Detecting and Adapting to Concept Drift in Continually Evolving Stochastic Processes

Sunanda Gamage
University of Moratuwa, Sri Lanka
+94 76 7528 928
rcsunanda@gmail.com

Upeka Premaratne
University of Moratuwa, Sri Lanka
+9411 265 0634 - 3328
upeka@ent.mrt.ac.lk

ABSTRACT

Many real world stochastic processes are non-stationary, which means that the probability distribution that generates data samples is time-varying. In the context of machine learning, this phenomenon is known as concept drift. It is important that machine learning models are able to adapt to concept drift in order to prevent degradation in accuracy. In this paper, we present two algorithms for drift detection and adaptation.

Drift is measured by continuously tracking a difference metric between probability distributions estimated from two sample windows preceding a time point. High values for the difference metric indicates that concept drift has occurred, and the model must be adapted. Adaptation is done by training a new model for the drifted process, and adding it to an ensemble of models. Previously trained models are retained, and their weights in the ensemble are adjusted to reflect similarity with the current probability distribution of the process. Experiments on simulated drift scenarios as well as real world datasets show that our algorithms detect drift with high accuracy, and adaptation results in improved model accuracy.

CCS Concepts

• Computing methodologies → Artificial intelligence → Machine Learning

Keywords

concept drift; drift detection; drift adaptation; incremental learning; ensemble methods; machine learning

1. INTRODUCTION

The vast majority of machine learning research focuses on building models (eg: regression models, classifiers) that are static; i.e. the model is trained one time on a training dataset and then applied to test data. This works fine when the underlying stochastic process that is modeled remains stationary. However, in many real world applications, the process is non-stationary, and it continuously changes with time.

For example, consider the video feed from a traffic camera, which can undergo changes from morning to night, and from summer to winter. The video feed process in this example undergoes gradual continuous change. The drift can also be abrupt and large, such as an impactful market event that affects stock prices.

The underlying stochastic process in these examples is non-stationary, which means that the joint probability distribution of the process attributes, $P(\boldsymbol{X})$, is time varying. This is known as concept drift. Applying a model that is trained one time with some initial training dataset to such a process results in performance degradation. Therefore, methods to characterize and detect the concept drift, and adapt machine learning models to drift by incremental learning are an important research direction in machine learning.

In this work, we build on the concept drift maps introduced in [2] and present a drift detection algorithm and an efficient ensemble technique of incremental learning to model a continuously changing stochastic process. The drift detection algorithm tracks a difference metric between probability distributions estimated from two sample windows preceding a time point. The estimated distribution is typically the joint probability distribution of attributes $P(\boldsymbol{X})$ for the unsurprised case, or joint attribute and class distribution $P(\boldsymbol{X}, Y)$ for the supervised case. When the continuous sum of this metric rises above a threshold m_{drift}, it is decided that the process has drifted and an adaptation should be made to the learned model.

At this point, a new model is learned from a suitable number of most recent samples, and it is added to an ensemble of previously learned models. The final time varying model is a weighted some of the sub models in the ensemble. The weights can be set in a way such that older sub models have less relevance (a form of forgetting). They can also be set to reflect the similarity between the probability distributions of the current sample window, and the windows corresponding to each sub model.

Our drift detection and adaption techniques can act as a wrapper method independent of the machine learning model being employed. The drift detection technique can be made to detect different types of drift [2] by choosing the relevant distribution for computing the difference metric. The methods work well for both supervised learning models such as classifiers, or unsupervised models such as hidden Markov models. The adaptation technique does not cause catastrophic forgetting. The forgetting factor can in fact be controlled by setting appropriate weights on the sub models in the ensemble. Experiment results of evaluating the methods on artificial and real world datasets show good performance in detecting drift and adapting to it.

The rest of this paper is organized as follows. Section 2 presents related work in detecting concept drift and adaptation methods. In Section 3, the concept drift problem is formally defined, and novel algorithms for drift detection and adaptation are described. In Section 4, we present the results of evaluating the method on several artificial datasets and a real world dataset. Some important implementation details and possible optimizations are described in Section 5. Conclusions and future directions are given in Section 6.

