On Utilizing Weak Estimators to Achieve the Online Classification of Data Streams*

Hanane Tavasoli†, B. John Oommen‡ and Anis Yazidi§

Abstract

Classification, typically, deals with unique and distinct training and testing phases. This paper pioneers the concept when these phases are not so clearly well-defined. More specifically, we consider the case where the testing patterns can subsequently be considered as training patterns. The paradigm is further complicated because we assume that the class-conditional distributions of the features/classes are non-stationary, as in the case of most real-world applications. Specifically, we consider the model where the training phase is non-stationary and that it is, further, *interleaved with the testing, and where it can be done online and in a real-time manner.*

We propose a novel online classifier for complex data streams which are generated from non-stationary stochastic properties. Instead of using a single training model with "counters" that maintain important data statistics, our online classifier scheme provides a real-time self-adjusting learning model. The learning model utilizes the multiplication-based update algorithm of the Stochastic Learning Weak Estimator (SLWE) at each time instant as a new labeled instance arrives. In this way, the data statistics are updated every time a new element is seen, without requiring that we have to rebuild the model when changes occur in the data distributions. Finally, and most importantly, the model operates with the understanding that the correct classes of previously-classified patterns become available at a later juncture subsequent to some time instances. This forces us to update the training set, the training model and *the class conditional distributions* as the testing proceeds.

The results from rigorous empirical analysis on two-dimensional/multi-dimensional and binomial/multinomial distributions are remarkable. We also report some results on two real-life datasets adapted to this model of computation, demonstrating the advantages of the novel scheme for both binomial and multinomial non-stationary distributions.

Keywords : *Weak Estimators, Learning Automata, Non-Stationary Environments, Classification in Data Streams*

*A preliminary and very brief version of this paper, which makes the claim on the results, was presented at *IEA/AIE16, the 2016 International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, in Morioka, Japan, in August 2016. We are very grateful for the feedback from the anonymous Referees of the *original* submission. Their input significantly improved the quality of this final version.

†This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. E-mail: h.tavasoli@gmail.com.

‡Author's status: *Chancellor's Professor*, *Fellow of the IEEE*, and *Fellow of the IAPR*. This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. The author is also an Adjunct Professor with Agder University College in Grimstad, Norway. E-mail: oommen@scs.carleton.ca.

§Author's status: *Professor*. This author can be contacted at: Oslo Metropolitan University, Department of Computer Science, Pilestredet 35, Oslo, Norway. E-mail: anisy@oslomet.no.

DOI: https://dx.doi.org/10.1016/j.engappai.2019.08.015

1 Introduction

In the past few years, due to the advances in computer hardware technology, large amounts of data have been generated and collected and are stored permanently from different sources. Some the applications that generate data streams are financial tickers, log records or click-streams in web tracking and personalization, data feeds from sensor applications, and call detail records in telecommunications. Analyzing these huge amounts of data has been one of the most important challenges in the field of Machine Learning (ML)[1] and Pattern Recognition (PR).

Traditionally, ML and PR methods (as explained in [11, 22]) are assumed to deal with static data stored in memory, which can be read several times. On the contrary, streaming data grows at an unlimited rate and arrives continuously in a single-pass manner that can be read only once. Further, there are space and time restrictions in analyzing streaming data. Consequently, one needs methods that are "automatically adapted" to update the training models based on the information gathered over the past observations whenever a change in the data is detected.

Mining streaming data is constrained by limited resources of time and memory. Since the source of data generates a potentially unlimited amount of information, loading all the generated items into the memory and achieving offline mining is no longer possible. Besides, in Non-Stationary Environments (NSEs), the source of data may change over time, which leads to variations in the underlying data distributions. Thus, with respect to this dynamic nature of the data, the previous data model discovered from the past data items may become irrelevant or even have a negative impact on the modeling of the new data streams that become available to the system.

A vast body of research has been performed on the mining of data streams to develop techniques for computing fundamental functions with limited time and memory, and it has usually involved the sliding-window approaches or incremental methods. In most cases, these approaches require some *a priori* assumption about the data distribution or need to invoke hypothesis testing strategies to detect the changes in the properties of the data.

1.1 Salient Aspects of our Paradigm

In this article, we will study classification problems in NSEs, where sequential patterns are arriving and being processed in the form of a data stream that was potentially generated from different sources with different statistical distributions. The classification of the data streams is closely related to the estimation of the parameters of the time varying distribution, and the associated algorithms must be able to detect the source changes and to estimate the new parameters whenever a switch occurs in the incoming data stream.

Pioneering aspects of this Paradigm: Apart from the "traditional" classification problem involving unique and distinct training and testing phases, this paper pioneers the concept when these phases are not so clearly well-defined. Rather, we consider the fascinating phenomenon in which the testing patterns can subsequently be considered as training patterns, once their true class identities are known. Thus, the model operates with the understanding that the correct classes of previously-classified patterns become available at subsequent time instances (after some time has lapsed), thus requiring us to update the training set and the training model. This renders the whole PR problem intriguing.

[1] As requested by the AE and Referees, a table of abbreviations has been provided. However, to avoid distractions from the main focus of the paper, this table is included in the Appendix. It can be easily moved into the body of the paper if suggested by the publishing staff. We thank the Referees for requesting this.

Real-life aspects of this Paradigm: Interestingly enough, the current model is not merely hypothetical. Rather, it is, in actuality, the model that is encountered in most real-life applications. Consider, for example, a cancer identification program. After the measurements are made and the features are available, the trained PR program classifies the testing sample that is presented to the system. However, after the machine achieves the classification, it is *always* the case that a physician comes into the loop and informs the system, and the patient, whether the PR classification decision was correct or not. This *new* information, provided by the physician (possibly, due to additional tests and measurements), can be then used to update the training process, and to enhance the classification phase.

Hybrid nature of this Paradigm: The knowledgeable reader will see that this current model represents a hybrid of traditional statistical PR, and the so-called "Sequential" PR. As far as we know, such a hybrid paradigm is completely unexplored in the literature. While the data arrives in streams, the class identity of any given sample is known after a few time instances after the original classification, and this *true* identity can be subsequently used to update the class conditional distributions prior to any further classification is attempted.

Testing aspects of this Paradigm: Unfortunately for us, data for this model of PR is not currently available, although it happens all the time. That is why we had to work with artificial data in our initial submission. But this current paper also includes results from real-life data adapted for this model of computation.

Salient theoretical aspects of our solution: While all of the above issues make the paper novel in and of itself, we have chosen to render the problem more complex by considering the case where the classes' stochastic properties potentially vary with time as more instances become available. From this perspective, with regard to the training, we will argue that using "strong" estimators that converge with probability of 1 is inefficient for tracking the statistics of the data distributions in NSEs. However, "weak" estimator approaches are able to rapidly unlearn what they have learned and adapt the learning model to new observations.

1.2 Unique Applications of the SLWE in Non-stationary Environments

A huge body of research is available when it concerns the prior art of classification in non-stationary data streams. This will be surveyed, in some detail, in Section 2. However, at this juncture[2], we briefly refer to the preliminary theoretical foundations for this present work found in [52], where a particular family of weak estimators, referred to as Stochastic Learning Weak Estimation (SLWE) methods, was introduced. The fascinating feature of "weak" estimators, that make them converge in a "weak" manner, render them most effective methods for estimation in NSEs, and consequently, in the classification of such data. It is pertinent to mention that the SLWE has been successfully used to solve simple two-class classification problems [52] by applying it on non-stationary *one*-dimensional datasets. It has also been used in many real-life application domains, as mentioned below. However, the application of the model in *multi*-dimensional datasets has been unreported. This is, precisely, what we highlight in this article, namely, the performance of the SLWE for more complex classification schemes with *multi*-dimensional data, and for the more intricate scenario where the identity of a testing sample is known after a small time interval, forcing the training stream to be continuously updated, and the class-conditional distributions to, thus, be non-stationary. The applications of the SLWE's theoretical results to this model of computation is completely new, and to the best of our knowledge, quite pioneering. We briefly elaborate on this here.

[2]We are grateful to the anonymous Referee who specifically requested this section.

Although the state-of-the-art will be surveyed later, we briefly mention that online data stream mining techniques have been applied in several key areas for the monitoring of streaming data, such as in spam-filtering [7, 74], network intrusion detection [17, 29, 36, 73], and time varying PR [53, 64]. De Oca *et al.* [17] proposed a nonparametric algorithm for network surveillance applications in NSEs. They adapted the classic Cumulative Sum (CUSUM) change detection algorithm that uses a defined time-slot structure in order to handle time varying distributions. Hajji [29] developed a parametric algorithm for real time detection of network anomalies. He used stochastic approximation of the MLE function in order to monitor the non-stationary nature of network traffic and to detect unusual changes in it. Kim *et al.* [36] proposed a multi-chart CUSUM change detection procedure for the detection of DOS attacks in network traffic. Robinson *et al.* [57] proposed a method for monitoring and detecting behavioral changes from an event stream of patient actions.

On the other hand, the SLWE approach has been used successfully in a variety of real-life ML applications specifically those involving estimating binomial/multinomial distribution in NSEs. Rueda and Oommen [59] utilized the SLWE approach for data compression in NSEs, in which the SLWE was applied for an adaptive single-pass encoding process to estimate and update the probabilities of the source symbols. It was also shown in [52] that using the weak estimator for distributional change detection in NSEs surpassed the performance of the MLE method. Oommen and Misra [51] applied the weak-estimation learning scheme to propose a new fault-tolerant routing approach for mobile ad-hoc networks, named the Weak-Estimation Fault-Tolerant Routing (WEFTR) algorithm. They utilized the SLWE approach to estimate the probability of the delivery of packets among the available paths at any moment. The SLWE was also used by Stensby *et al.* [64] for language detection and tracking multilingual online documents. Zhan *et al.* [74] applied the SLWE for anomaly detection, specifically for the detection of spam emails, when the underlying distributions changed with time. They employed the SLWE approach for spam filtering based on a naïve Bayes classification scheme. Oommen *et al.* [53] also proposed a strategy for learning and tracking the users' time varying interests to find out their preferences in social networks. Most recently, Khatua and Misra [34] developed a controllable reactive jamming detection scheme, referred to as Controllable Reactive Jamming Detection (CURD), which applies the CUSUM-test and the weak estimation approach in order to estimate the probability of collisions in packet transmission.

Theoretical Extensions of the SPL: In [30], Hammer and Yazidi presented a method by which the SLWE could detect abrupt changes in data streams. To achieve this goal, two parallel weak estimators were run in parallel. In this sense, both estimators were updated simultaneously upon reception of the same observation. By an appropriate choice of the updating parameters, one of the two parallel weak estimators was rendered adequate to dynamic environments by imposing a choice of the update parameter around 0.9 which allowed adaptivity. The second weak estimator was designed to cope with stationary environments by using a time-dependent learning parameter that gradually approached unity as time proceeded. A sophisticated statistical test based on the distance between both the estimators in terms of the respective variance and mean was developed to detect changes. Intuitively, an increase in the distance between both estimators is a sign of a change in the underlying distribution. Furthermore, whenever a change was detected, the estimate of the stationary SLWE was re-initialized since it was deemed to be off-track. In [46], Mofrad *et al.* proposed to enhance the Stochastic Point Location (SPL) by combining it with the SLWE. By virtue of introducing the SLWE tracking as a part of the SPL mechanism, the authors showed that their enhanced SPL solution not only outperformed the classical SPL solution in terms of speed and accuracy, but it was also able

to estimate the probability according to which the Oracle provideed the truth. Furthermore, the article provided some real-life applications that showed how the SLWE could improve the estimates of the discretized estimator proposed in [71].

Mohan *et al.* [47] developed an SLWE that worked in a batch mode in order to estimate the matching probabilities of different firewall rules. Those estimates were given as inputs to a re-ordering algorithm that optimized the matching time of the firewall by taking into account the matching probability of each rule obtained using the SLWE, and some precedence relationships between the rules.

In [31], the authors modeled the data center consolidation problem using a large-scale Markov Decision Process. In order to track the time-varying computing resource usage in the data center, the authors extended the SLWE by combining it with a sliding window approach. When the number of observation in the sliding window was less than a given threshold computed using confidence intervals, the estimates were based on the SLWE. In the opposite case, whenever enough number of samples were collected, the sliding window approach was used. On the other hand, Bhaduri and his collaborators [6] presented a simple method in order to allow continuous time updates of the SLWE.

Some recent applications of the SPL: In [3], the SLWE was used to estimate the First Order Markov probabilities in a activity recognition system. Those probabilities were further employed to detect switching points. Among some of the recent applications of the SLWE, we report the problem of link prediction [14]. The latter work resorted to the estimates from the SLWE as a feature in a deep learning model to predict the likelihood creation and destruction of links in an evolving network over time.

Recent Concept Drift Schemes: The work in [18] introduced ExStream, an incremental learning system for data streams. ExStream detected concept drift by detecting changes in the data stream using three different levels. ExStream is based on a similar idea to (ADaptive WINdows) ADWIN [9, 10]. In this sense, ExStream ran different parallel explanatory models and observed the magnitude of their changes in each model in order to detect concept drift.

Recently, an new concept drift method called Diversity and Transfer-based Ensemble Learning (DTEL) was proposed [65]. DTEL employed an ensemble approach for classification and for adapting to concept drift. Instead of training models from scratch whenever a new "chunk" of the data was received, DTEL used the concept of transfer learning and therefore relied on historical models as initial models. The transferred models were further trained with new data.

The outcome of research in this area can also be used in other real-life applications, and we mention one such application that is particularly interesting because of its current ML implications. Indeed, such a PR scheme can be applied for the detection of the source of news streams. For example, an observed stream could be either a live video broadcasted from a TV channel or news released in a textual form, for example, on the internet. This problem has been studied by several researchers by considering shots from the video and using them in the classification to fall into one of a few predefined classes. Our method could, however, be used to simplify this problem by considering the news streams that arrive in the form of text blocks extracted from the closed captioning text embedded in the video streams. A similar problem was considered by Oommen and Rueda in [52], in which they analyzed bit streams generated from two sources of news, namely "Sports" and "Business".

Finding suitable non-stationary data streams to be used for testing our paradigm (i.e., where testing samples become training samples after a time interval) is challenging because almost all the real-world benchmark data sets provided by, for example, the UCI machine learning repository are designed for stationary environments. Further, as mentioned before, the data for sets where the training and testing are interleaved is non-existent. It should be noted that the available non-stationary news streams utilized in [52] only included *binomial* data, as no multinomial data sets were available for testing. Thus, due to the lack of multinomial non-stationary real-world datasets, we will first use synthetic benchmarks, and then achieve the testing for datasets *extracted* from the more general, readily available, data for NSEs. The results that we present are conclusive.

