Efficient Multistream Classification using Direct Density Ratio Estimation

Ahsanul Haque*, Swarup Chandra*, Latifur Khan*, Kevin Hamlen*, and Charu Aggarwal†
*Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA
Email: {axh129430, src093020, lkhan, hamlen}@utdallas.edu
†IBM T. J. Watson Research Center, Yorktown, NY, USA, Email: charu@us.ibm.com

***Abstract*—Traditional data stream classification techniques assume that the stream of data is generated from a single non-stationary process. On the contrary, a recently introduced problem setting, referred to as Multistream Classification involves two independent non-stationary data generating processes. One of them is the source stream that continuously generates labeled data instances. The other one is the target stream that generates unlabeled test data instances from the same domain. The distributions represented by the source stream data is biased compared to that of the target stream. Moreover, these streams may have asynchronous concept drifts between them. The multistream classification problem is to predict the class labels of target stream instances, while utilizing labeled data available from the source stream. In this paper, we propose an efficient solution for multistream classification by fusing drift detection into online data shift adaptation. Experiment results on benchmark data sets indicate significantly improved performance over the only existing approach for multistream classification.**

Keywords*—*Multistream Classification; Data Shift adaptation; Direct Density Ratio Estimation

I. Introduction

Data stream mining has attracted researchers due to its importance in today's connected digital world. However, until recently, researchers have focused on mining a single stream of data [1], [2], [3]. Even if data is received from more than one streams simultaneously, all of them are assumed to be generated from a non-stationary data generating process [4]. Therefore, all such streams can be combined into a single stream of data, as data from these streams represent the same distribution. However, combining streams may not be effective in particular scenarios, especially if these streams represent different distributions with asynchronous and independent concept drifts among them. This type of scenarios may arise if data is generated by different but related non-stationary processes.

For example, consider building a model for predicting sentiment of tweets [5]. Typically, sentiment is not provided as the ground truth along with a tweet. So, in order to collect training data, a few users may agree to provide tweets along with sentiment label information. On the contrary, tweets on which the model needs to analyze the sentiment may come from any twitter user. Users providing the training data may represent only a small portion of the population. Therefore, if we assume two streams of data, one from the twitter users providing labeled data, another from the whole population of twitter users, a sampling bias may exist between the distributions represented by these streams of data. This type of data shift between streams of data may occur due to limited supervision, or lack of control over the data generating process [4].

A new problem setting called *Multistream Classification* has been introduced in [4] to address similar scenarios as discussed above. It involves two simultaneous streams of data. One of the streams, called the *source stream*, provides labeled training data. The other stream, called the *target stream*, provides only unlabeled test data. The classification task is to use the labeled data from the source stream for classifying unlabeled data from the target stream efficiently. As pointed out before, combining these streams may result in a different overall distribution compared to the individual distributions. Moreover, independent and asynchronous concept drifts may occur in either stream over time. Therefore, traditional techniques for data stream mining may not be effective if applied on the combined stream.

In this paper, we propose an efficient approach for multistream classification. The approach uses two sliding windows for storing recent instances from the source and the target stream. Data shift between these streams is addressed by weighing each source instance based on the density ratio. Let $P_S(\cdot)$ and $P_T(\cdot)$ be the distributions represented by recent source and target data instances respectively. Density ratio for an instance $\boldsymbol{x}$ is defined by $\beta(\boldsymbol{x}) = \frac{P_T(\boldsymbol{x})}{P_S(\boldsymbol{x})}$. A Gaussian kernel model is used in the proposed approach for direct density ratio estimation. The model is updated online with incoming instances. An ensemble classifier is used for classification, where each model is trained on weighted source stream instances.

The proposed approach has inherent capability of addressing asynchronous concept drifts in multistream classification. It uses density ratios estimated by the Gaussian kernel model for detecting any change between distributions represented by weighted source and target stream data. If a significant change is detected, the Gaussian kernel model for density ratio estimation is updated. Moreover, weights for the source stream labeled data are re-evaluated using the updated kernel model, and the ensemble classifier is updated. Efficiency of the proposed approach stems from the fact that it uses the same kernel model for addressing both data shift and asynchronous concept drift in multistream classification. Experiment results on benchmark data sets show the efficiency of the proposed approach in terms of both accuracy and execution time.

II. Related Work

A fundamental assumption in data mining is that both training and test data represent the same data distribution [6]. However, this assumption may be violated in practical real-world applications due to limited supervision, or lack of control over the data gathering process. Addressing arbitrary

differences between training and test data distribution is a difficult problem [7]. That is why, most approaches addressing this challenge assume that the training and test data distributions, denoted by $P_{tr}(\cdot)$ and $P_{te}(\cdot)$ respectively, are related by a covariate shift assumption. More specifically, the relationship between the training and test data distributions is such that $P_{tr}(y|\boldsymbol{x}) = P_{te}(y|\boldsymbol{x})$ and $P_{tr}(\boldsymbol{x}) \neq P_{te}(\boldsymbol{x})$, where $\boldsymbol{x}$ and y denote covariates and label of the data instance respectively. Kernel Mean Matching [7] and KLIEP [8] are among the techniques that are available in the literature for handling covariate shift in data. However, these approaches work only on fixed-size training and test data.