2. RELATED WORK

2.1 Detection techniques

Several works in the literature have developed drift detection techniques. In [1], drift is measured by tracing the error rate at test time, where a rise in error is considered as a sign of drift. Two threshold levels, warning level and drift level, are predefined, and when error increases above the drift level, the detector triggers an alarm. This method is computationally efficient, but needs the true labels at test time to compute the error rate (supervised detection).

A novel method introduced in [2], called concept drift maps, computes a difference metric in real-time between probability distributions estimated from two sample windows preceding a time point. This method allows the use of any probability distribution that represents the dataset, including the distribution of any subset of attributes. Using a distance metric instead of test error rate means it is well suited for cases where true labels are not revealed at test time (unsupervised drift detection). This work however focuses on measuring concept drift and visualizing it using maps, and does not describe a method for detection.

2.2 Adaptation techniques

There is a large body of work that explores adaptation methods for concept drift. An excellent overview of these methods and their key design principles is given in [3] and [13]. A common approach is to divide the historical data into blocks or windows that represent a context (a period where no drift has occurred or the process is stable) and use it to train a learner [4][5]. Alternatively, training sets can be formulated by instance weighing, where weights are assigned according to the age of instances or their performance with regard to the current concept [7][8].

The adaptation method described in [1] retrains a new model on the most recent context when a drift is detected, and any previous model is discarded. This leads to catastrophic forgetting, which means that any previously learned knowledge is completely forgotten, and recurring patterns will have no benefit from it.

In order to retain previous knowledge, [5][6] use an ensemble of models. In [6], Sequentially arriving samples are batched together by a fixed batch size, and weights of previous sub models in the ensemble are updated based on their performance on the batch. Then, the batch is sampled to extract the instances that were not classified accurately by previous sub models, and a new sub model is trained on them. This method does not detect drifts, but keeps updating the ensemble weights and adding new sub models as new batches arrive. This is computationally inefficient. Furthermore, this technique requires that true labels of new examples be available (supervised adaptation).

Fields that are related to learning under concept drift are "domain adaptation" and "transfer learning". In these areas, the topic of adapting a model trained on one domain or task to a different one is studied. The majority of this work considers one source domain and one or a few target domains. However, some researchers have extended domain adaptation to the incremental setting [9][10][11]. In [9], a method to adapt a base classifier to a change in class probability is described. The authors also present an extension for the case of sequentially arriving examples. However, this method does not work for changes in distributions of attributes or class conditional attributes.

3. METHODOLOGY

3.1 Problem definition

Consider a stochastic process characterized by a joint distribution $P(\boldsymbol{X})$ over the random variables $\boldsymbol{X} = \{X_1, \dots, X_n\}$. If our task is classification or regression, a target variable Y is also present, and the process is represented by the joint distribution $P(\boldsymbol{X}, Y)$ over $\{\boldsymbol{X}, Y\}$. A sequence of samples generated by this process are observed.

We say a concept drift has occurred in the process between two time points t, u when $P_t(\boldsymbol{X}, Y) \neq P_u(\boldsymbol{X}, Y)$. Since it is often not practical to estimate the distribution in effect at a specific point in time, we consider drift occurring between time intervals $[t_1, t_2]$ and $[u_1, u_2]$, which is characterized by $P_{[t_1,t_2]}(\boldsymbol{X}, Y) \neq P_{[u_1,u_2]}(\boldsymbol{X}, Y)$.

The drift detection problem is to quantify the above change in probability distributions with time, and identify when significant drift has occurred. The adaptation problem is to make necessary changes to a learned model so that it can be applied on the process and results in minimal error.