1.3 Contributions of the Paper

The novelty and contributions of the paper can be briefly stated as follows:

- This paper, first of all, presents a fairly comprehensive survey of the field of the online classification of data streams.
- However, unlike the strategies reported in the literature, our novel *online* classification scheme is composed of three phases:
  1. In the first phase, the model learns from the available labeled samples;
  2. In the second phase, the learned model predicts the class label of the unlabeled instances that are currently being observed;
  3. In the third phase, after knowing the true class labels of these recently-classified instances, the classification model is adjusted in an *online* manner;

  As far as we know, such a PR and ML paradigm is pioneering.
- Most of the data stream mining approaches build an initial model from a sliding window of recently-observed instances. Subsequently, they refine the learning model periodically or whenever its performance degrades based on the current window. We, however, deal with the phenomenon of concept drift in NSEs, with a scheme that is more efficient and accurate than the state-of-the-art. We emphasize that it does not invoke a sliding window. Further, this non-stationarity could even be *abrupt*.
- Our classifier scheme provides a real-time self-adjusting learning model, utilizing the multiplication-based update algorithm of the SLWE, as new labeled instances arrive. Instead of maintaining counters to keep data statistics, we have used a technique that invokes data estimators that do not converge w.p. 1. Thus, we do not need to rebuild our model when a change in the distributions is detected.
- The paper also includes extensive experimental results for both the binomial and multinomial (two dimensional and multi-dimensional) distributions. It demonstrates the efficiency of the proposed classification schemes in achieving a good performance for data streams involving non-stationary distributions under different scenarios of concept drift, where the training and testing samples are not completely dichotomized. The paper also contains results from two real-life scenarios. The performance measure for these experiments are the superiority of the estimates obtained for the MLEW and SLWE, and the corresponding recognition accuracies obtained when using them in classification.

1.4 Organization of the paper

In Section 2, we review the literature[3] available on the families of approaches reported for classification and estimation in stationary and NSEs. We also survey the available estimation approaches for streams with unknown dynamics in NSEs. We proceed with discussing the issues and challenges encountered when one learns from data streams and provide a brief explanation about the theoretical properties of the SLWE. In Section 3, we present the details of the design and implementation of the online classifier within this new paradigm. We then explain how this solution can be used to perform online classification, and present the new experimental framework for concept drift in Section 4. This section and the next also contain the experimental results we have obtained from rigorous testing. Section 6 concludes the paper.

2 The State-of-the-Art

In this section, we briefly survey[4] the state-of-the-art when it concerns classification in dynamic data streams and for the estimation problem in such domains[5]. Thus, in Section 2.1, we survey the field of achieving classification in NSEs, and in Section 2.2, we survey the techniques used for *training* in such domains.

2.1 Survey of Classification of Data Streams in NSEs

According to the data stream mining literature, algorithms have one or more of the following modules: a Memory module, an Estimator module, and a Change Detector module [8]. The Memory module is a component that stores summaries of all the sample data and attempts to characterize the *current* data distribution. Data in NSEs can be handled by three different approaches, namely, by using partial memory, by window-based approaches and by instance-based methods. The term "partial memory" refers to the case when only a part of the information pertaining to the training samples are stored and used regularly in the training. In window-based approaches, the data is presented as "chunks", and finally, in instance-based methods, the data is processed upon its arrival. In dynamic environments with non-stationary distributions, the Memory module indicates the forgetting mechanism of the mining algorithm in order to adapt the learning model to newer observations and to forget old information.

The Estimator module uses the information contained in the Memory or only the observed information to estimate the desired statistics of the time varying data stream. The Change Detector module involves the techniques or mechanisms utilized for detecting explicit drifts and changes, and provides an "alarm" signal whenever a change is detected based on the estimator's outputs.

Apart from the above schemes, many other incremental approaches have been proposed that infer change points during estimation, and use the new data to adapt the learning model trained from historical streaming data. The learning model in incremental approaches is adapted to the most recently-received instances of the streaming data. Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of training examples available at time $t = 1 \ldots n$. An incremental approach produces a sequence of hypotheses $\{\ldots, H_{i-1}, H_i, \ldots\}$ from the training sequence, where each hypothesis, H_i, is derived from the previous hypothesis, H_{i-1}, and the example x_i. In general, in order to detect concept changes in these types of approaches, some characteristics of the data stream (e.g., performance measures, data distribution,

[3]The bibliography we include is very extensive, as requested by the anonymous Referees.

[4]We are extremely grateful to the anonymous Referee who requested such a detailed and dichotomized survey.

[5]It is not so easy to dichotomize the research done to differentiate between the training and testing (classification) phases. But, we hope that the reader sees the intent of the survey in the two subsections.

properties of data, or an appropriate statistical function) are monitored over time. When the parameters switch during the monitoring process, the algorithm should be able to adapt the model to these changes.

We now briefly review some *other* schemes used for learning in NSEs. The review here will not be exhaustive because the methods explained can be considered to be the basis for other modified approaches.

2.1.1 FLORA

Widmer and Kubat [69], presented the FLORA family of algorithms as one of the first supervised incremental learning systems for a data stream. The initial FLORA algorithm used a fixed-size sliding window scheme. At each time step, the elements in the training window were used to incrementally update the learning model. The updating of the model involved two processes: an incremental *learning* process that updated the concept description based on the new data, and an incremental *forgetting* process that discarded the out-of-date (or stale) data.

The initial FLORA system did not perform well on large and complex data domains. Thus, FLORA2 was developed to solve the problem of working with a fixed window size, by using a heuristic approach to adjust the window size dynamically. Further improvements of the FLORA were presented to deal with recurring concepts (FLORA3) and noisy data (FLORA4).

2.1.2 Statistical Process Control (SPC)

The SPC was presented by Gama *et al.* [25] for change detection in the context of data streams. The principle motivating the detection of concept drift using the SPC is to trace the probability of the error rate for the streamed observations. While monitoring the errors, the SPC provides three possible states, namely, "in control", "out of control" and "warning" to define a state when a warning has to be given, and when levels of changes appear in the stream. When the error rate is lower than the first (lower) defined threshold, the system is said to be in an "in control" state, and the current model is updated considering the arriving data. When the error exceeds that threshold, the system enters the "warning" state. In the "warning" state, the system stores the corresponding time as the warning time, t_w, and buffers the incoming data that appears subsequent to t_w. In the "warning" mode, if the error rate drops below the lower threshold, the "warning" mode is canceled and the warning time is reset. However, in case of an increasing error rate that reaches the second threshold, a concept change is declared and the learning model is retrained from the buffered data that appeared after t_w.

2.1.3 ADWIN

Bifet and Gavalda [9, 10] proposed an adaptive sliding window scheme named ADWIN for change detection and for estimating statistics from the data stream. It was shown that the ADWIN algorithm outperforms the SPC approach and that it has the ability to provide rigorous guarantees on false positive and false negative rates. The initial version of ADWIN keeps a variable-length sliding window, W, of the most recent instances by considering the hypothesis that there is no change in the average value inside the window. To achieve this, the distributions of the sub-windows of the W window are compared using the Hoeffding bound, and whenever there is a significant difference, the algorithm removes all instances of the older sub-windows and only keeps the new concepts for the next step. Thus, a change is reliably detected whenever the window shrinks, and the average over the existing window can be considered as an estimate of the current average in the data stream.

To be more specific, consider a sequence of real values $\{x_1, x_2, \ldots, x_t, \ldots\}$ that is generated according to the distribution D_t at time t. Let n denote the length of the W window, $\hat{\mu_t}$ be the observed average of the elements in W, and μ_w be the true average value of μ_t for $t \in W$. Whenever two "large enough" sub-windows of W demonstrate "distinct enough" averages, the system infers that the corresponding expected values are different, and the older fragment of the window should be dropped. The observed average in both sub-windows are "distinct enough" when they differ by more than the threshold ϵ_{cut}:

$$\epsilon_{cut} = \sqrt{\frac{1}{2m}.\ln\frac{4}{\delta'}}, \text{where} \tag{1}$$

$$m = \frac{1}{1/n_0 + 1/n_1}, \quad \text{and} \quad \delta^{'} = \frac{\delta}{n}, \tag{2}$$

where n_0 and n_1 denote the lengths of the two sub-windows, and where δ is a confidence bound. The above, Eq. (1) and (2), are a direct consequence of invoking the bounds imposed by the Hoeffding inequality [9, 10].

Using the Hoeffding bound greatly overestimates the probability of large deviations for distributions with a small variance, which degrades the ADWIN's performance. Besides, it is also computationally demanding [55].

The ADWIN approach is, in fact, a linear estimator enhanced with a change detector. In order to improve the basic ADWIN method's performance, Bifet [8] replaced the linear estimator by an adaptive Kalman filter, where the covariances of $w(n)$ and $v(n)$ were set to $n^2/50$ and $200/n$ respectively, where n is the length of the window maintained by ADWIN.

2.1.4 Other Miscellaneous Approaches

To perform classification in dynamic systems, a common strategy is to transfer acclaimed classification schemes for static data to a dynamic environment by training the classifier on a sliding window, or an exponential weighting of historic data. A challenge is that for most classifiers, like the Support Vector Machine (SVM), it is challenging to recursively update the parameters of the classifiers when new data arrive. Several attempts have been suggested for different classifiers to overcome this challenge.

The Hoeffding tree is a decision tree classifier for data streams [19]. Traditional decision trees need to scan the training data many times to select the splitting attribute. However, this requirement is infeasible in the data stream environment. To overcome this limitation, the Hoeffding bound is used to choose an optimal splitting attribute within receiving a sufficient amount of data objects.

Seidl *et al.* proposed a novel index-based classifier called the Bayes tree [62]. Adapted from the R*-tree [5], the Bayes tree generates a hierarchical Gaussian-mixture tree to represent the entire data set. Each tree node contains statistics of the data objects including a minimum bounding rectangle, the number of data objects, the linear sum and the quadratic sum of all the data objects.

The SVM has demonstrated its prominent performance in many ML problems with static data sets. However, it is very expensive to use SMVs in large-scale applications due to its time and memory complexity. Tsang *et al.* proposed the Core Vector Machine (CVM) algorithm that uses the Minimum Enclosing Ball (MEB) to reduce its complexity [67].

For other classifiers based on linear regression (ordinary least squares), the parameters of the model can be updated recursively, [32, 33]. Even the least squares approach that uses Tikhonov regularization (ridge regression)

can be updated recursively like the traditional least squares approach.

In [45], the authors proposed and evaluated OLIN, an online classification system. OLIN dynamically adjusts the size of the training window and the number of new samples needed between the time instances when the model is reconstructed to characterize the current rate of concept drift. The advantage of OLIN is that by using a fixed amount of computer resources, it is able to produce models, which have nearly the same accuracy as the ones that would be produced by periodically re-constructing the model from *all* accumulated instances.

Another scheme, which deals with *incremental* ensemble classifiers for NSEs, was described in [54]. These authors proposed a scheme for using ensembles for stream classification. Their method, M3, was based on a weighted majority ensemble of heterogeneous model types, and these were blended using Reinforcement Learning techniques. In its central mechanism, M3 was quite standard. However, it was designed in such a way so as to be able to specifically adapt quickly to the dynamic and evolving nature of data from the NSE. It further required very little overhead to achieve this. The authors showed that for the UCI benchmark and for synthetic streams, M3 was able to display a strong gain over the baseline method when the knowledge of the ground truth was limited.

In [41], the authors proposed *One*-class classifiers which were able to achieve incremental learning for data streams with concept drift. This scheme also provided a forgetting mechanism. This paper reported a novel modification of the weighted *one*-class SVM adapted to the NSEs, and it was able to adapt its decision boundary to new, incoming data, and to also employ a forgetting mechanism which boosted the ability of the classifier to follow the model changes. To do this, they proposed several different strategies for *incremental* learning and forgetting, and evaluated them on the basis of several real data streams, and demonstrated the power of their methods.

2.2 Survey on Training Schemes in NSEs

Estimation theory is a fundamental subject that is central to the fields of PR and data mining. The majority of problems in PR require the estimation of the unknown parameters that characterize the underlying distributions.

In this section, we present a brief survey of how parameter and distribution estimation play a crucial role in classification and learning. This section surveys, in some detail, the literature available on the families of approaches for estimation, and proceeds to discuss the issues and challenges encountered when one learns from data streams. In particular, we focus on the special issues to be considered when we work with change detection. Indeed, we shall survey estimation approaches that have been developed to learn from streams with unknown dynamics in NSEs.

2.2.1 MLE-based Schemes

The majority of Machine Learning approaches have been developed to deal with data domains in which the underlying distributions are stationary. Learning in these environments is similar to batch learning. It is pertinent to mention that all of the benchmark data sets that deal with Machine Learning and PR fall into this class.

One encounters an additional problem in learning when the data is based on the properties of data streams. The issue at stake is that the data distribution, in these cases, can be non-stationary, implying that the distribution or characterizing aspects of the features change over time. In NSEs, the data generation phenomenon itself may change over time, which, in turn, leads to a variation in the data distribution. The goal of learning approaches in NSEs is to estimate the parameters of the distributions, and to adapt to any abrupt and/or gradual changes occurring in the environment. In other words, the learning and classification models must be updated when significant changes

in the underlying data stream are detected.

The important issue here is that the estimation and training must be achieved without a knowledge of when and how the environment has changed, rendering the problem to be far from trivial.

2.2.2 Stale and Recent Samples

It is very important to understand that in NSEs, old observations become irrelevant to the current state or might even have a negative effect on the learning process. For data domains of these kinds, the estimation mechanisms should be able to incorporate phenomena akin to concept drift. They should be able to forget outdated data and adapt the estimation to the more recently-observed data elements. Within the context of the new model that we investigate, the body of this paper deals with training and testing in such NSEs, except that in our cases, tested samples become training samples at a subsequent juncture.

Change detection, in and of itself, is a very complex task, as its design is intended to be a trade-off between detecting real changes and avoiding false alarms. A significant amount of work has been performed in the area of concept change detection by both the statistical and Machine Learning communities. The subject of change detection was first employed in the manufacturing and quality control applications in the 1920-1930's [4, 56]. By the introduction of sequential analysis, later in the 1950-1960's, sequential detection procedures were developed, which considered the sequence of observations to detect unusual trends and patterns in the data.

With parallel considerations, learning in NSEs is of great importance, and this problem is closely related to that of detecting concept changes and also of estimating the dynamic distribution associated with a set of data. Basseville and Nikiforov [4], Chen and Gupta [13], and Sebastian and Gama [61] have provided fairly good and detailed surveys on the topic of change detection methods. The methods presented in the literature are different with respect to the type of change they are expected to detect, and the underlying assumptions made about the streaming data. As mentioned earlier, in general, most algorithms in the data stream mining literature have one or more of the following modules: a Memory module, an Estimator module, and a Change Detector module [8].

2.2.3 Sliding Window Approaches

Traditionally available methods that cope with non-stationary distributions resort, in one way or the other, to the so-called *sliding window* approach, which is a limited-time variant of the well-known Maximum Likelihood Estimation (MLE) scheme. The latter model is useful for discounting stale data in data stream observations. Data samples arrive continuously and only the most recent observations are used to compute the current estimates. Any data occurring outside the current window is forgotten and replaced by the new data. The problem with using sliding windows is the following: If the time window is too small the corresponding estimates tend to be poor. As opposed to this, if time window is too large, the estimates prior to the change of the parameter have too much influence on the new estimates. Moreover, the observations during the entire window width must be maintained and updated during the process of estimation.

