz Core i7 machines with 8GB of memory, using the MOA system [Bifet et al., 2010a], an environment for running experiments with data streams.

Our evaluation follows the Test-Then-Train methodology, in which each individual instance in $\mathcal{T}$ is used to test the classifier and then it becomes available for training. In order to simulate a real situation of data stream analysis where, it is not possible to create large training sets, we provided a small training seed containing 1% of all data set for each classifier.

We ranged labeling budget parameter between 0.01 and 1 for our approaches and AC. To AC we set Random Variance Uncertainty [Žliobaitė et al., 2013, 2011] strategy to sampling from the stream. HAT algorithm we found that the default parameters achieved the best results and ILAC we setup the confidence and support thresholds as 0.01, and maximum size of rules as 3.

4.2 Sentiment Analysis

There is a growing trend in performing sentiment analysis using classification-related techniques, where the classification model is built from a labeled sentiment training set by learning the underlying characteristics that distinguish one sentiment from another (i.e., happiness, madness, surprise, suspicion). The success of these classifiers rests on their ability to judge attitude by means of textual-patterns present in the data, which usually appear in the form of (idiomatic) expressions and combinations of words.

Sentiment analysis have been used for different purposes such as, product [Turney, 2002], movie [Pang et al., 2002] and restaurant [Snyder and Barzilay, 2007] reviews. As result the information obtained from the analysis of the customers reviews is becoming

a strategy key on e-commerce systems, recommendation systems, content providers, and search engines.

Recently social networks have attracted attention to application of sentiment analysis given the ubiquitous reach, easy communication and collection of data. Often social media sites offers Application Programming Interfaces (API's) which enables easily crawl public data from them users. Also these sites are responsible for more than two-thirds of all Internet users[nie, 2009]. Besides, the content collected from these social media sites have been used to advertising on-the-fly by recognizing customer sentiment in real-time being a potential technology breakthrough[Hof, 2013].

A success case was demonstrated in 2013's National Football League's Superbowl (a premier sporting event in the USA) where a well known manufacturer of *Oreo* cookies took advantage of a third quarter blackout (and associated Twitter sentiment) to embed a contextual advertisement. Another example at the same event was the advertisement for a Hollywood movie, where, based on the initial advertisement which happened before the start of the first quarter (and associated Twitter sentiment), the decision on which of several possible advertisements to run later on in the program was apparently taken as a runtime decision.

In particular, Twitter has proven itself to be an authoritative source of breaking news, some of which concerning important topics and events of huge impact world wide [Jansen et al., 2009]. Lightweight and easy communication mechanisms within Twitter, such as microblogging, make users eager to express and share their opinions. As a result, sensitive information is created almost in the same time the event is happening in the real world, and it becomes available shortly after it is created.

However, sentiment analysis over Twitter real-time instances is particularly challenging, because:

1. As mentioned above, Twitter (and other social media channels) follows the data stream model[1] restricting memory and time of processing;

2. The training-set is potential noisy, since training instances may be (incorrectly) labeled using either the current classifier [Santana et al., 2011] or author- provider sentiment indicators (i.e., emoticons and hash tags [Barbosa et al., 2012]);

3. Either sentiment distribution or the characteristics related to certain sentiments may change over time in almost unforeseen ways (i.e., sentiment drifts);

[1] There are three main source streams in Twitter. The Firehose provides all status updates from everyone in real-time, which corresponds to more than 140 million tweets daily (on average). Spritzer and Gardenhose are two subsamples of the Firehose. The sampling rates are 5% and 15%, respectively. Also, specific subsamples may be created by inspecting specific events or topics.

Given the challenges of real-time sentiment analysis over Twitter we performed a systematic set of experiments using sentiment-rich Twitter data collected from three important events in 2010. We employed different sentiments expressed in different languages.

4.2.1 Sentiment Stream Analysis

In our context, the task of learning sentiment streams is defined as follows. At time step n, we have as input a training window referred to as $\mathcal{D}_n$, which consists of a set of records of the form $< d, s_i >$, where d is a instance (represented as a list of terms), and s_i is the sentiment implicit in d. Messages in $\mathcal{D}_n$ are uniquely identified and the sentiment variable s draws its values from a pare-defined and discrete set of possibilities (e.g., s_1, s_2, ..., s_k). The training window is used to build a classifier relating textual patterns in the instances to their corresponding sentiments. A sequence of future instances referred to as $\mathcal{T} = \{t_n, t_{n+1}, \ldots\}$, consists of instances for which only their terms are known, while the corresponding sentiments are unknown. The classifier obtained from $\mathcal{D}_n$ is used to score the sentiments for instance t_n in $\mathcal{T}$. Once processed, instance t_n is included into D_{n+1}, so that another classifier is built.

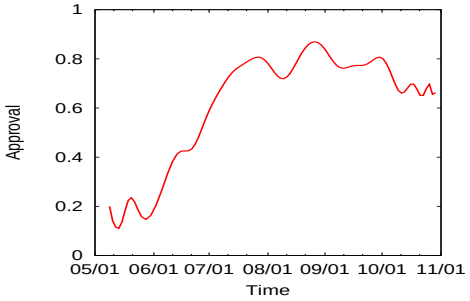
There are countless strategies for devising a classifier for sentiment analysis. The majority of these classification strategies, however, are not well-suited to deal with real-time data coming on streams. Some strategies [Breiman et al., 1984; Cortes and Vapnik, 1995] are specifically devised for offline classification, and this is problematic because producing classifiers on-the-fly would be unacceptably costly. Even updating the models in scenarios with high-speed streams would be excessively lengthy. In such hard circumstances, alternate classification strategies may become more convenient.

4.2.1.1 Brazilian Presidential Elections

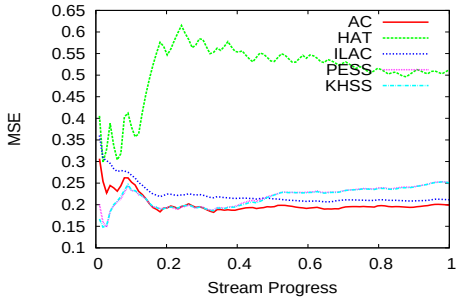
The presidential election campaigns were held from June to October 2010. The candidate Dilma Rousseff launched a Twitter page during a public announcement, and she used Twitter as one of the main sources of information for her voters. The campaign attracted more than 500,000 followers and as a result Dilma was the second most cited person on Twitter in 2010. The election came to a second round vote, and Dilma Rousseff won the runoff with 56% of the votes.