3.2 Detection technique

For drift detection, we build on a method called "concept drift maps" introduced in [2] for measuring concept drift. To quantify drift, we continuously compute a difference metric $d_t \in [0,1]$ between the probability distributions $P_{[t_1,t_2]}(\boldsymbol{Z})$ and $P_{[t_2,t]}(\boldsymbol{Z})$ as illustrated in Figure 1. $\boldsymbol{Z}$ can be any set of variables that captures the drift in the process, such as $\boldsymbol{Z} = \boldsymbol{X}$ or $\boldsymbol{Z} = \{\boldsymbol{X}, Y\}$ or $\boldsymbol{Z} = \{\boldsymbol{X}|Y\}$. In this work, we use the metric Total Variation Distance [12] as it is a symmetric metric that is efficient to compute.

$$d_t(Z) = \frac{1}{2}\sum_{z \in Z} |P_{[t_2,t]}(\boldsymbol{Z}) - P_{[t_1,t_2]}(\boldsymbol{Z})| \quad (1)$$

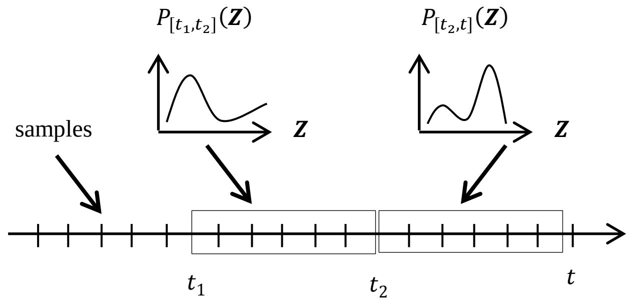
$P_{[t_1,t_2]}(\boldsymbol{Z})$
$P_{[t_2,t]}(\boldsymbol{Z})$
samples
$\boldsymbol{Z}$
$\boldsymbol{Z}$
t_1
t_2
t

Figure 1: Time windows for drift measurement

$d_t(Z)$ is tracked over time, and values above some low threshold are summed. The purpose of having the low threshold is to prevent noise or other artifacts from affecting the cumulative sum and triggering false alarms. When the difference metric is rising, and the cumulative sum increases above a certain threshold, a drift is detected and reported.

The variable $\boldsymbol{Z}$ used for the distance metric calculation will affect what type of concept drift will be captured by the detector. $\boldsymbol{Z} = \{\boldsymbol{X}, Y\}$ captures drift in its full form while the drift reflected by $\boldsymbol{Z} = \{\boldsymbol{X}|Y\}$ is known as virtual drift. $\boldsymbol{Z} = y$ captures drift due to class imbalance. The above choices of $\boldsymbol{Z}$ require that true class labels of test data instances be revealed to the detector (supervised drift detection). In the unsupervised case, $\boldsymbol{Z} = \boldsymbol{X}$ can be used to efficiently capture concept drift, and we use this choice of $\boldsymbol{Z}$ in our detection algorithm.

3.3 Adaptation technique

We propose an adaptation technique that employs an ensemble of weighted sub models (base learners) in this work. Initially, the ensemble has only the originally learned sub model. When a drift is detected by the detection method, the following actions are taken

1. A new sub model is trained from the examples in the current window $[t_2, t]$ and added to the ensemble.

2. Weights of sub models in the ensemble are updated based on the distance $d_{curr,i}$ between the current distribution and the distribution of the window on which the sub model was trained.

$$w_i = 1 - d_{curr,i} \quad (2)$$

Thus, the weight of a sub model represents the similarity between the current distribution of the process and the distribution that existed at the time of training the sub model. Since the final model is made of the weighted sum of the sub model outputs, this ensures that sub models that are most relevant to the current context of the process are given higher weights.

The sub models in the ensemble can be any type of learner, and the adaptation algorithm imposes no constraint on what it can be. However, if it is a supervised learner, true class labels of test data instances be must be revealed in order to train a new sub model (supervised adaptation). On the other hand, unsupervised base learners like Hidden Markov Models and Gaussian Mixture Models can be trained without the true labels test data.

4. EXPERIMENTAL EVALUATION

The proposed drift detection and adaptation algorithms were evaluated against artificial datasets and a real-world dataset that is commonly used in concept drift research. Both datasets represent classifier tasks, and we run the experiments with two classifiers; a neural network with a sigmod activation and a decision tree model. The adaptation problem is a supervised classifier adaptation; i.e. the test data labels are revealed after a prediction is made on them.

The proposed drift detection and adaptation method (labeled "with_adaptation") was compared with the following baselines.

- Only the original model with no change (labeled "no_adaptation")
- Model trained on all available data in the stream (labeled "all_data")
- Model trained on only the latest window (labeled "latest_window")

Note that the "all_data" and "latest_window" models are not practically feasible, as they require training a new model at every data point, which is computationally very expensive. They are used in experiments as baselines for comparison purposes. All tests were repeated five times, and the average result is presented.