The algorithm, typically, considers a window of size W and divides the data stream into a sequence of data chunks. At each time step, learning is carried out based only on last W samples that are included in the window. Sliding window models are designed based on the assumption that the most recent information is more relevant than the historical data, which is similar to *first in-first out* data structures. At time t_j, when element j arrives,

element $j - W$ is forgotten, where W indicates the size of the window [24]. In fact, at every time instant, the learning model of the data stream is generated using only the W samples resident in the window.

Several sliding window models have been presented in the literature [43, 48]. Kuncheva [43] presented a semi-parametric log-likelihood change detector based on Kullback-Leibler (KL) statistics. The author applied a log-likelihood framework that accommodates the KL distance and the Hotelling's t^2 test for equal means in order to detect changes in streaming multi-dimensional data. An implementation of the fixed cumulative windowing scheme was proposed by Kifer *et al.* [35]. The authors here applied two sliding windows in their scheme, the first being a reference window, which was used as a baseline to detect changes, and the second being a "current window" to collect samples. They proposed an algorithm based on the Kolmogorov Smirnoff (KS) test statistic using a KS Tree. The latter specified if the observed samples were generated from the same distribution. The high computational cost of maintaining a balanced KS tree, is the main problem associated with this approach.

2.2.4 Change Point Detection Schemes

Apart from the sliding window approach, many other methods have been proposed, which deal with the problem of detecting change points during estimation. In general, there are two major competitive sequential change-point detection algorithms: Page's Cumulative Sum (CUSUM) [4] detection procedure, and the Shiryaev-Roberts-Pollak detection procedure. In [63], Shiryayev used a Bayesian approach to detect changes in the parameter's distribution, where the change points were assumed to obey a geometric distribution. CUSUM is motivated by a maximum likelihood ratio test for the hypothesis that a change has occurred. Both approaches utilize the log-likelihood ratio for the hypotheses that the change occurred at the point, and that there is no change. Inherent limitations of CUSUM and the Shiryaev-Roberts-Pollak approaches for on-line implementation are the demanding computational and memory requirements. In contrast to the CUSUM and the Shiryaev−Roberts−Pollak approaches, our SLWE avoids the intensive computations of ratios, and does not invoke hypothesis testing.

In earlier works [38, 39, 40], Koychev *et al.* introduced the concept of Gradual Forgetting (GF). The GF process relies on assigning weights that decrease over time to the observations. In this sense, the GF approach assigns most weight to the more recent observations, and a lower weight to the more-distant observations. Hence, the influence of old observations (on the running estimates) decreases with time. The GF can be an enhancement to the sliding window paradigm since observations within each sliding window are weighted using a GF function[40] .

2.2.5 Other Miscellaneous Estimation Schemes

As opposed to the above-mentioned papers, in the last decades, a wide range of techniques for estimation in dynamically changing environments have appeared. We provide here a brief overview of representative *on-line* approaches particularly relevant to the family of techniques pioneered in the present paper. For a more detailed and comprehensive treatment, we refer the reader to recent surveys [26, 42], which include approaches based on *adaptive windowing*, *aging factors*, *instance selection* and *instance weighting*. Additionally, there is a distinction between *passive* approaches, which intrinsically adapt to changes, and *active* approaches that search for changes.

With regard to categorizing these approaches, they differ in terms of their memory management and forgetting mechanisms [26]. A memory management module aims to control the data points that are to be kept in memory for learning. As opposed to this, *forgetting* refers to the task of erasing previously-learned information. One data management strategy, referred to as "instance selection", is based on storing instances relevant to the current

situation. Here, the simplest approach is to only keep a single data point in memory, with the task of forgetting being modeled by the gradual dilution of its relevance through adaptation of the model's parameters [20]. Such a strategy can be expanded to store multiple data points, using a window of fixed size. The data points in the window are used to establish the current estimate, and at each time step, the oldest data point is replaced with the newest one, using a First-In First-Out (FIFO) policy. Slow changes are best captured by a larger window, while rapid changes require a correspondingly smaller window. To address this conflict, an alternative scheme involves variable-sized sliding windows, referred to as *adaptive windowing* [69]. In brief, the overall strategy is to allow the window size to grow over time, and to use a dedicated change detector to reset the window size when a change is detected, resulting in windows with an adaptive size. More advanced approaches carefully pick out prototypes, representing the concept being tracked [12], thus handling concept drift by replacing the prototypes themselves to reflect changes.

Another strategy, which avoids windowing, involves using *aging* factors. In this approach, while training data spans all of the data points, each data point is assigned a weight that reflects its importance. Recent data points are prioritized, for instance using a fading factor that gradually reduces the influence of older data points [15]. The decay can be linear [38] or exponential [37]. By using weights combined with instance selection, referred to as instance weighting, one can take advantage of the ability of SVMs to process weighted instances [37].

As mentioned above, most of the above approaches are passive in the sense that they gradually adjust to changes, without explicitly pinpointing them. In contrast, another class of approaches actively searches for changes using change detectors. One solution is to maintain an ensemble of estimators, combined using voting, such as the dynamic integration of classifiers found in [68]. The impact of each estimator is based on assessing them on recent data, prioritizing the one that produces the least error, where the change of priority indicates an underlying change in the distribution of data points. One can also monitor estimation error using various mechanisms, or the raw data itself. Changes in estimation error signal a change in the distribution of data points, triggering re-estimation [58]. In all brevity, changes are detected based on comparing sections of data, using statistical analysis to detect distributional changes, i.e., abrupt or gradual changes in the mean of the data points when compared with a baseline mean with a random noise component. One option is also to keep a reference window and to compare recent windows with the reference window to detect changes [21]. This can, for example, be done based on comparing the probability distributions of the reference window and the recent window using the KL divergence [16, 60].

For a more comprehensive survey on classification of data streams under concept drift we refer the reader to the following two recent surveys [2, 23]. We will not go into any more details here in the interest of brevity. However, using a completely different tool-set, Oommen and Rueda [52] presented a strategy by which the parameters of a binomial/multinomial distribution can be estimated when the underlying distribution is non-stationary, which is the core of this research endeavor. This scheme is explained in the next subsection.

2.3 Stochastic Learning Weak Estimator (SLWE)

The method proposed by Oommen and Rueda [52] has been referred to as the Stochastic Learning Weak Estimator (SLWE), and is based on the principles of *continuous* stochastic Learning Automata (LA)[6].

Since the SLWE has been earlier used for *two*-class *uni*-dimensional pattern classification problems, the aim of this section is:

[6]This will be the focus of this paper, and we elaborate on this in greater detail later.

1. To demonstrate that the SLWE is a powerful tool for *multi*-dimensional classification problems operating in NSEs;

2. To demonstrate that the SLWE is very effective for *uni*-dimensional and *multi*-dimensional domains when the number of classes is greater than two;

3. To, most importantly, show the power of the SLWE within the context of the paradigm that we are currently investigating, namely, the one in which the testing patterns can subsequently be considered as training patterns, once their true class identities are known. More precisely, the model operates with the understanding that the correct classes of previously-classified patterns become available at subsequent time instances (after some time has lapsed), requiring us to update the training set and the training model.

Most of the data stream mining approaches have an estimator module in order to keep the statistics of the data distribution in NSEs updated. However, it can be argued that using "strong" estimators such as the MLE and the Bayesian estimators that converge with probability of 1 are inefficient for dynamic NSEs. In NSEs, it is essential to use estimator schemes which can adapt to the model promptly according to the new observations. In other words, effective methods for estimation in NSEs are those which are able to quickly unlearn what they have learned.

Using the principles of stochastic learning, Oommen and Reuda [52] proposed a strategy to solve the problem of estimating the parameters of a binomial or multinomial distribution efficiently in NSEs. This method is referred to as the SLWE, where the convergence of the estimate is "weak", i.e., with respect to the first and second moments. Unlike the traditional MLE and the Bayesian estimators, which demonstrate strong convergence, the SLWE converges fairly quickly to the true value, and it is able to just as quickly "unlearn" the learning model trained from the historical data in order to adapt to the new data.

The SLWE is an estimator method that estimates the parameters of a binomial/multinomial distribution when the underlying distribution is non-stationary. In NSEs, the SLWE updates the estimate of the distribution's probabilities at each time-instant based on the new observations. The updating, however, is achieved by a *multiplicative* rule. To formally introduce the SLWE, let X be a random variable of a multinomial[7] distribution, which can take the values from the set $\{\text{'1'}, \ldots, \text{'}r\text{'}\}$ with the probability of S, where $S = [s_1, \ldots, s_r]^T$ and $\sum_{i=1}^{r} s_i = 1$. In the other words: $X =$ 'i' with probability s_i.

Consider $x(n)$ as a concrete realization of X at time 'n'. In order to estimate the vector S, the SLWE maintains a running estimate $P(n) = [p_1(n), p_2(n), \ldots, p_r(n)]^T$ of vector S, where $p_i(n)$ is the estimation of s_i at time 'n', for $i = 1, \ldots, r$. The value of $p_i(n)$ is updated with respect to the coming data at each time instance, where Eqs. (3) and (4) show the updating rules:

$$p_i(n+1) \leftarrow p_i + (1-\lambda)\sum_{j \neq i} p_j \quad \text{when } x(n) = i \tag{3}$$

$$\leftarrow \lambda p_i \quad \text{when } x(n) \neq i. \tag{4}$$

Similar to the binomial case, the authors of [52] explicitly derived the dependence of $\mathrm{E}[P(n+1)]$ on $\mathrm{E}[P(n)]$, demonstrating the ergodic nature of the Markov matrix. The paper[8] also derived two explicit results concerning

[7] The case of estimating binomial distributions is a particular case of multinomial distributions where $r = 2$.

[8] In the interest of brevity, the proofs of the theorems were omitted in the initial submission. However, they have now been included to satisfy the request of one anonymous Referee. We believe that their inclusion improves the quality of this version, and we are grateful for the Referee's suggestion.

the convergence of the expected vector $P(.)$ to S, and the rate of convergence based on the learning parameter, λ.

Theorem 1. Consider $P(n)$, the estimate of the multinomial distribution S at time 'n', which is obtained by Eqs. (3) and (4). Then, $\mathrm{E}[P(\infty)] = S$.

Proof. Equations (3) and (4) can be rewritten as follow:

$$\begin{aligned} p_i(n+1) &\leftarrow p_i + (1-\lambda)(1-p_i) \quad \text{w.p. } s_i && (5)\\ &\leftarrow \lambda p_i \quad \text{w.p. } \sum_{j\neq i} s_j\,. && (6)\end{aligned}$$

Therefore, the expected value of P at time '$n+1$' given the estimated probabilities at time 'n', P, is:

$$\begin{aligned} \mathrm{E}[p_i(n+1)|P] &= p_i s_i + (1-\lambda-p_i+\lambda p_i)s_i + \lambda p_i(1-s_i) && (7)\\ &= p_i s_i + s_i - \lambda s_i - p_i s_i + \lambda p_i s_i + \lambda p_i - \lambda p_i s_i && (8)\\ &= (1-\lambda)s_i + \lambda p_i\,. && (9)\end{aligned}$$

Taking expectations a second time leads to:

$$\mathrm{E}[p_i(n+1)] = (1-\lambda)s_i + \lambda \mathrm{E}[p_i(n)]\,. \tag{10}$$

Both $\mathrm{E}[p_i(n+1)]$ and $\mathrm{E}[p_i(n)]$ converge to $\mathrm{E}[p_i(\infty)]$ as $n \to \infty$, and hence it can be concluded that:

$$\begin{aligned} \mathrm{E}[p_i(\infty)](1-\lambda) &= (1-\lambda)s_i && (11)\\ \Rightarrow \mathrm{E}[p_i(\infty)] &= s_i\,. && (12)\end{aligned}$$

This procedure can be repeated for every component p_i of P, which proves the first claim. □

Theorem 2. Consider $P(n)$, the estimate of the multinomial distribution S at time 'n', which is obtained by Eqs. (3) and (4). The expected value of P at time '$n+1$' is related to the expectation of $P(n)$ as $\mathrm{E}[P(n+1)] = \mathbf{M}^T \mathrm{E}[P(n)]$, where $\mathbf{M}$ is a Markov matrix. Further, every off-diagonal term of the stochastic matrix, $\mathbf{M}$, has the *same* multiplicative factor, $(1-\lambda)$, and the final solution of this vector *difference equation* is independent of λ.

Proof. According to Equation (9), the conditional expected probability, $\mathrm{E}[p_1(n+1)|P]$ can be written as:

$$\mathrm{E}[p_1(n+1)|P] = (1-\lambda)s_1 \sum_{j=1}^{r} p_j + \lambda p_1\,. \tag{13}$$

Generally, for $i = 1, \ldots, r$ the conditional expectations of $p_i(n+1)$, is as follows:

$$\mathrm{E}[p_i(n+1)|P] = (1-\lambda)s_i \sum_{j=1}^{r} p_j + \lambda p_i\,. \tag{14}$$

By organizing the terms of Equation (14) in a vectorial manner for all $i = 1, \ldots, r$, it can be concluded that

$\mathrm{E}[P(n+1)] = \mathbf{M}^T \mathrm{E}[P(n)]$, where the stochastic matrix, $\mathbf{M}$, is:

$$\mathbf{M} = \begin{bmatrix} (1-\lambda)s_1 + \lambda & (1-\lambda)s_2 & \cdots & (1-\lambda)s_r \\ (1-\lambda)s_1 & (1-\lambda)s_2 + \lambda & \cdots & (1-\lambda)s_r \\ \vdots & \vdots & \ddots & \vdots \\ (1-\lambda)s_1 & (1-\lambda)s_2 & \cdots & (1-\lambda)s_r + \lambda \end{bmatrix} . \tag{15}$$

By solving the following vectorial difference equation and considering the limit as n tends to infinity, the limiting solution for $\mathrm{E}[P(n)]$ is obtained.

$$\mathrm{E}[P(\infty)] = \mathbf{M}^T \mathrm{E}[P(\infty)] . \tag{16}$$

The Equation (15) can be rewritten as follows:

$$\mathbf{M} = (1-\lambda)[S|S|\cdots|S]^T + \lambda \mathbf{I} , \tag{17}$$

which shows that every element of $(\mathbf{M} - \lambda\mathbf{I})$ has a common multiplicative factor of $(1-\lambda)$ (as does every off-diagonal element of $\mathbf{M}$ itself). $\mathbf{M}$ Matrix in Equation (16) is replaced by the result of Equation (17) as follows:

$$\mathrm{E}[P(\infty)] = (1-\lambda)[S|S|\cdots|S]^T \mathrm{E}[P(\infty)] + \lambda \mathbf{I} \mathrm{E}[P(\infty)] . \tag{18}$$

Simplifying the above equation leads to the result that:

$$\mathrm{E}[P(\infty)] = S , \tag{19}$$

which can be verified for all $i = 1, \ldots, r$ as follows:

$$\begin{aligned} \mathrm{E}[p_i(\infty)] &= (1-\lambda)s_i \sum_{j=1}^{r} \mathrm{E}[p_j(\infty)] + \lambda \mathrm{E}[p_i(\infty)] && (20) \\ &= (1-\lambda)s_i + \lambda \mathrm{E}[p_i(\infty)] && (21) \\ \Rightarrow \mathrm{E}[p_i(\infty)](1-\lambda) &= (1-\lambda)s_i , && (22) \end{aligned}$$

indicating that $\mathrm{E}[p_i(\infty)] = s_i$, and proving the theorem. □

Theorem 3. Consider $P(n)$, the estimate of the multinomial distribution S at time 'n', which is obtained by Eqs. (3) and (4). Then, all the non-unity eigenvalues of $\mathbf{M}$ are exactly λ, and therefore the convergence rate of P is fully determined by λ.