Dilma Rousseff Election Campaign We collected 66,643 messages in Portuguese referencing Dilma Rousseff in Twitter during her campaign. The dataset contains

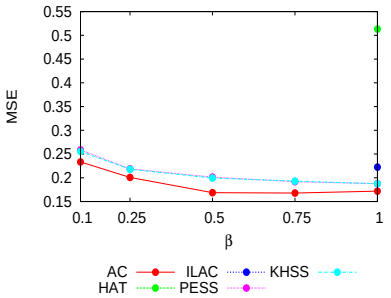
62,089 distinct terms, and a message is posted every 50 seconds, on average. We labeled these messages in order to track the population sentiment of approval during this period. As shown in Figure 4.1 (a), approval varied significantly over the time due to several polemic statements and political attacks, and our goal is to score approval during her campaign.



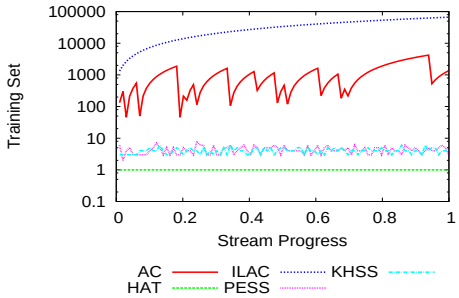
(a) Approval over Dilma Rousseff's campaign. Approval sentiment varied greatly from 05/2010 to 11/2010.



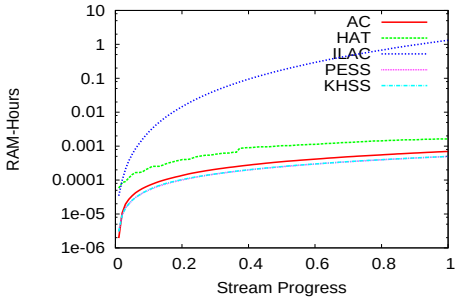
(b) MSE numbers as stream evolves.



(c) X-Y scatter plot correlating labeling budget and MSE.



(d) Size of the training sets as stream evolves.



(e) RAM-Hours as the stream evolves.

Figure 4.1. Brazilian Presidential Elections. Tweets are in Portuguese.

Figure 4.1 (b) shows the results in terms of MSE obtained for the evaluation of the classifiers in this dataset. The x-axis represents different time steps (i.e., each instance that passes in the stream), while the y-axis shows the MSE so far. As it can be seen, our proposed algorithms provide a comparable approximation in comparison

with the baselines as the stream evolves.Both, PESS and KHSS, started much better than the other competing algorithms, but after a point (around 0.3) the models deteriorate. Although PESS and KHSS increases the approximation errors, the total approximation error in the stream is close to AC and ILAC. Figure 4.1 (c) shows a scatter-plot correlating labeling efforts and area under error curve. PESS, KHSS and AC have labeling effort control through labeling budget parameter whereas ILAC as well as HAT requires that every streaming instance to be labeled in order to update their models. As can be seen MSE decreases for PESS, KHSS and AC as more efforts are spent in labeling. With a labeling budget of 0.25 PESS, KHSS and AC outperform ILAC. HAT has the worst performance even requiring that every instance being labeled. PESS and KHSS are comparable to AC but AC achieves lower MSE for all labeling budget values.

Figure 4.1 (d) shows how size of training varies as the stream evoltes. We assume that HAT requires only the target instance for updating its tree model, and thus we consider that the training set is composed only by the target instance. The AC algorithm requires much more instances within each training set. An abrupt decrease in the number of training messages is always observed after drifts. Although ILAC performs a data projection strategy that filters irrelevant instances at each time step, it is clear that the number of training instances still increases as the stream evolves. The proposed algorithms requires very small training sets, since the Pareto frontier at each time step is composed by few instances, but these instances are still able to make the classifier robust to drifts as the stream evolves. Further, despite being less stringent than PESS, the proposed KHSS algorithm also requires small training sets.

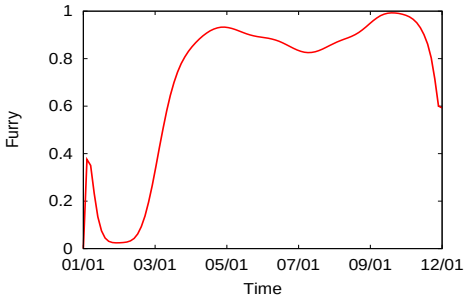
Figure 4.1 (e) shows RAM-Hours numbers for the algorithms. AC, as well as PESS and KHSS, are clearly the best performers in terms of amount of computing resources required. ILAC is the worst performer.

4.2.1.2 TIME's Person of the Year

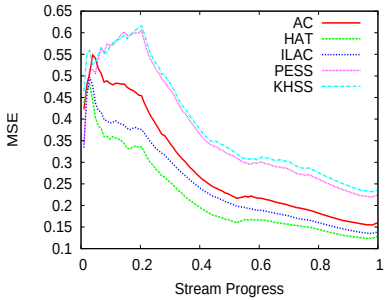
Every year, TIME magazine selects the person (or a group of persons) that has mostly influenced during the year. The chosen person for 2010 was Mark Zuckerberg. The reader choice, however, was Julian Assange, with an overwhelming superiority of votes.

Zuckerberg and Assange We collected 5,616 instances in English referencing Julian Assange and Mark Zuckerberg from 1-15-2010 to 12-21-2010. The dataset contains 7,294 distinct terms, and a message is posted every 45 seconds, on average. We labeled

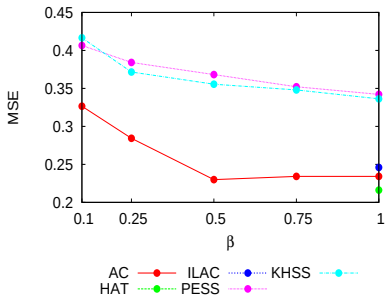
them in order to track diverse sentiments regarding the magazine's decision. Sentiments include (dis)approval, surprise (since the reader choice was pointing to Julian Assange), and even fury. As shown in Figure 4.2 (a), furry about Julian Assange varied significantly over the time. Furthermore, between 01-15-2010 and 04-01-2010 there were a mixture of feelings on the choice of Person of the Year, what causes sudden drifts. In this period adaptation is more required than memorability.