4.1 Results on artificial datasets

The artificial dataset consists of two features and a binary class variable. The class conditional probability distributions $P(\boldsymbol{X}|Y)$ are Gaussians with an identity covariance matrix. To simulate three different scenarios of drift, the following three variations of the dataset were used.

Abrupt drift – Initially the means are $E(\boldsymbol{X}|Y=0) = [0,0]$ and $E(\boldsymbol{X}|Y=1) = [3,3]$. Simulate abrupt drift by setting $E(\boldsymbol{X}|Y=0) = [0,3]$ and $E(\boldsymbol{X}|Y=1) = [3,0]$ in the middle of the data stream.

Gradual drift – Initially the means are $E(\boldsymbol{X}|Y=0) = [0,0]$ and $E(\boldsymbol{X}|Y=1) = [3,3]$. Simulate gradual drift by moving the distributions within a period of time to $E(\boldsymbol{X}|Y=0) = [3,0]$ and $E(\boldsymbol{X}|Y=1) = [6,3]$

Recurring context – The distributions are similar to the abrupt drift scenario, except that the two sets of distributions are switched at time intervals, causing two contexts to recur in the data stream.

Three data streams simulating the above three drift scenarios were given to our detection and adaptation method. Figure 2 shows the concept drift maps and illustrates where the actual drift occurred, and where the detection algorithm has triggered detections. In all three scenarios, the total variation distance measure between two preceding windows has increased as expected. The increase is apparent in the abrupt drift and recurrent context cases, while it is more subtle in the gradual drift case. There is a delay between the point where actual drift has occurred and the distance measure has peaked, which can be controlled by tuning the sample window size used for estimating the distributions and computing the distances. Small window sizes result in a smaller delay, but the accuracy of the distribution estimation, and therefore the accuracy of the distance measure is adversely affected. We found a window size of 500 samples to be a good tradeoff point. In all three drift scenarios, the detection algorithm has correctly captured the drift that has occurred. In some cases, drift has been detected multiple times after actual drift, which is acceptable as the adaptation algorithm will favor the most relevant sub model in the ensemble.

Figure 3 compares the online error rates (calculated considering a small moving window) of our adaptation method with other baselines. The overall average error results of these experiments are given in Table 1. The error rate graphs and the average values clearly illustrate how the model adapted by our algorithm ("with_adaptation" model) is vastly superior than the "no_adaptation" model in all three drift scenarios. The "latest_window" model performs on par with the adapted model, and the "all_data" model is a close second. However, these two models are not practically feasible as they are computationally expensive, and only provide a lower bound.

4.2 Results on real world dataset

The real-world dataset we have used to verify our methods is the Australian New South Wales Electricity Market dataset (ELEC), which has also been used in other concept drift research [1][14][15]. The dataset contains 45312 labeled examples collected from the Australian electricity market from 7 May 1996 to 5 December 1998. Each example on the dataset has 5 features, the day of week, the time stamp, the NSW electricity demand, the Vic electricity demand, the scheduled electricity transfer between states and the class label. The class label identifies the change of the price related to a moving average of the last 24 hours (whether it has moved up or down).

The classifiers employed in this experiment were also a neural network trained with stochastic gradient descent, and a decision tree. Initially, the model was trained with 10000 examples, and the remaining examples were given to the detection and adaptation algorithm as a data stream.

Figure 4 shows the concept drift map and the locations where the detection algorithm has detected drifts. Figure 5 compares the online error rate of the algorithm and other baselines on this dataset. The total average error results are given in Table 2. In this dataset, we do not know where actual drift occurs. The detected locations roughly correspond with those found in [1]. Moreover, after detection and adaptation, there is a clear drop in error (Figure 5) in comparison to the "no_adaptation" model, which implies that the detected locations correspond to actual drifts, and that the adaptation has resulted in higher accuracy.