Proof. The rate of convergence of the vector difference equation was analyzed by finding the eigenvalues of $\mathbf{M}$, namely $\xi_1, \xi_2, \ldots, \xi_r$. $\mathbf{M}$ can be decomposed as follows:

$$\mathbf{M} = \mathbf{\Phi} \mathbf{\Lambda} \mathbf{\Phi}^{-1} , \tag{23}$$

where:

$$\mathbf{\Phi} = \left[\begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \middle\| \begin{bmatrix} -\frac{s_2}{s_1} \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \middle\| \begin{bmatrix} -\frac{s_3}{s_1} \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \middle\| \cdots \middle\| \begin{bmatrix} -\frac{s_r}{s_1} \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \right], \tag{24}$$

contains the eigenvectors of $\mathbf{M}$ and $\mathbf{\Lambda} = diag(1, \lambda, \lambda, \ldots, \lambda)$, which includes the eigenvalues of $\mathbf{M}$. Thus, $\xi_1 = 1$, and $\xi_i = \lambda$ for $i = 2, \ldots, r$. Considering the fact that the second largest eigenvalue determines the rate of convergence of the matrix defining the vector difference equation, it can be concluded the second claim is true since λ is an eigenvalue of multiplicity $r - 1$. Therefore, a small value of λ leads to fast convergence and a large variance and a large value of λ leads to slow convergence and a small variance. □

Theoretically, since the derived results are asymptotic, they are valid only as $n \to \infty$. However[9], in practice, by choosing λ from the interval $[0.9, 0.99]$, the convergence happens after a relatively small value of 'n'. Indeed, if λ is as "small" as 0.9, the variation from the asymptotic value will be in the order of 10^{-50} after 50 iterations. In other words, the SLWE will provide good results even if the distribution parameters change after 50 steps. The experimental results in [52] demonstrated a good performance achieved by using the SLWE in dynamic environments.

We conclude this section by briefly citing some more recent extensions to the SLWE. Yazidi *et al.* [70] resorted to *discretizing* the probability space [1, 44, 50, 66], and performing a controlled random walk on this discretized space. It is well known in the field of LA that discretized schemes[10] achieve faster convergence speed than their continuous counterparts [1, 49]. By virtue of discretization, the estimator proposed in [70] realizes fast adjustments of the running estimates by performing "jumps", and it is thus able to robustly and quickly track changes in the parameters of the environment's distribution. Adapting the discretized estimator proposed in [70] for the current research problem, is still open.

We shall now explain how we can apply the SLWE for the online classification of data streams based on our new model of computation.

3 Online Classification Using the SLWE

In traditional ML learning and particularly supervised learning, the training phase is performed in an *offline* manner, i.e., the training set is used to learn the stochastic properties of each class. Subsequently, the learned model is deployed and used to classify unlabeled data instances that appeared in the form of data streams.

In many real life applications, it is not possible to analyze the stochastic model of the classes in an *offline* manner because of their dynamic natures. In fact, *offline* classifiers assume that the entire set of training samples can be accessed. However, in many real life applications, the entire training set is not available either because it arrives gradually or because it is not feasible to store it so as to infer the model of each class. Consequently, one is forced to constantly make the classifier update the learning model using the newly-arriving training samples.

[9] In the field of LA, the rate of convergence and accuracy of the schemes are determined by the learning parameter, which in this case is λ. For the family of Absolutely Expedient schemes like the L_{RI}, the accuracy increases as the parameter, λ, increases. The speed of convergence simultaneously decreases with λ. In most real-life applications, the value of λ is set to be between 0.9 and 0.99.

[10] Historically, the concept of discretizing the probability space was pioneered by Thathachar and Oommen in their study on Reward-Inaction LA [66]. Since then, it has catalyzed significant research in the design of discretized LA [1, 27, 28, 44, 50].

We present a novel *online* classifier scheme, that is able to update the learned model using a single instance at a time. Our goal is to predict the source of the arriving instances as accurately as possible, with the added complexity that the testing patterns can subsequently be considered as training patterns. To achieve this, we first define the general structure of the *Online* classifier, and then provide some experimental results on synthetic two-class binomial and multinomial datasets in the next section.

Online classifiers deal with data streams, in which the labeled and unlabeled samples are mixed. Therefore, the training, testing and deploying phases of the *online* classifiers are interleaved as they are applied to these types of data streams. This fascinating avenue is our domain, and we have investigated the performance of SLWE-based classifiers to this new scheme.

Devising a classifier that deals with the data streams generated from non-stationary sources poses new challenges, since the probability distribution of each class might change even as new instances arrive. An important characteristic of our model for *online* learning is that the actual source of the data is discovered shortly after the prediction is made, which can then be used to update the learned model. In other words, our *online* algorithm includes three steps, which are described in Algorithm 1. First, the algorithm receives a data element. Using it and the currently-learned model, the classifier predicts the source of that element. Finally, the algorithm receives the true class of the data, which is then used to update and refine the classification model.

In order to perform the *online* classification of the instances, we need to obtain the *a posteriori* probability of each class. Analogous to the previous classification models, we assign a label to the new unlabeled data element by comparing the obtained *a posteriori* probabilities and the estimated probability from the unlabeled test stream. Finally, after receiving the true label of the instance, the *a posteriori* probabilities are updated using the algorithm explained in Eqs. (3) and (4).

In this classification model the training phase and the testing phase are performed simultaneously, and so the problem can be described as follows. We are given the stream of unlabeled samples generated from different sources arriving in the form of a (Periodically Switching Environment) PSE, in which, after every T time instances, the data distribution and the source of the data might change. In this case, in addition to the switching of the source of the data elements, the probability distribution of each source also possibly changes at random time instances. The aim of the classification is to predict the source of the elements arriving at each time step by using the information in the detected data distribution, and also the information of current model of each class. In the *online* classification model, shortly after the prediction is made, the actual class label of the instance is discovered, which can be utilized to update the classification model to be used by the SLWE updating algorithm.

The process is formalized in Algorithm 1. The notation that we use requires some explanation. Consider Eqs. (3) and (4), in which $p_i(n)$ represents the probability of $x = i$ at time 'n'. Since we are now dealing with the vector of features representing the multinomial variable, the same quantity, $p_i(n)$, represents the probability of dimension 'i' in Algorithm 1. When we now consider the stream of observations, we should introduce a third "degree of freedom", i.e., the time instant when the observation arrives. However, by introducing this degree of freedom for an unending, possibly infinite, sequence, we will unnecessarily confuse matters. This is because the time instant at which the observation *arrives* is not crucial. Rather, what is crucial, is the testing sample whose identity is known after a certain time delay, as per this model of computation. Thus, we have referred to the next incoming sample by the generic symbol, x, as it leads to no additional confusion.

The metric that we have used for comparing dissimilarity is the Kullback-Leibler (KL) divergence, where if

$U = [u_1, \ldots, u_d]^T$, and $V = [v_1, \ldots, v_d]^T$, the KL divergence is:

$$KL(U||V) = \sum_i u_i \log_2 \frac{u_i}{v_i}. \tag{25}$$

Algorithm 1 Online Classification Algorithm

$X \leftarrow$ data stream for classification
$\hat{S} \leftarrow$ initialize *a priori* probabilities for each class
while there exists an instance $x \in X$ **do**
    Step 1. Receiving data:
    The model receives the unlabeled sample
    for all dimensions d of x **do**
        $p_i(n) \leftarrow$ Estimate the probability p_i using the SLWE's equations, Eqs. (3) and (4)
    end for
    Step 2. Prediction:
    $P(n) \leftarrow \{p_1(n), p_2(n), \ldots, p_d(n)\}$
    $\hat{\omega} \leftarrow \arg_i \min KL(\hat{S}_i||P(n))$
    Step 3. Updating the model:
    After some delay, t_d, the true category of the instance x is received
    $\omega \leftarrow$ true class of x
    Update *posterior* probabilities $\hat{S}$ using ω and the SLWE's equations, Eqs. (3) and (4)
end while

4 Experimental Results

In this section, we present the results of this classifier on synthetic data. To assess the efficiency of the SLWE-based *online* classifier, we applied it for two-dimensional and multi-dimensional binomial and multinomial randomly generated data streams. The rigorous empirical analysis was done for two classes and multiple classes. Our results demonstrate the applicability of our method on synthetic datasets, and proves the advantages of the introduced scheme for both binomial and multinomial non-stationary distributions. We also classified and compared the data streams' elements by following the traditional MLE with a sliding window, whose size is also selected randomly.

4.1 Binomial Data Stream

In the case of synthetic d-dimensional binomial data with *two* different non-stationary categories, the classification problem was defined as follows. We are given a stream of bit vectors that were drawn from *two* different periodically switching sources (classes), say, S_1 and S_2.

In order to perform the *online* classification, we assumed that we were provided with a small amount of labeled instances before the arrival of the data stream, which was used to obtain the *a posteriori* probability vector of '0' for each class, say, $\hat{S}_{11}$ and $\hat{S}_{12}$. To perform the class labeling of the newly-arriving unlabeled element, the SLWE estimated the probability of '0', which we refer to as $P_1(n)$. This probability allowed us to predict the source that the new instance belonged to. The probability vector that had the minimum distance to the estimated probability vector of '0', was chosen as the label of the observed element. The probability distances between the learned SLWE probabilities, $P_1(n)$, and the SLWE estimation during training, $\hat{S}_{11}$ and $\hat{S}_{12}$, were computed using the KL measure.

Thus, based on the SLWE classifier, the n^{th} element read from the test set was assigned to class S_j, where

$$j = \arg_i \min KL(\hat{S_{1i}}||P_1(n)), \tag{26}$$

where the KL divergence is given in Eq. (25).

Thus, based on the SLWE classifier, the n^{th} element read from the test set was assigned to the class, which had the minimum distance to the estimated probability, $P_1(n)$. After some delay, t_d, at time $n + t_d$, we assume that have received the true category of the n^{th} instance, which was then used to update the corresponding probability learned in the training phase. The true class label for the n^{th} instance was read and added to the previously-trained model by updating the probabilities of the corresponding class, based on the updating algorithm in Eqs. (3) and (4). For example, Fig. 1 demonstrates a sample of an individual one-dimensional non-stationary class with four concept drift points. As can be seen, the probability of '0', $\hat{S}_{11}$, was estimated using training samples arriving with delay of $t_d = 10$, by both the SLWE and the MLEW methods. For the SLWE, the value of λ was set to be 0.9, and the size of the window for MLEW was 80. It is evident that the SLWE was superior in tracking the probability to the MLEW method, and that it adjusted the corresponding probability at the concept drift points more quickly, which led to a better classification performance.

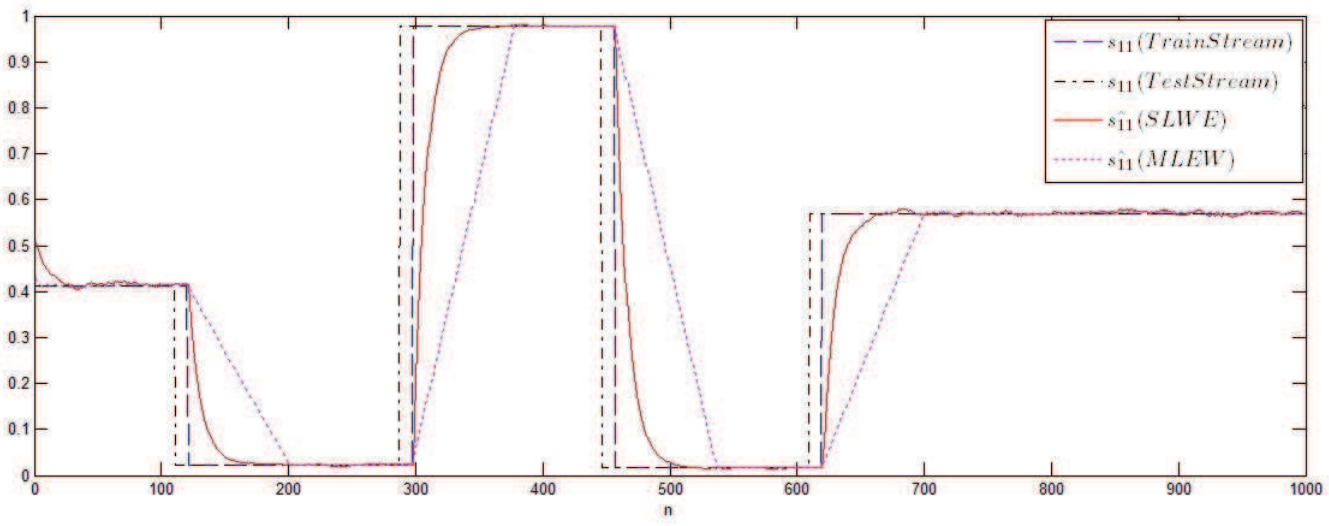

Figure 1: Plot of the averages for the estimates of s_{11}, obtained from the SLWE and MLEW at time n, using the available training samples that arrived with the delay of $t_d = 10$. The stochastic properties of each class switched four times at randomly selected times.

Since our aim here is to confirm the efficiency of the SLWE-based classifier for binomial datasets, which were generated from two non-stationary sources, the classification problem was tested extensively for numerous distributions, but only a subset of the final results are cited here. To carry out the experiments, various datasets were generated. However, in order to generate non-stationary classes, the probabilities were changed several times randomly. It should be noted that in each period of length T, the probability of each class was consistent, and the concept drift could only occur at the end of the specified periods. For the experiments we report, we investigated datasets that were generated for 40 periods with different periodicities from only *two* different binomial sources, in

which the probabilities of the distributions of each source changed at several randomly-selected switching points. Fig. 2 displays an example of the probability of '0' generated from the above mentioned sources, where, $T = 100$, and the concept drift occurred at time instances of 600 and 2,200.