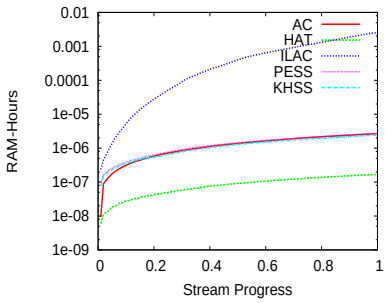
(a) Furry about Julian Assange varied from 01/2010 to 12/2012.



(b) MSE numbers as the stream evolves.



(c) X-Y scatter plot correlating the labeling budget and MSE.



(d) RAM-Hours as the stream evolves.

Figure 4.2. Person of the Year. Tweets are in English.

Figure 4.2 (b) shows the results in terms of MSE. As it can be seen, a better approximation is obtained by HAT and ILAC. For this dataset, AC achieved MSE numbers close to HAT and ILAC. At the end of the process, both PESS and KHSS algorithms reduced the approximation errors but, still worst than other algorithms.

Figure 4.2 (c) shows the trade-off between labeling budget and MSE. Again, MSE numbers decrease as more resources are available for labeling. AC achieves better MSE numbers when is available 0.5 of labeling budget, being extremely competitive against HAT and ILAC that require all the stream labeled.

Finally, Figure 4.2 (d) shows RAM-Hours numbers for the evaluated algorithms. The AC algorithm, as well as PESS and KHSS are, again, extremely competitive in terms of amount of computing resources required. ILAC is the worst performer.

4.2.1.3 FIFA World Cup

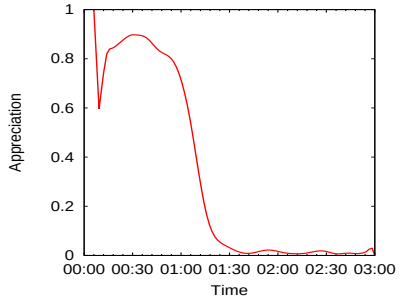
The 2010 Soccer World Cup involved 32 teams. The Brazilian team was defeated by the Dutch team on 07-02-2010, after a controversial match. The Brazilian team scored first, but soon after the Dutch team scored twice and won the match. A specific player, Felipe Melo, had decisive participation (for better and worse) in all three goals. Specifically, Figure 4.3 (a) shows how the appreciation for Felipe Melo expressed by Brazilian Twitter varied during the match.

The Brazilian Defeat We collected 1,020 messages referencing Felipe Melo. We randomly selected 4,646 of these messages, and we annotated them in order to track the sentiment of appreciation for the participation of Felipe Melo. This resulted in two datasets, the first one containing 3,214 annotated messages in Portuguese (8,101 distinct terms), and the second one containing 1,432 annotated messages in English (4,962 distinct terms). For these datasets, messages in the stream come in at a rate of 1.1 messages/sec.

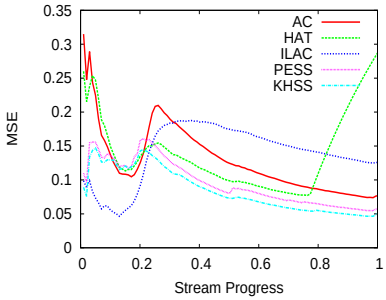
We start by analyzing the dataset in Portuguese. Figure 4.3 (b) shows the results in terms of MSE. We assume that these messages expose the sentiment expressed by Brazilian Twitter users during the match. As it can be seen, all algorithms were impacted by sudden drifts (as shown in Figure 4.3 (a)) that occurs during the match. ILAC achieved better MSE numbers in beginning, however, it delays to adapt after the concept drift and achieved worst MSE numbers. AC and HAT algorithms starts with worst MSE numbers and adapt their models reducing approximation errors. In the last moments of the stream, when Felipe Melo was excluded from the match and causes another drift, HAT had a sudden worsening in the MSE numbers. On the other hand, PESS and KHSS had shown extremely competitive with ILAC in the first moments, keeping a better approximation after the drift and reducing the MSE numbers. For this dataset, memorability is not mandatory (as the data distribution never returns to a pre-drift distribution), and thus PESS and KHSS were not able to provide significant improvements, although being the best performers overall.

The trade-off between labeling budget and MSE is shown in Figure 4.3 (c). PESS and KHSS achieved better MSE numbers than other algorithms with labeling budget equals to 0.1 and 0.25, being outperformed by AC after 0.5.

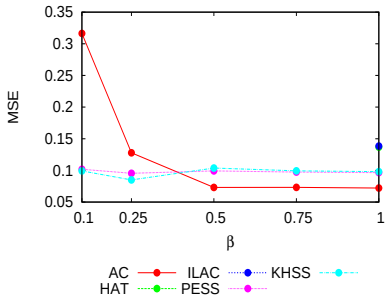
Figure 4.3 (d) shows the number of instances composing the training set at each time step. As in previous cases, AC and ILAC require much more training resources than other competing algorithms. PESS as well as KHSS require much less training instances, again, showing that the selective sampling strategy is effective in producing



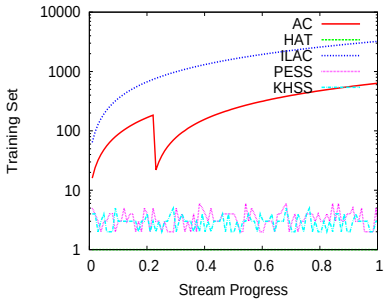
(a) Appreciation associated with Felipe Melo over the match.



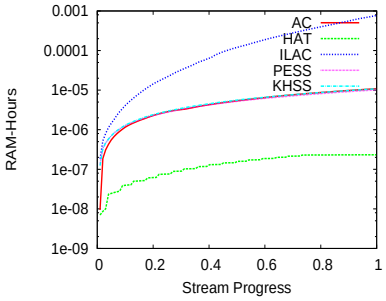
(b) MSE numbers as the stream evolves.