Table 1: Comparison of overall average error on different datasets with different adaptation methods

Model	Abrupt drift		Gradual drift		Recurring context		Real world dataset	
	Neural net	Decision tree	Neural net	Decision tree	Neural net	Decision tree	Neural net	Decision tree
With adaptation	0.0571	0.0652	0.0276	0.0403	0.0835	0.0693	0.5040	0.3952
No adaptation	0.2585	0.2257	0.1061	0.1460	0.2748	0.1608	0.5672	0.4183
All data	0.0668	0.0894	0.0415	0.0549	0.0700	0.0961	0.4377	0.3441
Latest window	0.0268	0.0430	0.0214	0.0310	0.0370	0.0499	0.4385	0.3372

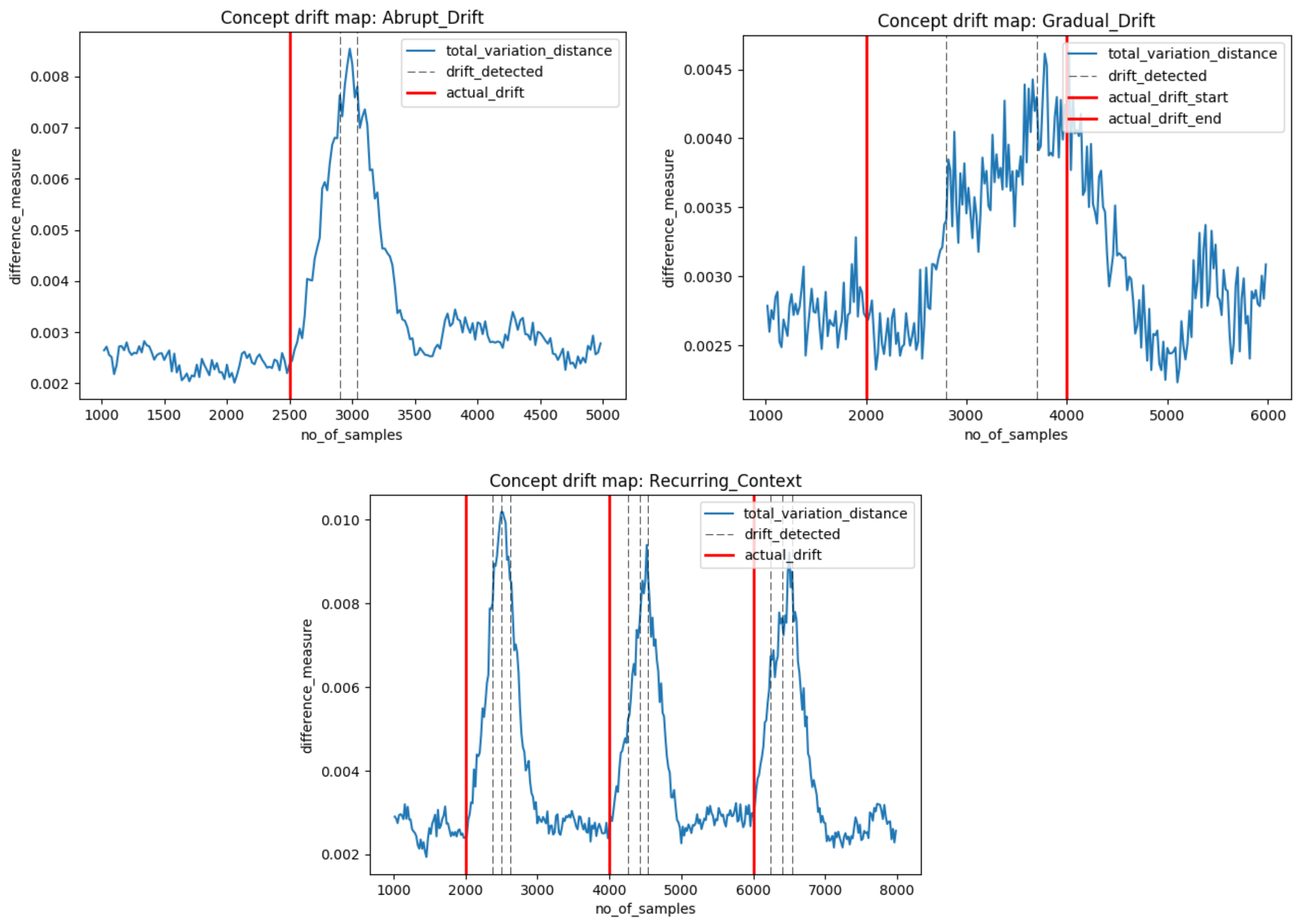

Figure 2: Concept drift maps of the three drift scenarios. Top left: abrupt drift, top right: gradual drift, bottom: recurrent context