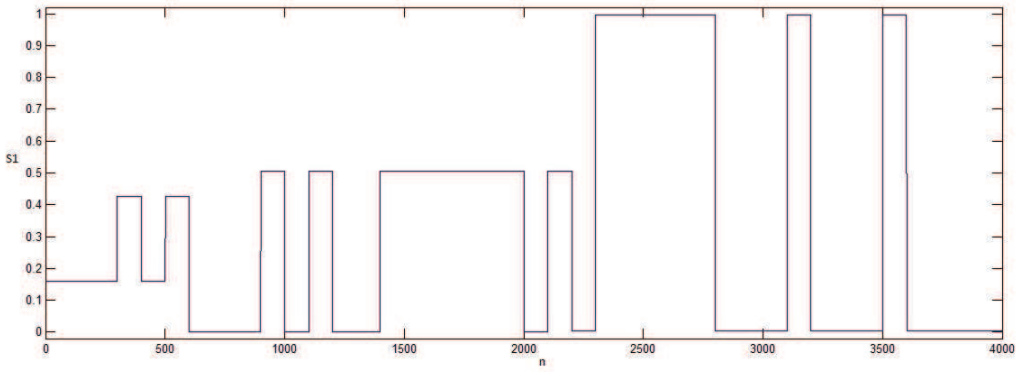

Figure 2: An example of the true underlying probability of '0', S_1, for a one-dimensional binary data stream. The data was generated using two different sources in which the period of switching was 100, and the stochastic properties of the classes switched two times.

In the first set of experiments we considered a scenario, in which a one-dimensional dataset was generated from *two* different binomial sources and the probabilities of the underlying distributions of the classes changed four times. We tested and trained the classifiers in the periodic environment in which the period of switching, T, was either fixed or chosen randomly. We performed the experiment on different test sets with various *known* periods (although this was unknown to the classifiers), $T = 50, 100, \ldots, 500$, and for each value of T, 100 experiments were done and the average value of the accuracies was reported as the result. For the MLEW, the window size W was computed as the nearest integer of a randomly generated value obtained from a uniform distribution $U[\frac{T}{2}, \frac{3T}{2}]$, and the last observed W labeled elements were used to train the MLE model at each time instance.

For the particular results which we report here, the specific values of S_{11} and S_{12} for the one-dimensional dataset were randomly set to be 0.4735 and 0.1221 respectively, which switched five times to the random values of {0.8664, 0.4981, 0.6635, 0.3153} and {0.3374, 0.1397, 0.8494 , 0.4523}, respectively. The concept drift times were set randomly, and were assumed to be unknown to the classifiers. The results obtained are provided in Table 1, from which we see that the SLWE-based classifier was able to detect the concept drift and adapt the learning model to new elements uniformly better than the MLEW. For example, when $T = 200$, the MLE-based classification resulted in the accuracy of 0.7667, while the SLWE-based classification performed significantly better with the accuracy of 0.8695. From the last row in the table, we observe that the results of the classification in periodic environments with a varying T chosen randomly from $[50, 150]$ were also similar to the fixed $T = 100$ case, as the classifier achieved the accuracy of 0.8551 and 0.8532 in the first and second environments, respectively, while the corresponding accuracies of the MLEW were only 0.7633 and 0.7648 respectively. One observes the absolute superiority of the SLWE-based classifier.

We repeated the above procedure on different 2-class binomial datasets with different dimensionalities. These sets were generated randomly based on random vectors with different random distribution probabilities involving 2, 3 and 4 dimensions. The results obtained are shown in Tables 2-4, and as can be seen, the advantage of the SLWE

T	MLEW	SLWE
50	0.7231	0.8125
100	0.7648	0.8532
150	0.7624	0.8556
200	0.7667	0.8695
250	0.7616	0.8834
300	0.7646	0.8731
350	0.7589	0.8667
400	0.7532	0.8810
450	0.7689	0.8765
500	0.7691	0.8673
Random $T \in (50, 150)$	0.7633	0.8551

Table 1: The ensemble results for 100 simulations obtained from testing binomial classifiers which used the SLWE (with $\lambda = 0.9$) and the MLEW for classifying one-dimensional data streams generated by two non-stationary different sources. The reader should observe the absolute superiority of the SLWE-based classifier.

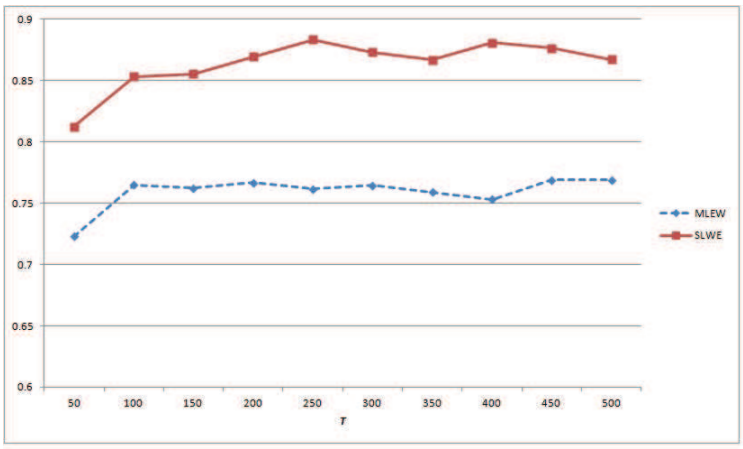

Figure 3: Plot of the accuracies of the MLEW and the SLWE binomial classifiers on a one-dimensional dataset generated from two non-stationary sources with different switching periods, as described in Table 1.

over the MLEW is consistent. For example, when $T = 250$, the MLEW achieved the accuracy of 0.7616, while the SLWE resulted in the accuracy of 0.8834. Similarly for the 2-dimensional data, the MLE-based classifier resulted in the accuracy of 0.7544 and the SLWE achieved significantly better results with the accuracy of 0.9705.

In the final set of experiments, we performed a detailed analysis of the SLWE-based *online* classifier relative to the dimension of the datasets, and analyzed the performance of the SLWE-based classifier on datasets with different dimensions. The classification procedure explained above was repeated 10 times over different datasets with fixed dimensions, and the ensemble average of the accuracies was obtained over these datasets. In each experiment, the classifiers were tested on a periodic environment and stochastic property of each class was changed four times. For each value of T, an ensemble of 100 experiments was performed. The obtained results are shown in Figures 7 and 8. It is evident that for the same switching period, when the dimensionality of the dataset is higher, the classifiers can process the data more efficiently. For example, in the case of $T = 150$ the SLWE-based classifier resulted in the average accuracy of 0.8094 over several different one-dimensional datasets, while with more useful information in 4-dimensional datasets, it yielded better results with the accuracy of 0.9519.

T	MLEW	SLWE
50	0.7314	0.9083
100	0.7657	0.9473
150	0.7475	0.9606
200	0.7676	0.9666
250	0.7544	0.9705
300	0.7526	0.9729
350	0.7620	0.9744
400	0.7595	0.9753
450	0.7547	0.9780
500	0.7551	0.9788
Random $T \in (50, 150)$	0.7534	0.9523

Table 2: The ensemble results for 100 simulations obtained from testing binomial classifiers which used the SLWE (with $\lambda = 0.9$) and the MLEW for classifying 2-dimensional data streams generated by two non-stationary different sources.

T	MLEW	SLWE
50	0.7182	0.8798
100	0.7469	0.9237
150	0.7416	0.9400
200	0.7652	0.9495
250	0.7512	0.9494
300	0.7450	0.9537
350	0.7476	0.9580
400	0.7597	0.9581
450	0.7555	0.9611
500	0.7551	0.9612
Random $T \in (50, 150)$	0.7504	0.9267

Table 3: The ensemble results for 100 simulations obtained from testing binomial classifiers which used the SLWE (with $\lambda = 0.9$) and the MLEW for classifying 3-dimensional data streams generated by two non-stationary different sources.

T	MLEW	SLWE
50	0.7145	0.8667
100	0.7496	0.9236
150	0.7491	0.9426
200	0.7671	0.9522
250	0.7544	0.9589
300	0.7574	0.9597
350	0.7515	0.9629
400	0.7506	0.9651
450	0.7557	0.9670
500	0.7498	0.9705
Random $T \in (50, 150)$	0.7512	0.9357

Table 4: The ensemble results for 100 simulations obtained from testing binomial classifiers which used the SLWE (with $\lambda = 0.9$) and the MLEW for classifying 4-dimensional data streams generated by two non-stationary different sources.

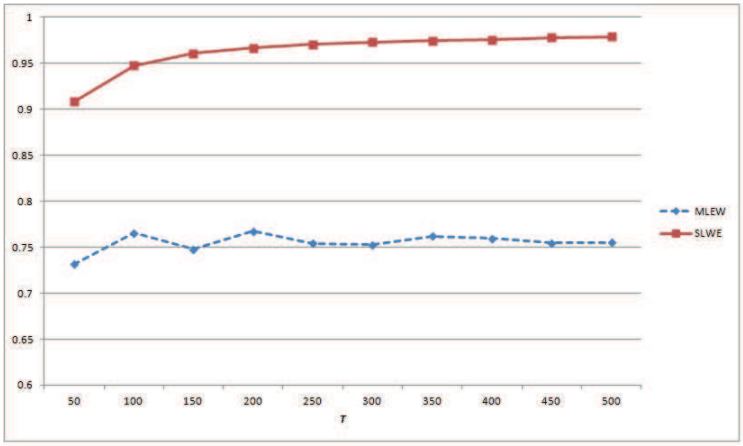

Figure 4: Plot of the accuracies of the MLEW and the SLWE binomial classifiers on a 2-dimensional dataset generated from two non-stationary sources with different switching periods, as described in Table 2.

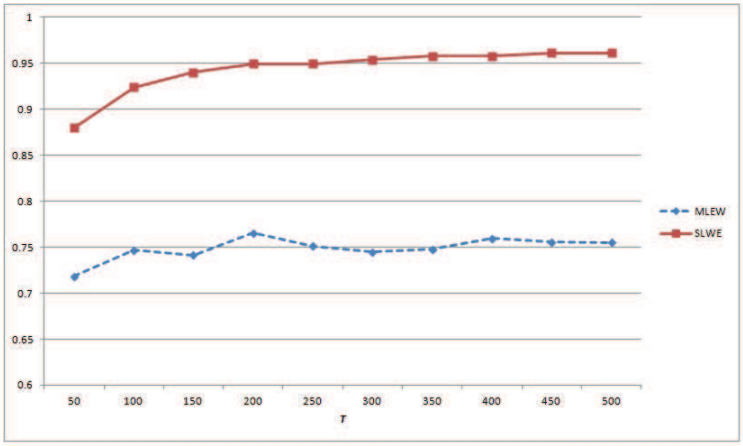

Figure 5: Plot of the accuracies of the MLEW and the SLWE binomial classifiers on a 3-dimensional dataset generated from two non-stationary sources with different switching periods, as described in Table 3.

d	50	100	150	200	250	300	350	400	450	500
1	0.7673	0.7972	0.8094	0.8152	0.8190	0.8235	0.8217	0.8265	0.8246	0.8233
2	0.8290	0.8723	0.8851	0.8951	0.9009	0.9031	0.9055	0.9055	0.9051	0.9058
3	0.8433	0.8883	0.9006	0.9087	0.9121	0.9176	0.9197	0.9216	0.9198	0.9205
4	0.8871	0.9343	0.9519	0.9596	0.9643	0.9671	0.9700	0.9716	0.9733	0.9742

Table 5: The results obtained from testing classifiers which used the SLWE for different binomial datasets with different dimensions, d, which were generated with a fixed switching period of $T = 50, 100, \ldots, 500$, and the stochastic properties of each class switched to different values at four random time instances.

4.2 Multinomial Data Stream

In this section, we report the results for simulations performed for *multinomial* data streams with *two* different non-stationary categories. The multinomial classification problem is a generalization of the binomial case introduced earlier. Here, the classification problem was defined as follows. We are given a stream of unlabeled multinomially distributed random d-dimensional vectors, which take on the values from the set $\{1, \ldots, r\}$, and which are generated

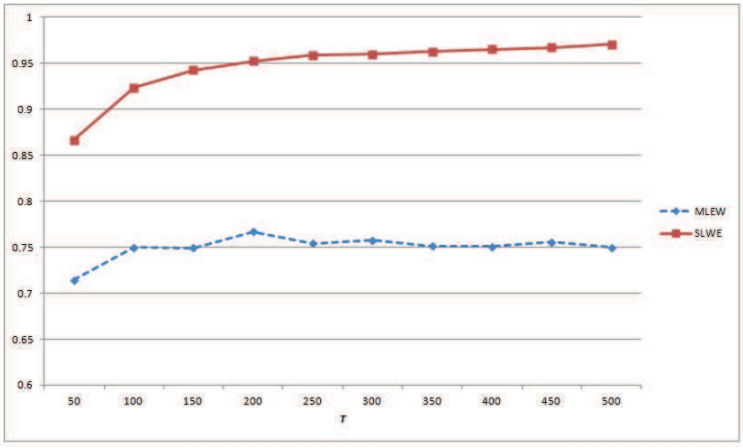

Figure 6: Plot of the accuracies of the MLEW and the SLWE binomial classifiers on a 4-dimensional dataset generated from two non-stationary sources with different switching periods, as described in Table 4.

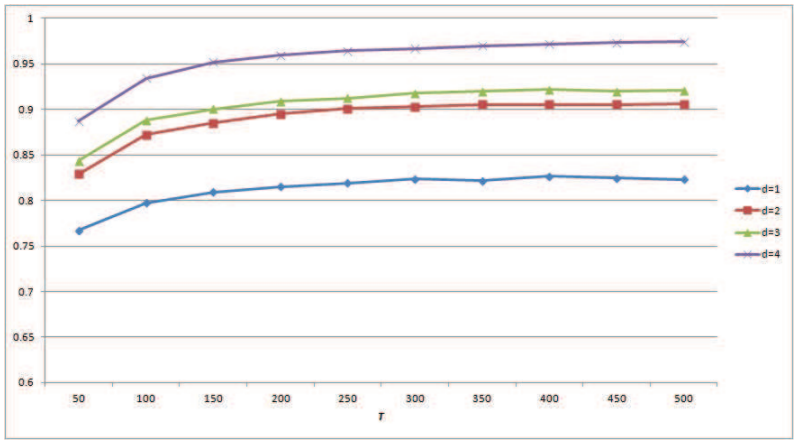

Figure 7: Plot of the accuracies of the SLWE classifier for different binomial datasets with different dimensions, d, over different values of the switching periodicity, T. The numerical results of the experiments are shown in Table 5.

from *two* different periodically switching sources (classes), say, S_1 and S_2. Each class was characterized with probability values, S_{i1}, S_{i2}, which demonstrate the probability of the value 'i', where $i \in \{1, \ldots, r\}$. In this case, similar to the binomial case, the probabilities of the distributions of each class could possibly change with time as the data stream continued to appear.