(c) X-Y scatter plot correlating labeling budget and MSE numbers.



(d) Size of the training sets as the stream evolves.

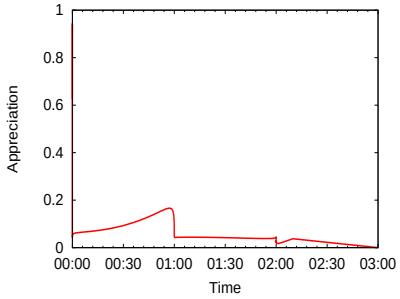


(e) RAM-Hours as the stream evolves.

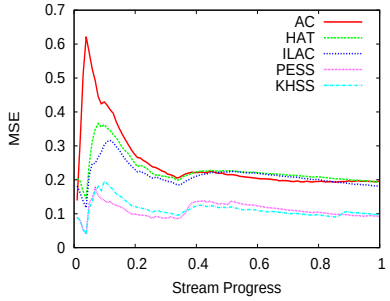
Figure 4.3. The Brazilian Defeat. Tweets are in Portuguese.

small and effective sets at each time step.

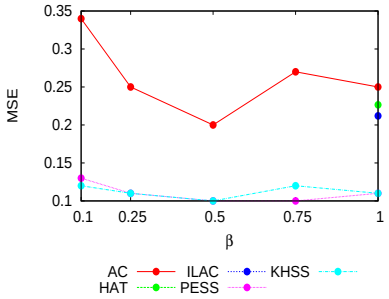
Finally, Figure 4.3 (f) shows RAM-Hours numbers. In this case, AC, as well as PESS and KHSS, are clearly the best performers in terms of amount of computing resources required. ILAC is the worst performer.



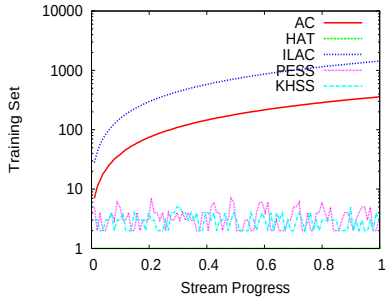
(a) Appreciation associated with Felipe Melo over the match.



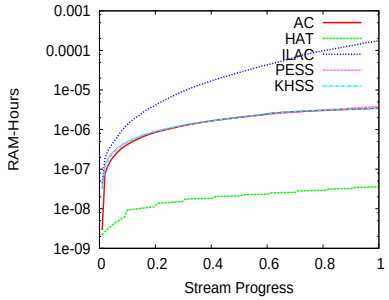
(b) MSE numbers as the stream evolves.



(c) X-Y scatter plot correlating labeling budget and MSE numbers.



(d) Size of the training sets as the stream evolves.



(e) RAM-Hours as the stream evolves.

Figure 4.4. The Brazilian Defeat. Tweets are in English.

The last set of experiments of sentiment data streams concerns the evaluation of the same event, but using the dataset composed of messages in English. Figure 4.4 (a) shows the appreciation regarding Felipe Melo. By inspecting the dataset, we confirmed that most of the messages were posted by people in Europe, specially German, English and Dutch users. Such users showed conflicting or mixed sentiments as the match

goes. As a consequence, drifts occur, but eventually the sentiment distribution returns to a pre-drift distribution, making memorability an important property while scoring sentiments in this dataset.

Figure 4.4 (b) shows MSE numbers as the stream evolves for each classifier. As can be seen, AC is the worst performer at beginning, but achieves a performance close to HAT and ILAC. PESS and KHSS had the best approximation.

The trade-off between labeling budget and MSE is shown in Figure 4.4 (c). PESS and KHSS are clearly better other algorithms. AC achieve its better result with labeling budget equals to 0.5, outperforming ILAC and HAT.

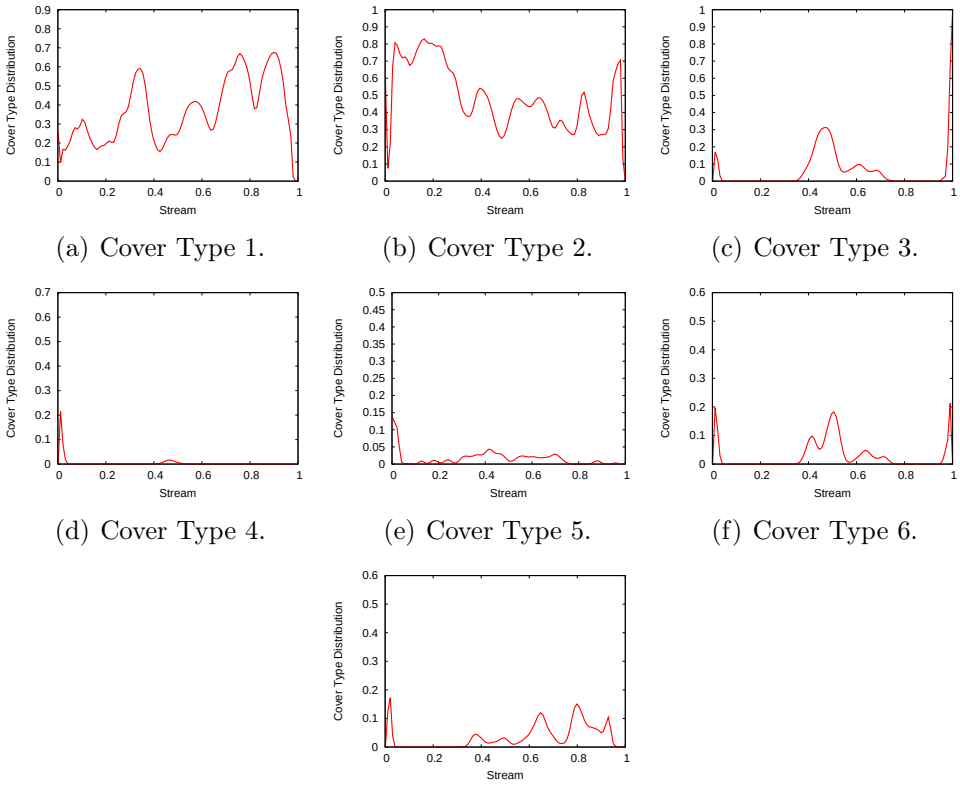
Figure 4.4 (d) shows the number of instances composing the training set at each time step. AC and ILAC require much more training instances than the other algorithms. PESS as well as KHSS requires much less training instances.