5. IMPLEMENTATION AND OPTIMIZATIONS

In the detection algorithm, the method used for estimating the probability distribution of a sample window is kernel density estimation (KDE), as it is a non-parametric method suitable for estimating complex high dimensional distributions. It is also a well-studied method with many techniques to improve estimation error and complexity [16][17]. When calculating the Total Variation Distance between two KDE estimated distributions, equation (1) becomes;

$$d_t(Z) = \frac{1}{2}\int_{z\in Z} |P_{[t_2,t]}(\boldsymbol{Z}) - P_{[t_1,t_2]}(\boldsymbol{Z})| \quad (3)$$

For high dimensional data, this numeric integration can be done efficiently with Monte Carlo Integration.

It must be noted that in a system with real-time performance expectations, the detection and adaptation algorithms must run in parallel with the main classification task of the ensemble. This ensures that the computations needed for tracking the concept drift map in the detection algorithm do not affect the main classification task. It is also more practical to run the detection algorithm after a certain number of examples are received, rather than at each example in the stream, as one data point does not change the distribution significantly.

We also suggest several potential optimizations to our algorithms. Since our detection method uses the Total Variation Distance between two sample windows, there is a delay between when the actual drift occurs and when it is detected (this delay can be controlled by tuning the sample window size used to estimate the distributions). The online error rate during this period is high, as seen in Figure 3 and Figure 5. In the supervised case (class labels of test instances are revealed to the detection algorithm), the revealed error rate could be incorporated into the detection algorithm, which will enable quick drift detection. However, this optimization is not possible in the unsupervised detection case.

Another potential optimization in the supervised case is to modify the sub model weights in adaptation (equation (2)). Since the true class labels are available, the weights of each sub model could be determined by taking into account how well each sub model performs on the latest data batch.

In this work, we have used $P(\boldsymbol{X})$ to compute the concept drift maps in order to capture drifts in the data distribution. It is also possible to use the distribution of any subset $\boldsymbol{X}_s \subset \boldsymbol{X}$ for this purpose. By identifying subsets of features that have caused the drift, it is possible to train the new sub model of the ensemble with only those features. This will result in a better performing model with low variance, especially in the case of high dimensional datasets. However, this will introduce high computational complexity, and an efficient technique must be employed to select the subsets used in computing the concept drift maps.