Analogues to the binomial case, the multinomial data stream classification started with the estimation of the *a priori* probability of each possible value of 'i', in all the 'd' dimensions, for each class 'j' from the available labeled instances, which we refer to as $\hat{S}_{ij}$. To assign a label to the newly arriving unlabeled element, the SLWE estimated the probabilities of each possible value 'i', in all the 'd' dimensions, from the unlabeled instances, which we refer to as $P_i(n)$. Thereafter, these probabilities were used to predict the class label that a new instance belonged to class 'j', with the probability vector of the $\hat{S}_j = \{\hat{S}_{1j}, \hat{S}_{2j}, \ldots, \hat{S}_r\}$, that had the minimum distance to the estimated probability of $P = \{P_1(n), P_2(n), \ldots, P_r(n)\}$, was chosen as the label of the observed element. The distances between the learned SLWE probabilities, $P_i(n)$, and the SLWE estimation during training, $\hat{S}_{ij}$ were again computed using the KL divergence measure, using Eq. (25).

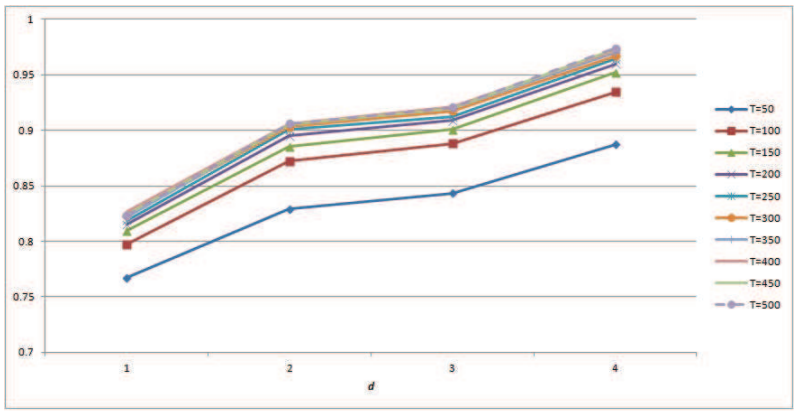

Figure 8: Plot of the accuracies of the SLWE classifier for different binomial datasets involved data from two non-stationary classes. Each dataset was generated with a different switching period, T, and a different dimensionality, d. The numerical results of the experiments are shown in Table 5.

Thereafter, after some delay, t_d, at time $n + t_d$ the algorithm received the true class of the n^{th} instance and used it to refine and update the true class probabilities. The true value of the category for the n^{th} instance was read and added to the previously trained model by updating the probability of the corresponding class based on the updating algorithm in Eqs. (3) and (4).

The classification procedure explained above was performed on multinomial data streams generated from two different classes where the probability of the distributions of each class switched four times. For the results which we report, each element of the data stream could take any of the four different values[11], namely 1, 2, 3 or 4. The specific values of S_{i1} and S_{i2}, were changed and set to random values four times at random time instances, which were assumed to be unknown to the classifiers. The results are shown in Table 6, and again, the uniform superiority of the SLWE over the MLEW is noticeable. For example, when T=100, the MLEW-based classifier yielded an accuracy of only 0.7443, but the corresponding accuracy of the SLWE-based classifier was 0.8012. We also notice that the results of the classification in periodic environments with a varying T chosen randomly from $[50, 150]$ were also similar to the fixed $T = 100$ case, as the classifier achieved the accuracy of 0.8092 and 0.8012 in the first and second environments, respectively. The results also show that the SLWE-based algorithm handles the concept drift and provides a comparatively remarkable performance considering the problem at hand.

The experiment explained above was repeated on different 2-class multinomial datasets with different dimensionalities. These sets were generated randomly based on random vectors with different random distribution probabilities involving 2, 3 and 4 dimensions and each element could take on four different values. The results obtained are shown in Tables 7-9, from which we see that classification using the SLWE was uniformly superior to classification using the MLEW. For example, for the 2-dimensional data, when $T = 250$, the MLE-based classifier resulted in the accuracy of 0.7544 and the SLWE achieved significantly better results with the accuracy of 0.9706 – *which by al metrics is remarkable*. Here the accuracy of the classifier, similar to the binomial case, increased with the dimensionality of the datasets as the classifiers could process the data more efficiently. Thus, in the case of $T = 150$ the SLWE-based *online* classifier resulted in the average accuracy of 0.9181 over several different two-dimensional datasets, while with more useful information in 4-dimensional datasets, it yielded better results with the accuracy of 0.9569.

[11] The experiments were done for numerous other settings. We present here a subset of the results in the interest of space and brevity.

T	MLEW	SLWE
50	0.6786	0.7582
100	0.7443	0.8012
150	0.7476	0.8153
200	0.7595	0.8197
250	0.7514	0.8282
300	0.7509	0.8367
350	0.7587	0.8322
400	0.7598	0.8354
450	0.7635	0.8344
500	0.7574	0.8387
Random $T \in (50, 150)$	0.7523	0.8092

Table 6: The ensemble results for 100 simulations obtained from testing multinomial classifiers which used the SLWE (with $\lambda = 0.9$) and the MLEW for classifying one-dimensional data streams generated by two non-stationary different sources.

T	MLEW	SLWE
50	0.6913	0.8560
100	0.7402	0.9082
150	0.7463	0.9333
200	0.7480	0.9393
250	0.7478	0.9430
300	0.7397	0.9474
350	0.7486	0.9463
400	0.7525	0.9535
450	0.7554	0.9505
500	0.7518	0.9559
Random $T \in (50, 150)$	0.7436	0.9241

Table 7: The ensemble results for 100 simulations obtained from testing multinomial classifiers which used the SLWE (with $\lambda = 0.9$) and the MLEW for classifying 2-dimensional data streams generated by two non-stationary different sources.

T	MLEW	SLWE
50	0.6912	0.8798
100	0.7422	0.9384
150	0.7596	0.9576
200	0.7490	0.9657
250	0.7630	0.9709
300	0.7609	0.9743
350	0.7515	0.9772
400	0.7485	0.9797
450	0.7663	0.9814
500	0.7570	0.9821
Random $T \in (50, 150)$	0.7504	0.9328

Table 8: The ensemble results for 100 simulations obtained from testing multinomial classifiers which used the SLWE (with $\lambda = 0.9$) and the MLEW for classifying 3-dimensional data streams generated by two non-stationary different sources.

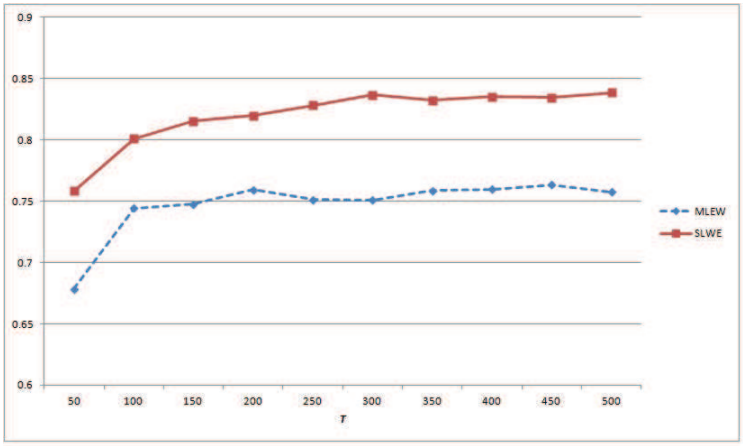

Figure 9: Plot of the accuracies of the MLEW and the SLWE multinomial classifiers on a one-dimensional dataset generated from two non-stationary sources with different switching periods, as described in Table 6.

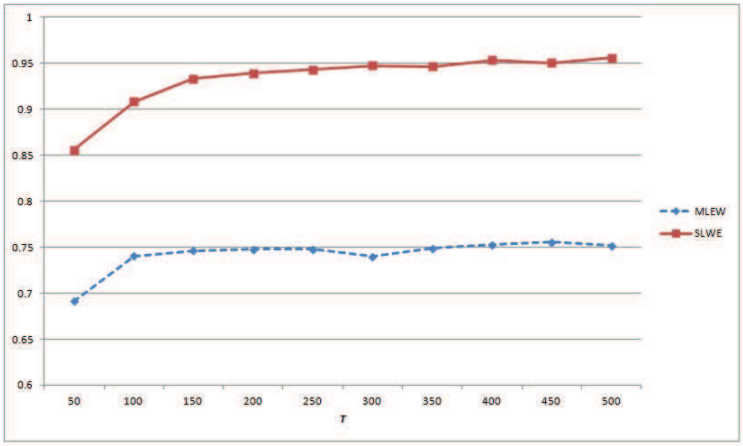

Figure 10: Plot of the accuracies of the MLEW and the SLWE multinomial classifiers on a 2-dimensional dataset generated from two non-stationary sources with different switching periods, as described in Table 7.

5 Real Life Applications

5.1 A Language-based Application

The solution that we have proposed is not only theoretical in nature, as was explained in the introduction. Rather, as mentioned in the introduction, it is, in actuality, the model that is encountered in most real-life applications, for example, in medical diagnosis. Besides, this current model presents a hybrid of traditional statistical PR, and the so-called "Sequential" PR. Our goal, in this section, is to substantiate the general model and the solution proposed, for real-life applications[12]. Indeed, we have also utilized it in a real-life language detection domain involving the languages German and Chinese, where the respective texts were randomly generated by the following

[12]The Referees of the original version of the paper had requested a section that deals with the real-life applications of our scheme. Since data for the model of computation introduced in this paper was not readily available, we entered into a sequence of mails with the two experts in the field, who suggested a method for obtaining data for this model of computation by *extracting* it from the more general, readily available, data for NSEs. Their advice made this section possible and so we record our gratitude. Indeed, we are extremely grateful to the these experts for their advice.

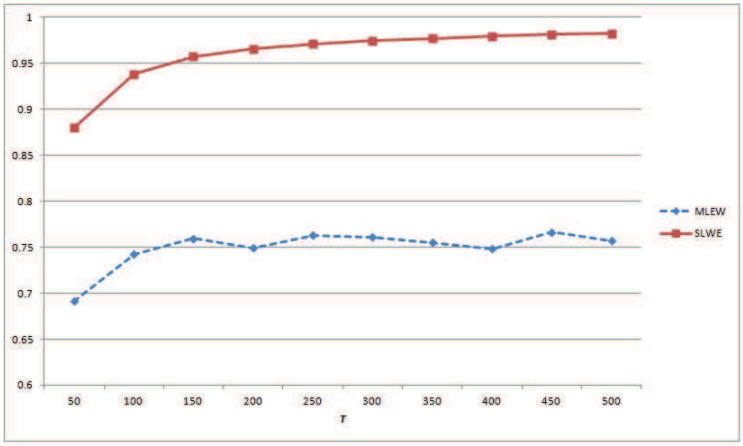

Figure 11: Plot of the accuracies of the MLEW and the SLWE multinomial classifiers on a 3-dimensional dataset generated from two non-stationary sources with different switching periods, as described in Table 8.

T	MLEW	SLWE
50	0.6906	0.8847
100	0.7535	0.9402
150	0.7557	0.9592
200	0.7532	0.9669
250	0.7550	0.9730
300	0.7531	0.9760
350	0.7522	0.9785
400	0.7460	0.9818
450	0.7572	0.9817
500	0.7562	0.9839
Random $T \in (50, 150)$	0.7526	0.9512

Table 9: The ensemble results for 100 simulations obtained from testing multinomial classifiers which used the SLWE (with $\lambda = 0.9$) and the MLEW for classifying 4-dimensional data streams generated by two non-stationary different sources.

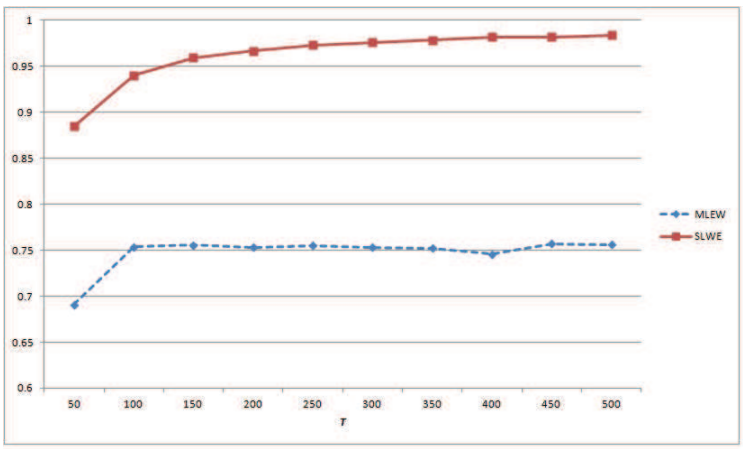

Figure 12: Plot of the accuracies of the MLEW and the SLWE multinomial classifiers on a 4-dimensional dataset generated from two non-stationary sources with different switching periods, as described in Table 9.

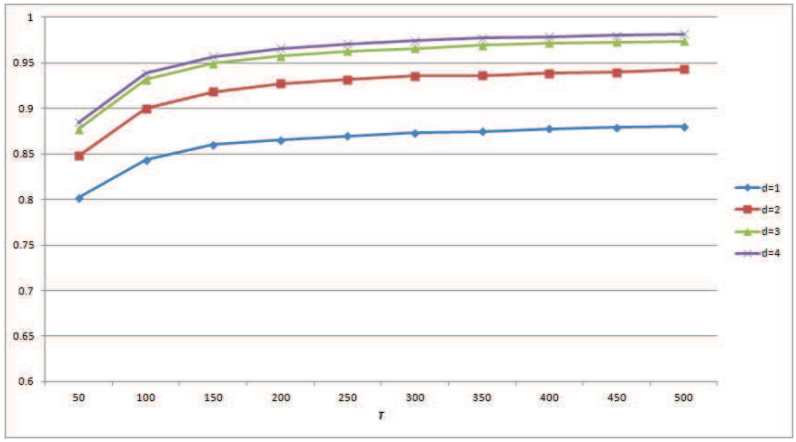

Figure 13: Plot of the accuracies of the SLWE classifier for different multinomial(r=4) datasets with different dimensions, d, over different values of the switching periodicity, T. The numerical results of the experiments are shown in Table 10.

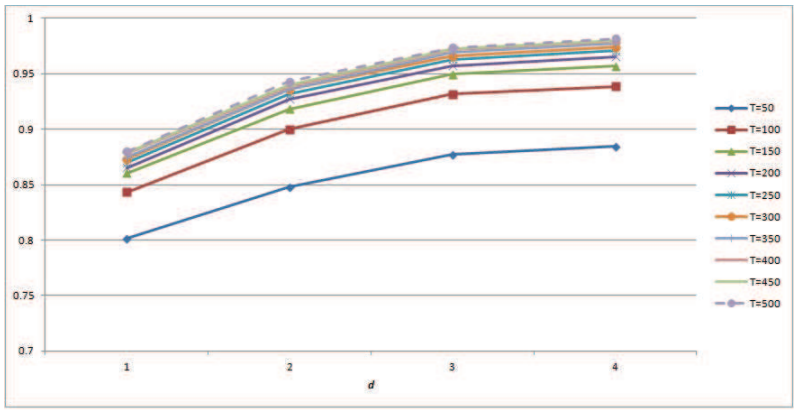

Figure 14: Plot of the accuracies of the SLWE classifier for different multinomial datasets involved data from two non-stationary classes. Each dataset was generated with a different switching period, T, and a different dimensionality, d. The numerical results of the experiments are shown in Table 10.

d	50	100	150	200	250	300	350	400	450	500
1	0.8016	0.8432	0.8604	0.8653	0.8699	0.8730	0.8748	0.8778	0.8790	0.8798
2	0.8479	0.8999	0.9181	0.9273	0.9318	0.9357	0.9359	0.9387	0.9397	0.9430
3	0.8770	0.9316	0.9496	0.9574	0.9629	0.9660	0.9692	0.9718	0.9726	0.9737
4	0.8844	0.9387	0.9569	0.9656	0.9707	0.9741	0.9772	0.9788	0.9801	0.9817

Table 10: The results obtained from testing classifiers which used the SLWE for different multinomial datasets with different dimensions, d, which were generated with a fixed switching period of $T = 50, 100, \ldots, 500$, and the stochastic properties of each class switched to different values at four random time instances.

tool: `http://generator.lorem-ipsum.info/_chinese`. A similar real-life experiment was reported in [72].