The computational resources, measured by RAM-Hours metric, required by each algorithm can be seen in Figure 4.4 (e). HAT is the best performer in terms of RAM-Hours. AC, as well as PESS and KHSS achieved close numbers in this data set. ILAC is the worst performer.

4.3 Forest Cover Type

In this section we evaluate our approaches in the *Forest Cover Type* dataset from the *UCI KDD* archive. The goal is to predict the forest cover type from cartographic variables. The problem is defined by 54 variables of different types: continuous and categorical. The dataset contains 581.102 examples. We generated a stream from the temporal sort of the original dataset. This dataset contains 7 classes and Figure 4.5 shows the distribution of each cover type over the stream. As can be seen, this dataset contains sudden and recurrent drifts being a scenario where memorability can be essential. For this dataset we used AC and HAT as baselines.

Figure 4.6 (a) shows MSE numbers over the stream. PESS as well as KHSS starts with the worst MSE numbers, however, after 20% of the stream has passed, both ones reduce the approximation errors being very competitive to HAT. After that point, memorability becomes an important requirement and our approaches shown able to balance adaptation to the sudden drifts and recurrent concepts that appears as the stream evolves. As can be seen, HAT and AC have increasing MSE numbers while PESS as well as KHSS keeps the error rates constants. Although, HAT achieves better approximation, it requires the true label for every instance after the prediction (labeling budget equals to 1.0), while, our algorithms require only 20% to provide an



(a) Cover Type 1. (b) Cover Type 2. (c) Cover Type 3.

(d) Cover Type 4. (e) Cover Type 5. (f) Cover Type 6.

(g) Cover Type 7.

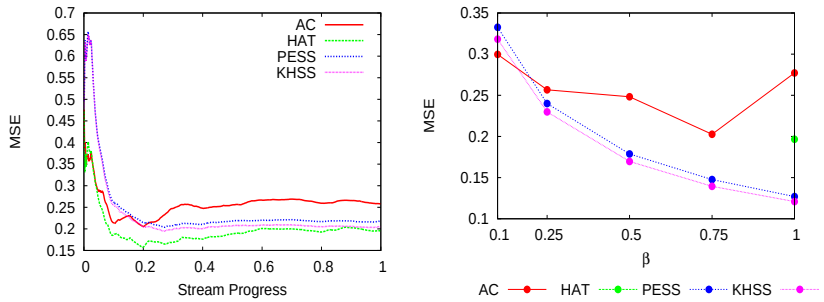
Figure 4.5. Cover type distribution over the stream.

approximation competitive with HAT. When PESS and KHSS operates with labeling budget of 1.0 achieves better results than HAT, as can be seen in Figure 4.6 (b). For an smaller labeling budget equals to 0.1 AC is better than PESS and KHSS, however, as more labeling budget is provided, our approaches outperform AC and HAT.

We also measure the amount of resources required for each algorithm. Figure 4.6 (c) shows the amount of training instances required for each algorithm. AC requires much more examples than PESS and KHSS. This result also shows that our selective sampling approaches are able to keep small training sets that provides enough information to a classifier predict over the stream. However, Figure 4.6 (d) shows that PESS as well as KHSS requires more RAM-Hours resources than others algorithms. Under this aspect AC performed better.

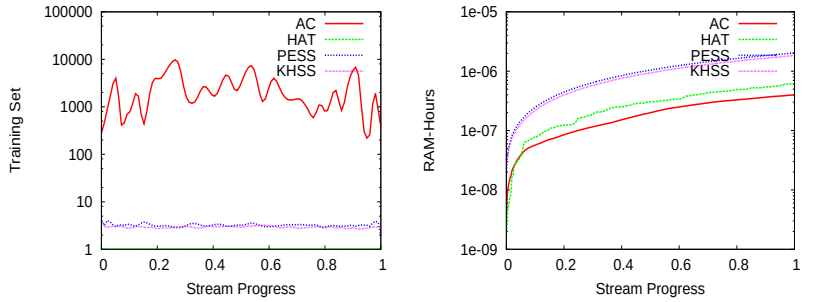
4.4 Spam Filtering Dataset

In this dataset we investigate the performance of our approaches in the spam filtering scenario. This dataset was used by Katakis et al. [2010] which proposed a



(a) MSE numbers as the stream evolves.

(b) X-Y scatter plot correlating labeling budget and MSE numbers.



(c) Size of the training sets as the stream evolves.

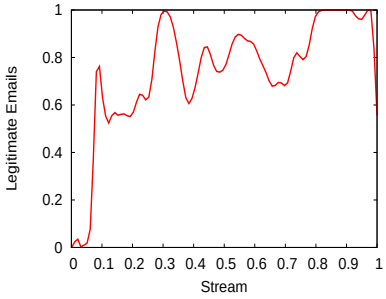
(d) RAM-Hours as the stream evolves.

Figure 4.6. Forest Cover Type

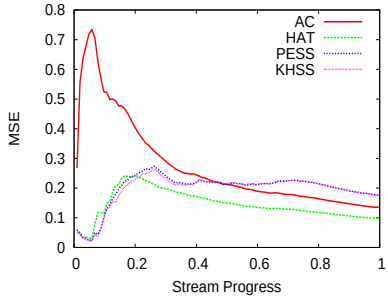
general framework for classifying data streams by exploiting stream clustering in order to dynamically build and update an ensemble of incremental classifiers. This dataset uses English email messages from the Spam Assassin Collection[2]. The task of this problem is discriminate between spam and legitimate messages. The spam ratio of the Spam Assassin collection is approximately 20%. The collection was sorted by the date and time that the mail was sent. The boolean bag-of-words approach was used for representing emails. This dataset consists of 9,324 instances, 500 attributes (words derived after applying feature selection with the χ^2 measure). This dataset contains gradual concept drift as mentioned by authors in [Katakis et al., 2010]. Figure 4.7 (a) shows the distribution of legitimate emails over the stream. As can be seen, the proportion of legitimate emails gradually grows, alternating with some local sudden and recurrent drifts.