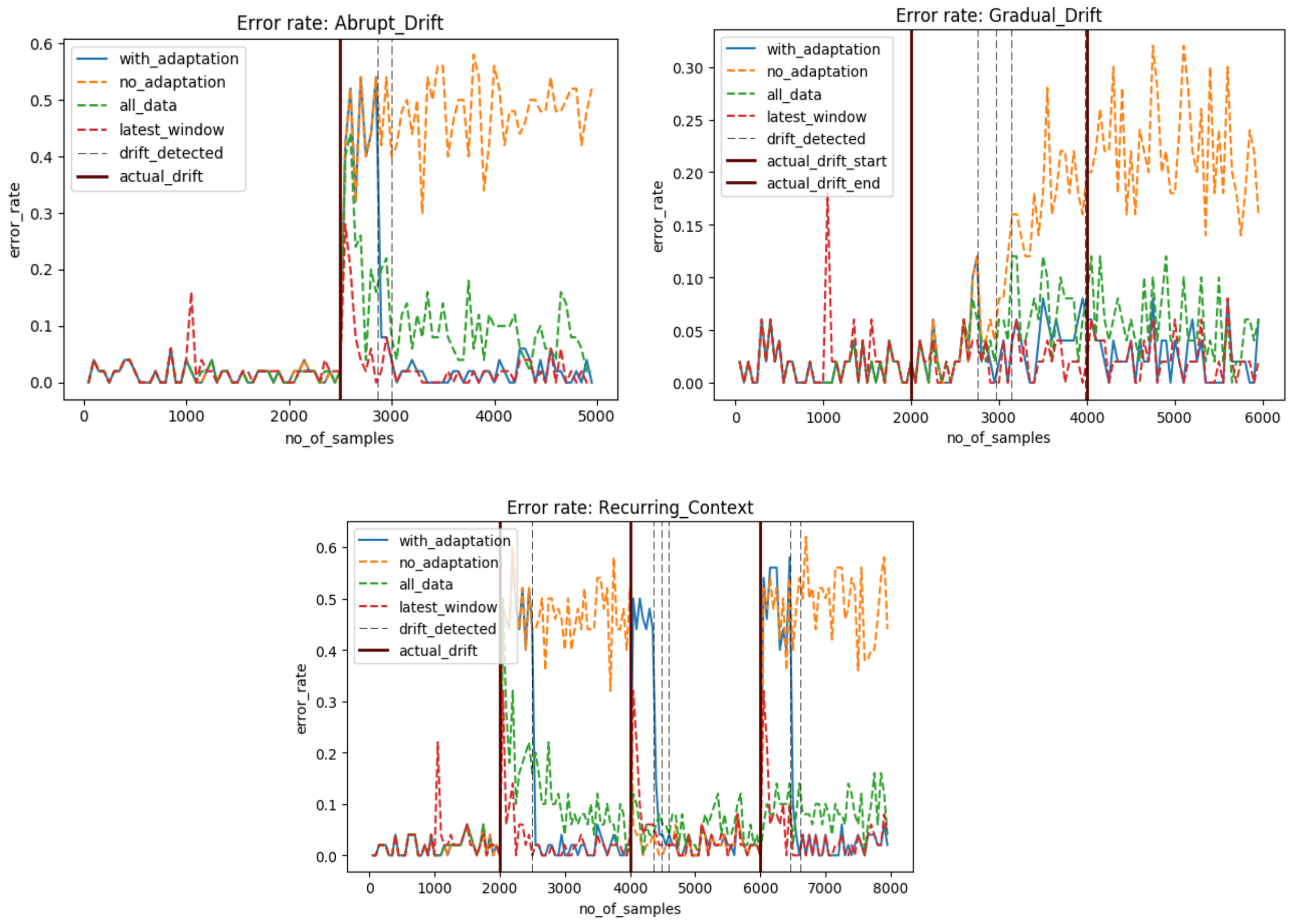

Figure 3: Comparison of online error rates of adaptation algorithm and other baselines (for neural network classifier). Top left: abrupt drift, top right: gradual drift, bottom: recurrent context

6. CONCLUSION

In this work, we presented an algorithms for concept drift detection and an efficient ensemble method for adaptation to drift. Our experiments show that these techniques perform well on simulated drift scenarios (abrupt, gradual, recurrent) as well as a real world dataset. The method is also robust to noise, and retains knowledge for recurrent contexts.

In future works, we plan to implement some of the optimizations outlined in Section 5. In particular, we will focus on capturing concept drift in subsets of features and using them to train sub models in the ensemble. We will also test our drift detection and adaptation techniques on applications with highly time-varying datasets, such as network traffic analysis and anomaly detection.