To render the problem non-trivial, the text from both languages was scrambled in a random manner. The scrambled text was then transformed into a series of bits where each character in the text corresponded to 8 bits according to the ASCII code. This series of bits was presented to our algorithm where the aim of the exercise was to track the occurrence of the bit '1' which served as the recognizing feature to subsequently achieve the language

classification. We attempted to execute the classication of the corresponding piece of text with a delay of t_d of 8 bits. It is worth mentioning that the classification problem is hard in itself because the probability for the occurrence '1' approached 0.5 for both languages, and the difference was almost unobservable.

We ran the experiment for $3T$ characters, where in the first T block we fed the algorithm with German text. Then in the next T block we used Chinese, and in the last block we again used German. T thus denotes the period of change and is a variable that took three values in the experiments reported, namely, 50, 100 and 200.

We compared the accuracy of the MLEW for different values of the window size, W, as depicted in Figure 15, while the SLWE was executed for different values of the update parameter λ, as illustrated in Figure 16.

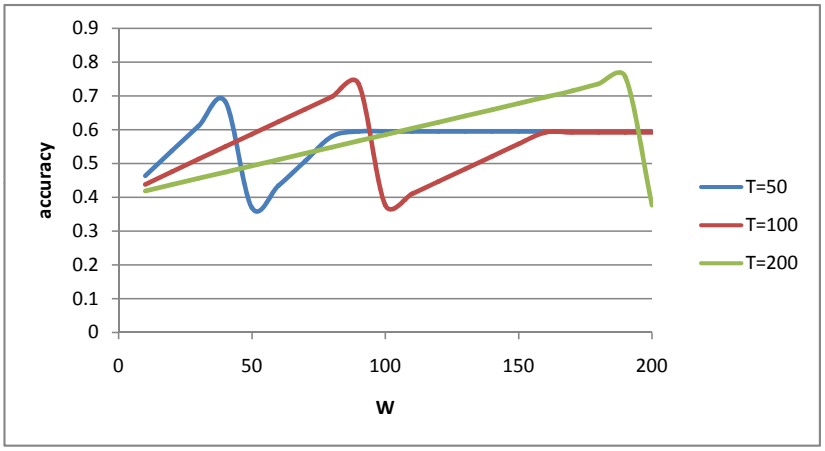

Figure 15: Plot of the classification accuracy of of the MLEW classifier for different values of T and size of the window, W, for the real-life application.

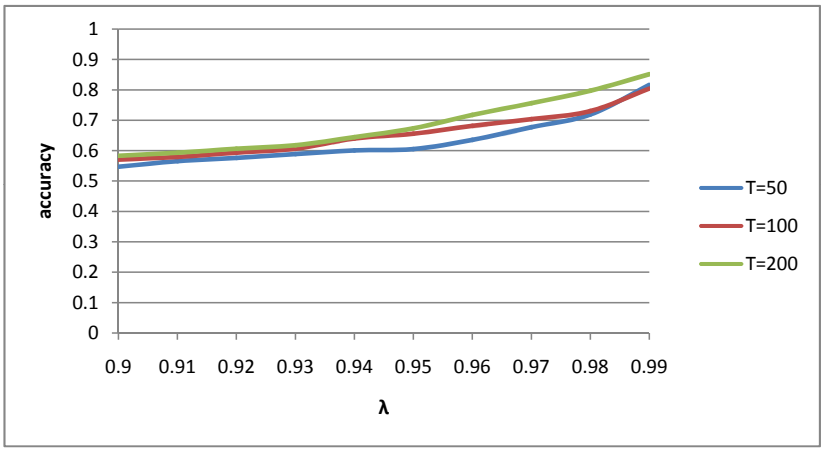

Figure 16: Plot of the classification accuracy of the SLWE classifier for different values of T and the update parameter, λ, for the real-life application.

The most important observation that we can make from this experiment is that the SLWE provided a high accuracy for $\lambda = 0.99$, which was *independent* of T. On the other hand, the performance of the MLEW was dependent on the value of T. Clearly, the SLWE does not suffer from the "parameter tuning" problem associated

with the MLEW. In fact, the quality of the estimate of the MLEW in a dynamic environment depended entirely on the window size when it was compared to the value of the switching period, T. This result clearly demonstrates the superiority of the SLWE for this real-life application domain.

From Figure 15 we see that the highest accuracy was when the window was around half of the switching period, i.e, $w \approx T/2$. Such a window size seemed to yield a good balance between the accuracy and the variance. Further, the delay was negligible when compared to the switching period.

To detail another case, consider the scenario when the window size W was approximately twice the switching period, i.e., $T \approx 2w$. In this case, the MLEW accuracy seemed to stagnate for all values of T to a quantity around 0.59. The possible qualitative explanation that we infer is that when the classifying samples are arriving from a certain class, the sliding window gets saturated with approximately equal numbers of samples from both classes, i.e, around W samples from each class. The presence of this mixture of samples from both classes in the sliding window deteriorates the computed estimate, and consequently reduces the classification accuracy.

From Figure 15, we observe that the worst performance for the MLEW is when the size of the sliding window approaches the length of the switching period, T. In fact, in such a scenario, the classification model is being maintained and updated based on samples from a certain class, while the currently-arriving samples, based on which the estimate is being computed, are from the opposite class! Such a scenario again clearly depicts the superiority of the SLWE over the MLEW.

5.2 A Temperature-history Application

To render this section concerning the testing for real-life data more comprehensive, we have attempted to achieve the testing on another real-life application[13].

The data set involved the recorded temperature observations from two cities, and the goal of the exercise was to use the non-stationary history to achieve the classification, while attempting, at the same time, to stay within this new model of computation. To obtain the data set, we considered $12,418$ observations of the daily temperature from January 1, 1982 to January 31, 2016 from two different countries, namely the United Kingdom[14], and Singapore (Changi)[15]. We again utilized the SLWE and the MLEW as in the previous experiments, and the settings of the experiments, and the respective results, are explained below.

For the SLWE, the value of λ was set to be 0.9, and the size of the window for the MLEW was 80. In order to obtain a multinomial stream, we resorted to invoking a uniform-bin histogram. The bins were uniform in the interval between the minimum and the maximum values in the stream. In our first experiment, we used 30 bins for the MLEW and the SLWE, and thus, we had features with a dimension of 30, as catalogued in Table 11.

To obtain non-stationary data, we randomly switched the underlying distribution using a switching probability of p_s. Thus, at each time instant, the underlying distribution switched to an alternate distribution a probability p_s. By virtue of the above, we see that the switching instances followed a geometric distribution with a parameter p_s, whence the probability of a switch after k instants was $p_s(1-p_s)^{k-1}$. Since we wanted to test the classification using this current model, we assumed a delay in knowing the true class identities, and to achieve this, we used the same delay that was used previously, i.e., $t_d = 10$. Since the switches occur at random, and in order to smooth the

[13] We are grateful to the Anonymous Referees for requesting testing on a second data set. Doing this set of experiments has been a major undertaking, but it has really improved the quality of the paper. Thank you very much.

[14] https://www.metoffice.gov.uk/climatechange/science/monitoring/ukcp09/

[15] http://www.weather.gov.sg/climate-historical-daily

p_s	MLEW	SLWE
1/500	0.822	0.839
1/1000	0.831	0.857
1/1500	0.838	0.863
1/2000	0.847	0.870
1/2500	0.856	0.879
1/3000	0.861	0.885
1/3500	0.872	0.894

Table 11: The ensemble results obtained for 50 simulations when testing multinomial classifiers for the "Temperature" data set for classifying multi-dimensional data streams generated by two non-stationary different sources. The classifiers used the SLWE (with $\lambda = 0.9$) and the MLEW. The dimension of the data was 30.

p_s	MLEW	SLWE
1/500	0.667	0.683
1/1000	0.672	0.705
1/1500	0.684	0.720
1/2000	0.693	0.737
1/2500	0.701	0.749
1/3000	0.728	0.766
1/3500	0.735	0.777

Table 12: The ensemble results obtained for 50 simulations when testing multinomial classifiers for the "Temperature" data set for classifying multi-dimensional data streams generated by two non-stationary different sources. The classifiers used the SLWE (with $\lambda = 0.9$) and the MLEW. The dimension of the data was 10.

estimates and obtain consistent results, we report the ensemble of 1,000 experiments.

Without belaboring the point (and to avoid repetition) we observe from the results that the SLWE is markedly superior to the MLEW. For example, when $p_s = \frac{1}{2,500}$, the MLEW-based classifier yielded an accuracy of only 85.6%, but the corresponding accuracy of the SLWE-based classifier was as high as 87.9%.

The story would not be complete if we were not able to test the data for different dimensionalities. In typical real-life data, such a testing would not have been possible, but since we created this data from the real-life data, we achieved this by changing the number of bins to 10. The results are shown in Table 12, when the dimension of the data is 10. It is very interesting to note that as we reduced the number of bins, it became increasingly more difficult to differentiate between the data streams. This actually demonstrates the difference between the present results and those given in the original paper that proposed the SLWE [52], in which the authors only worked with *uni*-dimensional data. Indeed, we observed that there was a drop in the classification accuracy as the dimensionality was decreased – which can be seen if we compare the results obtained here with those found in Table 11. For example, when $p_s = \frac{1}{500}$, we achieved an accuracy for the SLWE when the dimension is 30 of 82.2% while the accuracy dropped drastically to only 66.7% when the dimension was reduced to 10.

Figure 17 and Figure 18 depict the results of the experiments corresponding to Table 11 and Table 12.

6 Conclusion

In this paper we have considered the problem of classification when the traditional phases of training and testing are not so clearly well-defined, i.e., where the testing patterns can subsequently be considered as training patterns. This

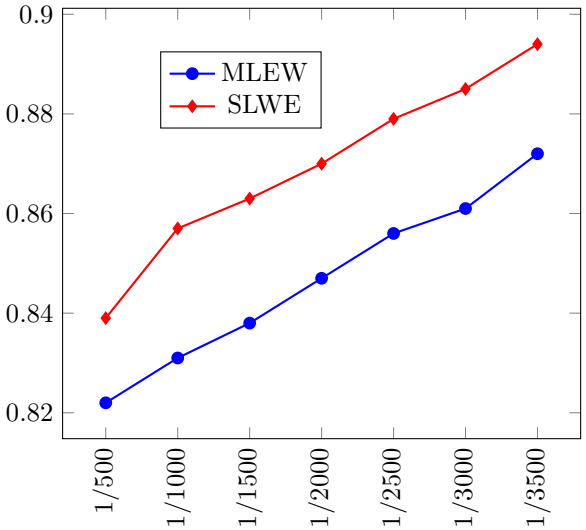

Figure 17: The ensemble results obtained for 50 simulations with the dimension of the data 30 corresponding to Table 11.

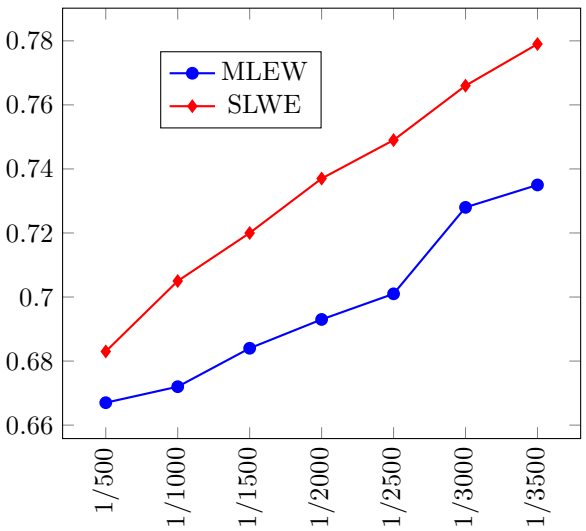

Figure 18: The ensemble results obtained for 50 simulations with the dimension of the data 10 corresponding to Table 12.

paradigm of classification is further complicated because we have assumed that the class-conditional distributions of the classes are time-varying or non-stationary. Here, we consider the scenario when the patterns arrive sequentially in the form of a data stream with potentially time-varying probabilities that change over time for each class. Our proposed *online* classification algorithm was used to perform the training and the testing simultaneously in three phases. In the first phase, the algorithm received a new unlabeled instance. After this, the scheme assigned a label to it based on the distributions' estimated probabilities using the SLWE. Finally, after a few time instances, the algorithm received the actual class of the instance and used it to update the training model by invoking the SLWE updating algorithm. In this way the training and testing phases are almost intertwined. Thereafter, the classification model was adjusted to the new available instances in an *online* manner.

To test our solution, we performed our proposed *online* classification on synthetic two-dimensional and multi-dimensional, binomial *and* multinomial data streams, which were generated from two non-stationary sources of data. In our experiments we used weak estimators characterized by the SLWE method to learn the probability distribution from binomial and multinomial data streams in the NSEs, where the statistical distribution of each class could possibly change with time as more instances became available. Experimental results for both binomial and multinomial random variables demonstrated the efficiency of the SLWE-based *online* classifier as it demonstrated a far superior classification performance on data streams, and it also handled the concept drifts in NSEs. These results indicated the uniform superiority of the SLWE-based classifier over the the classification scheme using a MLE-based sliding window method. Also, from the experiment results we see that when the dimensionality of datasets was higher the SLWE-based classifier could process the data more efficiently and achieved better a accuracy. Experiments conducted on two real-life data sets confirmed these assertions.

The effectiveness of the algorithm and the superiority of the SLWE in both of these cases were demonstrated through extensive experimentation on a variety of datasets. In summary, we list here the conclusions drawn from the experimental results that we obtained in this paper:

- A main advantage of the incremental SLWE-based classifier is that it does not require any assumption about how fast or how often the stream changes. As opposed to this, the sliding window MLWE approach needs some *a priori* knowledge of the systems behavior, in order to consider the window size.

- For the case of online classification, where the probability of each class could switch "unexpectedly", experimental results demonstrated the efficiency of the SLWE-based *online* classifier. These results also indicated that the SLWE-based incremental classifier was still superior to the classification scheme that used a sliding window and the MLE method.