Figure 4.7 (b) shows the MSE numbers over the stream. PESS as well as KHSS are very competitive against HAT algorithm in the first 20% of the stream, while AC achieves higher MSE numbers, being the worst one. However, in the rest of the stream AC reduces the approximation errors achieving competitive values in relation

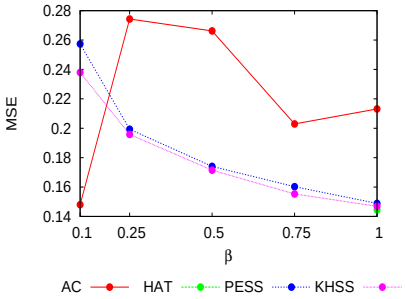
[2]The Apache SpamAssasin Project - http://spamassassin.apache.org/



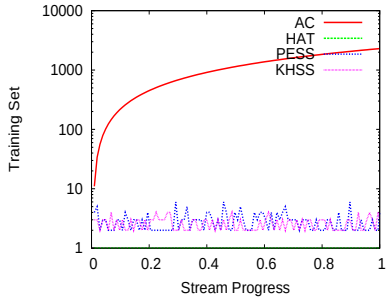
(a) Distribution of legitimate emails as the stream evolves.



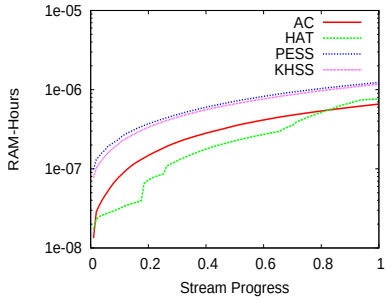
(b) MSE numbers as the stream evolves.



(c) X-Y scatter plot correlating labeling budget and MSE numbers.



(d) Size of the training sets as the stream evolves.



(e) RAM-Hours as the stream evolves.

Figure 4.7. Spam Filtering Dataset. Email messages are in English.

to HAT. PESS and KHSS slightly vary the approximation errors. Figure 4.7 (c) show correlation between labeling effort and MSE numbers. As can be seen HAT achieves better results than other algorithms, howver, AC achieves comparable results using less labeling efforts. PESS and KHSS decreases MSE numbers as more labeling budget is provided. In other hand, AC is irregular with different values of labeling budget. PESS as well as KHSS outperform AC for labeling budget greater than 0.25.

Figure 4.7 (d) shows the training set as the stream evolves. As in previous cases, PESS as well as KHSS keep small training sets and AC requires much more examples. In terms of RAM-Hours, AC and HAT are the best performers, as shown in Figure 4.7

(e).

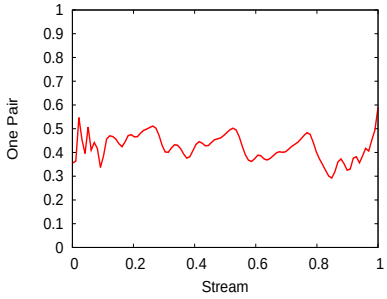
4.5 Poker Hand Prediction

In this dataset we experimental evaluate our algorithms in the poker hand prediction task. The *Poker Hand* dataset is from *UCI KDD* repository and consists of 1,000,000 instances and 11 attributes. Each record of the Poker-Hand dataset is an example of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank), for a total of 10 predictive attributes. There is one class attribute that describes the 11 poker hands. Figure 4.8 (a) shows the distribution of the *One Pair* hand as the stream evolves.

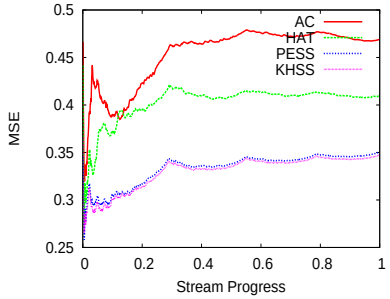
This stream contains a natural unbalance between the classes once some poker hands are more likely to appear than others. During the stream this feature causes gradual and recurrent drifts. That makes the balance between memorability and adaptation an important requirement. Figure 4.8 (b) shows the MSE number over the stream. The beginning of the stream (first 20%) has only two predominant classes well balanced. HAT and PESS as well as KHSS have a slightly approximation error decreases in contrast with AC that commit more errors. After this point, a scenario with gradual and recurrent drifts appears. At this point the algorithms must be prepared to adapts to new concepts and remember previous ones. Under this requirement our approaches achieves the lower MSE numbers, being the best performers. AC achieves the higher approximation errors.

Figure 4.8 (c) shows the correlation between labeling budget β and MSE numbers for PESS and KHSS. As in previous cases, the performance, in terms of MSE, improves as more labeling budget is available. For all labeling budget PESS and KHSS outperform AC as well as HAT. AC achieves better results at 0.5 of labeling budget, outperforming HAT.

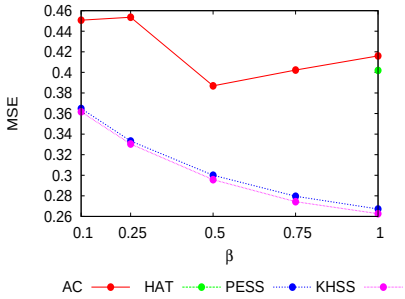
Figure 4.8 (d) shows that PESS as well as KHSS are able to keep small training sets that provide good generalization through the balance between adaptation and memorability obtained from our multi-objective model. In terms of RAM-Hours, our approaches require much more computational resources than AC and HAT, as can be seen in Figure 4.8 (e).



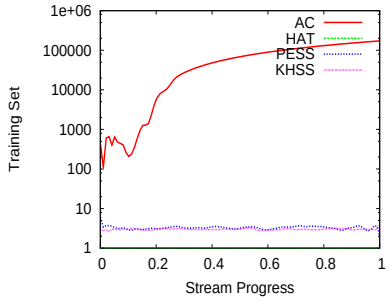
(a) Distribution of one pair hand over the stream.



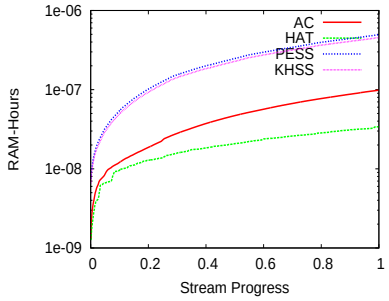
(b) MSE numbers as the stream evolves.