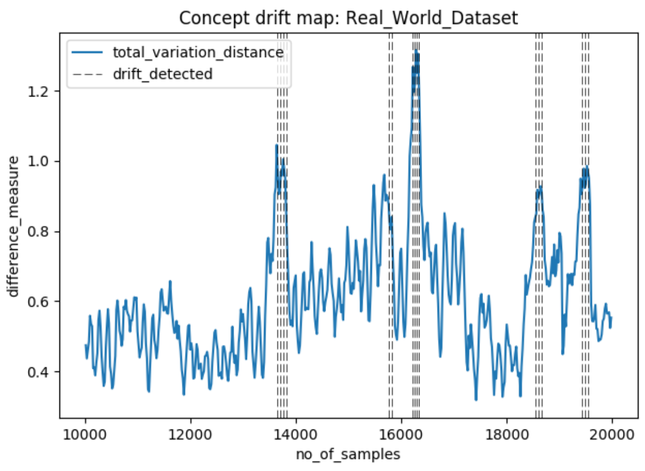

Figure 4: Concept drift map of real world dataset

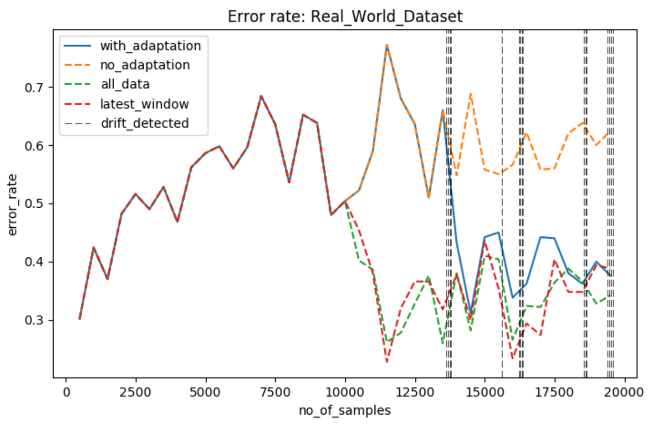

Figure 3: Comparison of online error rates of adaptation algorithm and other baselines on real world dataset (for neural network classifier: the data stream starts after the initial 10000 samples)

7. REFERENCES

[1] Gama, Joao, et al. "Learning with drift detection." Brazilian Symposium on Artificial Intelligence. Springer, Berlin, Heidelberg, 2004.

[2] Webb, Geoffrey I., et al. "Understanding Concept Drift." arXiv preprint arXiv:1704.00362 (2017).

[3] Žliobaitė, Indrė. "Learning under concept drift: an overview." arXiv preprint arXiv:1010.4784 (2010).

[4] Street, W. Nick, and YongSeog Kim. "A streaming ensemble algorithm (SEA) for large-scale classification." Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2001.

[5] Wang, Haixun, et al. "Mining concept-drifting data streams using ensemble classifiers." Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. AcM, 2003.

[6] Scholz, Martin, and Ralf Klinkenberg. "An ensemble classifier for drifting concepts." Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams. Porto, Portugal, 2005.

[7] I. Koychev. Koychev, Ivan. "Gradual forgetting for adaptation to concept drift." Proceedings of ECAI 2000 Workshop on Current Issues in Spatio-Temporal Reasoning,, 2000.

[8] Zhang, Peng, Xingquan Zhu, and Yong Shi. "Categorizing and mining concept drifting data streams." Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008.

[9] Royer, Amelie, and Christoph H. Lampert. "Classifier adaptation at prediction time." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[10] Hoffman, Judy, Trevor Darrell, and Kate Saenko. "Continuous manifold based adaptation for evolving visual domains." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014.

[11] Levinkov, Evgeny, and Mario Fritz. "Sequential Bayesian model update under structured scene prior for semantic road scenes labeling." Proceedings of the IEEE International Conference on Computer Vision. 2013.

[12] Levin, David Asher, Yuval Peres, and Elizabeth Lee Wilmer. Markov chains and mixing times. American Mathematical Soc., 2009.

[13] Tsymbal, Alexey. "The problem of concept drift: definitions and related work." Computer Science Department, Trinity College Dublin 106.2 (2004).

[14] Michael Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report

[15] Bifet, Albert, and Ricard Gavalda. "Learning from time-changing data with adaptive windowing." Proceedings of the 2007 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2007.

[16] Raykar, Vikas C., Ramani Duraiswami, and Linda H. Zhao. "Fast computation of kernel estimators." Journal of Computational and Graphical Statistics 19.1 (2010): 205-220.

[17] Elgammal, Ahmed, Ramani Duraiswami, and Larry S. Davis. "Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking." IEEE transactions on pattern analysis and machine intelligence 25.11 (2003): 1499-1504.

Authors’ background

Your Name	Title	Research Field	Personal website
Sunanda Gamage	Student	Machine learning	https://www.linkedin.com/in/sunandagamage/
Upeka Premaratne	Senior lecturer	Machine learning, Control systems	https://scholar.google.com/citations?user=fZoiEG4AAAAJ&hl=en