- The results also suggested that the classifier's performance improved with the switching period. Further, it was evident that the accuracy of classification in a periodic environment with random switching period was very close to the case involving the data stream with a fixed switching period, which indicates that the algorithm is efficient no matter how fast or how often the stream would change.

- By examining the results obtained, we can see that when the dimensionality of datasets was higher the SLWE-based classifier achieved a superior accuracy. On the other hand, datasets generated from larger number of classes led to a more complex classification problem, and this degraded the performance of the classifier. Both of these are intuitively appealing conclusions.

In this paper, for all the experiments conducted for synthetic data, we used the constant value of $\lambda = 0.9$ for the updating parameter of the SLWE. One direction for the future work would be that of adjusting this parameter depending on the performance of the classifier at each time instance. In fact, when the performance of classification drops significantly, it can be inferred that a change in the data distribution has occurred. Thereafter, in order to "unlearn" the model, the updating parameter of the SLWE could possibly be increased.

Finally, another avenue for future work could be the development of similar *online* classifiers for other distributions, such as the Gaussian, exponential, gamma, and Poisson. It would thus be interesting to utilize the SLWE estimator to estimate the characteristics of other distributions such as their mean and variance, and to use the analogous classifiers for outlier detection and one-class-classification problems.

It is also pertinent to mention that the SLWE has applications in almost all walks of life where one estimates non-stationary statistics. Indeed, in all such applications, the prior literature also shows that the SLWE is superior to the MLE. We believe that it has huge potential in many other real-life application domains including the well-known pH neutralization process.

References

[1] M. Agache and B. J. Oommen. Generalized pursuit learning schemes: New families of continuous and discretized learning automata. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 32(6):738–749, December 2002.

[2] C. Aggarwal. *Data Classification: Algorithms and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, 2014.

[3] E. Aslanci, K. Coşkun, P. Schüller, and B. Tümer. Detection of regime switching points in non-stationary sequences using stochastic learning based weak estimation method. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 787–792. IEEE, 2017.

[4] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall, Inc., Englewood Cliff, 1993.

[5] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *ACM SIGMOD Record*, volume 19, pages 322–331. Acm, 1990.

[6] M. Bhaduri, J. Zhan, and C. Chiu. A novel weak estimator for dynamic systems. *IEEE Access*, 5:27354–27365, 2017.

[7] S. Bickel and T. Scheffer. Dirichlet-enhanced Spam Filtering Based on Biased Samples. *Advances in Neural Information Processing Systems*, 19:161168, 2007.

[8] A. Bifet. *Adaptive Learning and Mining for Data Streams and Frequent Patterns*. PhD thesis, Departament de Llenguatges i Sistemes Informatics, Universitat Politcnica de Catalunya, Barcelona Area, Spain, 2009.

[9] A. Bifet and R. Gavalda. Kalman Filters and Adaptive Windows for Learning in Data Streams. In L. Todorovski and N. Lavrac, editors, *Proceedings of the 9th Discovery Science*, volume 4265, pages 29–40, 2006.

[10] A. Bifet and R. Gavalda. Learning From Time-changing Data With Adaptive Windowing. In *Proceedings SIAM International Conference on Data Mining*, volume 8, pages 443–448, 2007.

[11] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[12] A. Bouchachia. Fuzzy classification in dynamic environments. *Soft Computing*, 15(5):1009–1022, 2011.

[13] J. Chen and A. Gupta. On Change-Point Detection and Estimation. *Communication in Statistics Simulation and Computation*, 30(3):665–697, 2001.

[14] C. Chiu and J. Zhan. Deep learning for link prediction in dynamic networks using weak estimators. *IEEE Access*, 6:35937–35945, 2018.

[15] E. Cohen and M. Strauss. Maintaining time-decaying stream aggregates. In *Proc. 22nd ACM Symposium on Principles of Database Systems*, 2003.

[16] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006.

[17] C. M. De Oca, D. R. Jeske, Q. Zhang, C. Rendon, and M. Marvasti. A CUSUM Change-Point Detection Algorithm for Non-stationary Sequences with Application to Data Network Surveillance. *The Journal of Systems and Software*, 83:12881297, 2010.

[18] J. Demšar and Z. Bosnić. Detecting concept drift in data streams using model explanation. *Expert Systems with Applications*, 92:546–559, 2018.

[19] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM, 2000.

[20] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 71–80, New York, NY, USA, 2000. ACM.

[21] A. Dries and U. Rückert. Adaptive concept drift detection. *Stat. Anal. Data Min.*, 2(5):311–327, Dec. 2009.

[22] R. Duda, P. Hart, and D. Stork. *Pattern Recognition*. Wiley, second ed. edition, 2000.

[23] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. A survey of classification methods in data streams. In *Data streams*, pages 39–59. Springer, 2007.

[24] J. Gama. *Knowledge Discovery From Data Streams*. Chapman & Hall/CRC, 2010.

[25] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with Drift Detection. In A. Bazzan and S. Labidi, editors, *Advances in Artificial Intelligence SBIA 2004*, volume 3171 of *Lecture Notes in Computer Science*, pages 286–295. Springer Berlin Heidelberg, 2004.

[26] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, Mar. 2014.

[27] O.-C. Granmo, B. Oommen, S. Myrer, and M. Olsen. Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(1):166 –175, feb. 2007.

[28] O.-C. Granmo and B. J. Oommen. Solving stochastic nonlinear resource allocation problems using a hierarchy of twofold resource allocation automata. *IEEE Transactions on Computers*, 59:545–560, 2009.

[29] H. Hajji. Statistical Analysis of Network Traffic for Adaptive Faults Detection. *IEEE Transactions on Neural Networks*, 16(5), 2005.

[30] H. L. Hammer and A. Yazidi. Parameter estimation in abruptly changing dynamic environments using stochastic learning weak estimator. *Applied Intelligence*, 48(11):4096–4112, 2018.

[31] Z. Han, H. Tan, R. Wang, G. Chen, Y. Li, and F. C. M. Lau. Energy-efficient dynamic virtual machine management in data centers. *IEEE/ACM Transactions on Networking (TON)*, 27(1):344–360, 2019.

[32] S. S. Haykin. *Adaptive filter theory*. Pearson Education India, 2008.

[33] T. Kailath, A. H. Sayed, and B. Hassibi. *Linear estimation*, volume 1. Prentice Hall Upper Saddle River, NJ, 2000.

[34] M. Khatua and S. Misra. CURD: Controllable Reactive Jamming Detection in Underwater Sensor Networks. *Pervasive and Mobile Computing*, 13:203–220, August 2014.

[35] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the International Conference on Very Large Data Bases*, page 180191. Morgan Kaufmann, 2004.

[36] H. Kim, B. L. Rozovskii, and A. G. Tartakovsky. A Nonparametric Multichart CUSUM Test for Rapid Detection of Dos Attacks in Computer Networks. *International Journal of Computing & Information Science*, 2(3), 2004.

[37] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intell. Data Anal.*, 8(3):281–300, Aug. 2004.

[38] I. Koychev. Gradual forgetting for adaptation to concept drift. In *Proceedings of ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning*, pages 101–106, 2000.

[39] I. Koychev and R. Lothian. Tracking drifting concepts by time window optimisation. In M. Bramer, F. Coenen, and T. Allen, editors, *Research and Development in Intelligent Systems XXII*, pages 46–59. Springer London, 2006.

[40] I. Koychev and I. Schwab. Adaptation to drifting user's interests. In *Proceedings of ECML2000 Workshop: Machine Learning in New Information Age*, pages 39–46, 2000.

[41] B. Krawczyk and M. Woźniak. One-class classifiers with incremental learning and forgetting for data streams with concept drift. *Soft Computing*, 19(12):3387–3400, 2015.

[42] P. Kulkarni and R. Ade. Incremental learning from unbalanced data with concept class, concept drift and missing features: A review. *International Journal of Data Mining and Knowledge Management Process (IJDKP)*, 4(6):15–29, Nov. 2014.

[43] L. I. Kuncheva. Change Detection in Streaming Multivariate Data Using Likelihood Detectors. *IEEE Transactions on Knowledge and Data Engineering*, 25(5):11751180, 2013.

[44] J. K. Lanctôt and B. J. Oommen. Discretized estimator learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-22(6):1473–1483, November/December 1992.

[45] M. Last. Online Classification of Nonstationary Data Streams. *Intelligent Data Analysis*, 6(2):129–147, April 2002.

[46] A. A. Mofrad, A. Yazidi, and H. L. Hammer. On solving the spl problem using the concept of probability flux. *Applied Intelligence*, pages 1–24, 2019.

[47] R. Mohan, A. Yazidi, B. Feng, and J. Oommen. On optimizing firewall performance in dynamic networks by invoking a novel swapping window–based paradigm. *International Journal of Communication Systems*, 31(15):e3773, 2018.

[48] K. Nishida and K. Yamauchi. Detecting Concept Drift Using Statistical Testing. In *Proceedings of the 10th International Conference on Discovery Science*, page 264269. Spriger, 2007.

[49] B. J. Oommen and M. Agache. Continuous and discretized pursuit learning schemes: Various algorithms and their comparison. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 31:277–287, 2001.

[50] B. J. Oommen and E. Hansen. The asymptotic optimality of discretized linear reward-inaction learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-14(3), May/June 1986.

[51] B. J. Oommen and S. Misra. A Fault-tolerant Routing Algorithm for Mobile Ad Hoc Networks Using a Stochastic Learning-based Weak Estimation Procedure. In *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications.*, pages 31–37. IEEE, 2006.

[52] B. J. Oommen and L. Rueda. Stochastic Learning-based Weak Estimation of Multinomial Random Variables and Its Applications to Pattern Recognition in Non-stationary Environments. *Pattern Recognition*, 39(3):328–341, 2006.

[53] B. J. Oommen, A. Yazidi, and O. C. Granmo. An Adaptive Approach to Learning the Preferences of Users in a Social Network Using Weak Estimators. *Journal of Information Processing Systems*, 8(2), 2012.

[54] B. S. Parker, L. Khan, and A. Bifet. Incremental ensemble classifier addressing non-stationary fast data streams. In *2014 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 716–723. IEEE, 2014.

[55] R. Pears, S. Sakthithasan, and Y. S. Koh. Detecting Concept Change in Dynamic Data Streams. *Machine Learning*, 97(3):259–293, 2014.

[56] A. S. Polunchenko and A. G. Tartakovsky. State-of-the-Art in Sequential Change-Point Detection. *Methodology and Computing in Applied Probability*, 14(3):649–648, September 2012.

[57] W. N. Robinson, A. Akhlaghi, T. Deng, and A. Syed. Discovery and Diagnosis of Behavioral Transitions in Patient Event Streams. *ACM Transactions on Management Information Systems*, 3(1), 2012.

[58] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33(2):191 – 198, 2012.

[59] L. Rueda and B. J. Oommen. Stochastic Automata-based Estimators for Adaptively Compressing Files with Nostationary Distributions. *IEEE Transaction on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(5), 2006.

[60] R. Sebastião and J. a. Gama. Change detection in learning histograms from data streams. In *Proceedings of the Aritficial Intelligence 13th Portuguese Conference on Progress in Artificial Intelligence*, EPIA'07, pages 112–123, Berlin, Heidelberg, 2007. Springer-Verlag.

[61] R. Sebastiao and J. Gama. A Study on Change Detection Method. In *Proceedings the 14th Portuguese Conference on Artificial Intelligence*, pages 353–364, Berlin, Heidelberg, 2009. Springer.

[62] T. Seidl, I. Assent, P. Kranen, R. Krieger, and J. Herrmann. Indexing density models for incremental learning and anytime classification on data streams. In *Proceedings of the 12th international conference on extending database technology: advances in database technology*, pages 311–322. ACM, 2009.

[63] A. N. Shiryayev. *Optimal Stopping Rules*. Springer, 1978.

[64] A. Stensby, B. J. Oommen, and O.-C. Granmo. Language detection and tracking in multilingual documents using weak estimators. In *Proceedings of the 2010 joint IAPR international conference on Structural, syntactic, and statistical pattern recognition*, pages 600–609, Berlin, Heidelberg, 2010. Springer-Verlag.

[65] Y. Sun, K. Tang, Z. Zhu, and X. Yao. Concept drift adaptation by exploiting historical knowledge. *IEEE transactions on neural networks and learning systems*, (99):1–11, 2018.

[66] M. A. L. Thathachar and B. J. Oommen. Discretized reward-inaction learning automata. *Journal of Cybernetics and Information Science*, pages 24–29, Spring 1979.

[67] I. W. Tsang, A. Kocsor, and J. T. Kwok. Simpler core vector machines with enclosing balls. In *Proceedings of the 24th international conference on Machine learning*, pages 911–918. ACM, 2007.

[68] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Dynamic integration of classifiers for handling concept drift. *Inf. Fusion*, 9(1):56–68, Jan. 2008.

[69] G. Widmer and M. Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. In *Machine Learning*, volume 23, pages 69–101, 1996.

[70] A. Yazidi, B. Oommen, and O.-C. Granmo. A novel stochastic discretized weak estimator operating in non-stationary environments. In *2012 International Conference on Computing, Networking and Communications (ICNC)*, pages 364–370, Jan 2012.

[71] A. Yazidi and B. J. Oommen. Novel discretized weak estimators based on the principles of the stochastic search on the line problem. *IEEE transactions on cybernetics*, 46(12):2732–2744, 2016.

[72] A. Yazidi, B. J. Oommen, G. Horn, and O.-C. Granmo. Stochastic discretized learning-based weak estimation: a novel estimation method for non-stationary environments. *Pattern Recognition*, 60:430 – 443, 2016.

[73] N. Ye, S. Vilbert, and Q. Chen. Computer Intrusion Detection Through EWMA for Autocorrelated and Uncorrelated Data. *IEEE Transactions on Reliability*, 52(1):75–82, 2003.

[74] J. Zhan, B. J. Oommen, and J. Crisostomo. Anomaly Detection in Dynamic Systems Using Weak Estimators. *ACM Transactions on Internet Technology*, 11(1), July 2011.

Appendix: Table of Abbreviations

Abbreviation	Expanded Form
ADWIN	Adaptive Sliding Window
CURD	Controllable Reactive Jamming Detection
CUSUM	Cumulative Sum
CVM	Core Vector Machine
DTEL	Diversity and Transfer-based Ensemble Learning
FIFO	First-In First-Out
FLORA	Floating Rough Approximation
GF	Gradual Forgetting
KL	Kullback-Leibler
KS	Kolmogorov Smirnoff
LA	Learning Automata
MEB	Minimum Enclosing Ball
ML	Machine Learning
MLEW	Maximum Likelikehood Estimator with Windows
NSE	Non-Stationary Environment
OLIN	An Online Classification System
PR	Pattern Recognition
SLWE	Stochastic Learning Weak Estimator
SPC	Statistical Process Control
SVM	Support Vector Machine
WEFTR	Weak-Estimation Fault-Tolerant Routing

Table 13: This table contains the list of abbreviations used in the paper.