(c) X-Y scatter plot correlating labeling budget and MSE numbers.



(d) Size of the training sets as the stream evolves.



(e) RAM-Hours as the stream evolves.

Figure 4.8. Poker hand prediction.

4.6 Discussions

In this chapter we presented an extensive experimental evaluation of our algorithms. We deal with different problems modeled as data streams, such as sentiment analysis, forest cover type prediction, spam filtering, and poker hand prediction. Each of those datasets contains different types of concept drifts. Table 4.1 summaries types of concept drift found in each dataset. As can be seen, all datasets are combination of two or more concept drift types, what requires capabilities hypothesized in this work: Adaptation and Memorability.

When sudden concept drifts raise from the stream, the algorithm needs to quickly

Table 4.1. Concept Drift patterns for each dataset (X - Concept Drift type present).

	Concept Drift Type			
Dataset	Sudden	Incremental	Gradual	Recurrent
Presidential Elections	-	X	X	-
Person of the Year	-	X	X	-
FIFA World Cup - EN	X	-	-	-
FIFA World Cup - PT	X	-	-	-
Cover Type	X	-	X	X
Spam Filtering	X	-	X	X
Poker Hand	-	-	X	X

adapts to new concept, in other hand, when incremental or gradual concept drifts happen the algorithm have to change the concept learning in the same rate of the stream. Datasets highlighted in Table 4.1 are the ones where our algorithms outperformed the baselines. This results support our hypothesis that learning algorithms need Adaptation and Memorability capabilities. Furthermore, our algorithms were able to provide a proper balance between both capabilities to the classifier in different situations where mixtures between drift types are present.

The evaluation of correlation between labeling budget β and MSE numbers shown that, in general, as higher labeling budget better is class approximation. In terms of computing resources, PESS as well as KHSS were able to produce small training sets, capable to provide the proper balance between adaptation and memorability. However, even with small training sets, our approaches still requires more computational resources than HAT and AC in most cases.

By analyzing above results, our approaches had shown to be competitive in scenarios with recurrent and gradual drifts as well as with sudden drifts without recurrent concepts (Brazilian Defeat Portuguese). Further, about economic criteria, both criteria achieved very similar results and there is not significant difference between them.

Chapter 5

Conclusions

In this work we addressed the problem of learning on evolving data streams. We defined the main characteristics of data stream and discussed different types of changes that occur in streaming data. We focus on concept drifts that are a non-random class for changes on data. We presented some methodologies for learning on data streams, such as training windows, adaptive ensembles of classifiers, contextual classifiers, etc.

Our analysis lead to the development of a new algorithm called Economic-Efficient Selective Sampling (EESS), which model the requirements to learn on data streams in two capabilities: Adaptiveness and Memorability. Our algorithm select examples that optimize both capabilities applying the concept economic efficiency. We used this concept to simultaneously improving adaptiveness and memorability.

EESS may be instantiated using any efficiency criteria. In this work we employ Pareto criteria, which we called as PESS, and Kaldor-Hicks criteria, which we called as KHSS. In order to reduce the amount of labeling required we employed Random Sampling strategy labeling budget control.

We implemented EESS in MOA system and extensively evaluated PESS and KHSS in several scenarios and applications against three representatives of state-of-the-arts: Hoeffding Adaptive Trees, Active Classifier and ILAC. PESS as well as KHSS performed with similar results and shown to be robust to recurrent and gradual drifts. Also, in scenarios with sudden drift followed by a constant concept PESS as well as KHSS outperformed the baselines. Our strategies shown be very competitive against the baselines, outperforming or achieving similar results in almost all experiments in terms of class approximation.

Although this work achieved good results, it has some limitations. One of then is the fact that EESS was unable to achieve good result in Time's Person of the Year dataset. The achieved results was due heavily unbalance among the classes and se-

quential concept drifts. In this situation EESS strategy becomes incapable to provide a good balance between adaptiveness and memorability. In order to make EESS strategy robust in this scenario we intent to explore other dimensions to be inserted in the Utility Space. As the scenario in this dataset contains recurrent drifts we have to build a measure to emphasis memorability. Contextual information may be a good dimension for this issue once such information is robust to drifts as shown by Guerra et al. [2011].

Further add explore other source of information to create new dimensions, explore other learning algorithms may help to face scenarios as above commented. EESS is a wrapper for learning algorithms, so, we can plug any classifier to it and explore the features of both: the classifier and EESS. In addition to this, other learning methodologies, such as semi-supervised learning and reinforce learning may be also applied inside EESS.

EESS has a great potential as shown in this work. As future works we intent, in addition to create new dimensions and use other learning algorithms and methodologies, apply our algorithms in other data stream applications, such as advertising, recommendation and summarization.

Bibliography

(2009). Nielsen online report. social networks & blogs now 4th most popular online activity.

Alpaydin, E. (2004). *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Barbosa, G., Silva, I. S., Zaki, M., Jr., W. M., Prates, R., and Veloso, A. (2012). Characterizing the effectiveness of twitter hashtags to detect and track online population sentiment. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 2621–2626, New York, NY, USA. ACM.

Bifet, A. (2009). Adaptive learning and mining for data streams and frequent patterns. *ACM SIGKDD Explorations Newsletter*, 11(1):55–56.

Bifet, A. and Frank, E. (2010). Sentiment knowledge discovery in twitter streaming data. In *Proceedings of the 13th International Conference on Discovery Science*, DS'10, pages 1–15, Berlin, Heidelberg. Springer-Verlag.

Bifet, A., Frank, E., Holmes, G., and Pfahringer, B. (2012). Ensembles of restricted hoeffding trees. *ACM Trans. Intell. Syst. Technol.*, 3(2):30:1--30:20.

Bifet, A. and Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *In SIAM International Conference on Data Mining*.

Bifet, A. and Gavaldà, R. (2009). Adaptive learning from evolving data streams. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, IDA '09, pages 249--260, Berlin, Heidelberg. Springer-Verlag.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010a). Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604.

Bifet, A., Holmes, G., Pfahringer, B., and Frank, E. (2010b). Fast perceptron decision tree learning from evolving data streams. In *Proceedings of the 14th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part II*, PAKDD'10, pages 299--310, Berlin, Heidelberg. Springer-Verlag.

Bifet, A., Holmes, G., Pfahringer, B., and Gavaldà, R. (2011). Detecting sentiment change in twitter streaming data. *The Journal of Machine Learning Research*, 17:5–11.

Börzsönyi, S., Kossmann, D., and Stocker, K. (2001). The skyline operator. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 421–430.

Bramer, M. (2007). *Principles of Data Mining*. Springer.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and Regression Trees.

Chipman, J. (2008). Compensation principle. In Durlauf, S. N. and Blume, L. E., editors, *The New Palgrave Dictionary of Economics*. Palgrave Macmillan.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273--297.

Durstenfeld, R. (1964). Algorithm 235: Random permutation. *Communications of the ACM*, 7(7):420.

Feng, L., Chen, F., and Yao, Y. (2013). A concept similarity based data stream classification model. *Journal of Information & Computational Science*, 10(4):949–957.

Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining data streams: A review. *ACM SIGMOD Record*, 34(2):18–26.

Gama, J., , Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46.

Gama, J., ao, R. S., and Rodrigues, P. (2009). Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 329--338, New York, NY, USA. ACM.

Gomide, J., Veloso, A., Jr., W. M., Almeida, V., Benevenuto, F., Ferraz, F., and Teixeira, M. (2011). Dengue surveillance based on a computational model of spatio-temporal locality of twitter. In *ACM Web Science Conference (WebSci)*.

Guerra, P. H. C., Veloso, A., Meira, Jr, W., and Almeida, V. (2011). From bias to opinion: a transfer-learning approach to real-time sentiment analysis. In *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Hicks, J. (1939). The foundations of welfare economics. *The Economic Journal*, 49(196):696–712.

Hof, R. (2013). Real-time advertising has arrived, thanks to oreo and the super bowl. `www.forbes.com/`.

Jansen, B., Zhang, M., Sobel, K., and Chowdury, A. (2009). Twitter power: Tweets as electronic word of mouth. *Journal of the American Society for Information Science and Technology*, 60(11):2169–2188.

Kaldor, N. (1939). Welfare propositions in economics and interpersonal comparisons of utility. *The Economic Journal*, 49(195):549–552.

Katakis, I., Tsoumakas, G., and Vlahavas, I. (2010). Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, 22:371–391.

Kelly, M. G., Hand, D. J., and Adams, N. M. (1999). The impact of changing populations on classifier performance. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 367–371.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3).

Koychev, I. (2000). Gradual forgetting for adaptation to concept drift. In *ECAI 2000 Workshop on Current Issues in Spatio-Temporal Reasoning, Berlin, Germany*, pages 101–106.

Lourenco Jr., R., Veloso, A., Pereira, A., Meira Jr., W., Ferreira, R., and Parthasarathy, S. (2014). Economically-efficient sentiment stream analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 637–646, New York, NY, USA. ACM.

Marshall, A. (2001). *Principles of Economics*. MIT Press, Cambridge, MA, USA.

Masud, M., Gao, J., Khan, L., Han, J., and Thuraisingham, B. (2008). A practical approach to classify evolving data streams: Training with limited amount of labeled data. In *ICDM '08. Eighth IEEE International Conference on Data Mining, 2008*, pages 929–934.

Núñez, M., Fidalgo, R., and Morales, R. (2007). Learning in environments with unknown dynamics: Towards more robust concept learners. *The Journal of Machine Learning Research*, 8.

Palda, F. (2011). *Pareto's Republic and the new Science of Peace*. Cooper-Wolfling.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86.

Paul, M. J. and Dredze, M. (2011). You are what you tweet: Analyzing twitter for public health. In *ICWSM*.

Reinsel, D. and Gantz, J. (2012). The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east". International Data Corporation, sponsored by EMC Corporation.

Ribeiro, Jr., S. S., Davis, Jr., C. A., Oliveira, D. R. R., Meira, Jr., W., Gonçalves, T. S., and Pappa, G. L. (2012). Traffic observatory: A system to detect and locate traffic events and conditions using twitter. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, pages 5–11.

Santana, I., Gomide, J., Veloso, A., Jr., W. M., and Ferreira, R. (2011). Effective sentiment stream analysis with self-augmenting training and demand-driven projection. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 475–484. ACM.

Snyder, B. and Barzilay, R. (2007). Multiple aspect ranking using the good grief algorithm. In *In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 300–307.

Torres, D., Ruiz, J., and Sarabia, Y. (2011). Classification model for data streams based on similarity. In *Modern Approaches in Applied Intelligence*, volume 6703 of *Lecture Notes in Computer Science*, pages 1–9. Springer Berlin Heidelberg.

Tumasjan, A., Sprenger, T., Sandner, P., and Welpe, I. (2010). Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 178–185.

Turney, P. D. (2002). Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. cs.LG/0212032.

Veloso, A. and Dafe, J. (2012). UFMG respository of machine learning. `https://code.google.com/p/machine-learning-dcc-ufmg/`.

Žliobaitė, I. (2010a). *Adaptive Training Set Formation*. PhD thesis, Vilnius University.

Žliobaitė, I. (2010b). Learning under concept drift: an overview. *CoRR*, abs/1010.4784.

Žliobaitė, I. (2011). Combining similarity in time and space for training set formation under concept drift. *Intelligent Data Analysis*, 15(4):589–611.

Žliobaitė, I., Bifet, A., Gaber, M., Gabrys, B., Gama, J., Minku, L., and Musial, K. (2012). Next challenges for adaptive learning systems. *SIGKDD Explorations Newsletter*, 14(1):48–55.

Žliobaitė, I., Bifet, A., Holmes, G., and Pfahringer, B. (2010). MOA concept drift active learning strategies for streaming data. *The Journal of Machine Learning Research*, 11:1601--1604.

Žliobaitė, I., Bifet, A., Pfahringer, B., and Holmes, G. (2011). Active learning with evolving streaming data. In *Machine Learning and Knowledge Discovery in Databases*, volume 6913, pages 597–612.

Žliobaitė, I., Bifet, A., Pfahringer, B., and Holmes, G. (2013). Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–1.

Zhu, X., Zhang, P., Lin, X., and Shi, Y. (2010). Active learning from stream data using optimal weight classifier ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 40(6):1607–1621.