Kawahara and Sugiyama [9] extended KLIEP for direct online density ratio estimation. However, they considered a single stream of data, where set of training/reference and test data instances are determined by a sliding window. In this paper, we consider multistream classification problem [4], where two simultaneous data streams, i.e., source and target streams over the same domain are considered. MSC (Multi-Stream Classifier) [4] is the only known existing technique for multistream classification. However, it suffers from a number of shortcomings. First, it uses a hybrid ensemble classifier with complex update procedure. Second, it executes expensive concept drift detection simultaneously over the two data streams for handling asynchronous concept drifts. Third, once a concept drift is detected in either stream, it uses expensive batch algorithm for data shift adaptation. These shortcomings add a significant overhead to the execution time.

III. The Proposed Approach

In this paper, we propose an efficient solution for multi-stream classification by fusing drift detection into data shift adaptation. We refer to this approach as FUSION (eFficient mUltiStream classification using direct densIty estimatiON).

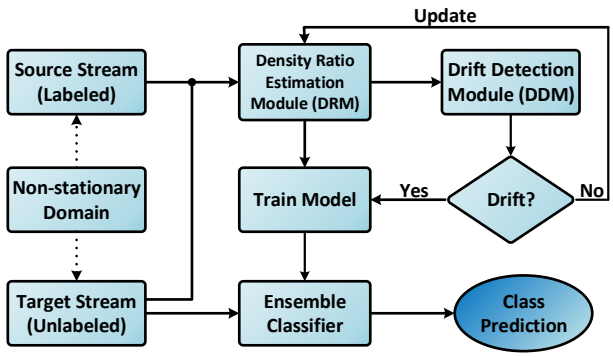


Fig. 1: Overview of FUSION

The core components of FUSION have been illustrated in Figure 1. It has four modules, i.e., *Density Ratio Estimation (DRM)*, *Drift Detection (DDM)*, *classification*, and *update*. Let $\mathcal{S}$ and $\mathcal{T}$ denote the source and target stream respectively. Two sliding windows are used to store recent instances from $\mathcal{S}$ and $\mathcal{T}$, referred to as source and target sliding window, and denoted by $\boldsymbol{\mathcal{W}}_S$ and $\boldsymbol{\mathcal{W}}_T$ respectively. The maximum allowable size, and the current sizes of $\boldsymbol{\mathcal{W}}_S$ and $\boldsymbol{\mathcal{W}}_T$ are denoted by N_m, N_S, and N_T respectively.

FUSION uses an ensemble classifier for classification. The first model in the ensemble is trained on the initial instances from $\boldsymbol{\mathcal{W}}_S$. However, to correct possible covariate shift between $\mathcal{S}$ and $\mathcal{T}$, each instance from the sliding window is given an importance weight. FUSION uses the density ratios estimated from the Density Ratio Estimation Module (DRM) as the importance weights. Label for any new instance arriving in $\mathcal{T}$ is predicted by taking the majority voting from the ensemble classifier. DRM updates the model for density ratio estimation incrementally with each incoming instance in either window.

As new instances arrive in $\mathcal{S}$ or $\mathcal{T}$, the Drift Detection Module (DDM) detects any significant drift between the distributions represented by weighted source stream data, and target stream data. Once a drift is detected, DRM is updated to incorporate the change, and weights for source stream data are re-evaluated. Subsequently, the new weights are used for training a new model, and updating the ensemble classifier. Next, we discuss the modules in FUSION.

A. Density Ratio Estimation Module (DRM)

FUSION uses a Gaussian kernel model similar to [8] for direct density estimation. The model is updated incrementally as new instances appear in $\mathcal{S}$ or $\mathcal{T}$. Next, we describe the density ratio estimation module used by FUSION, and its online update procedure.

1) Gaussian kernel model: At time t, we define source distribution P_S and target distribution P_T by the distributions represented by data instances in $\boldsymbol{\mathcal{W}}_S$ and $\boldsymbol{\mathcal{W}}_T$ respectively at that moment. Instances from $\mathcal{S}$ are labeled, and used to train the classification models. For any instance $\boldsymbol{x}$ from $\mathcal{S}$, $\beta(\boldsymbol{x}) = \frac{P_T(\boldsymbol{x})}{P_S(\boldsymbol{x})}$ is used as its importance weight in the learning process. Using Gaussian kernel model, $\beta(\boldsymbol{x})$ is estimated as follows:

$$\hat{\beta}(\boldsymbol{x}) = \sum_{i=1}^{N_T} \alpha_i K_\sigma \left(\boldsymbol{x}, \boldsymbol{\mathcal{W}}_T^{(i)}\right) \tag{1}$$

where $\boldsymbol{\mathcal{W}}_T^{(i)}$ is the i^{th} instances in $\boldsymbol{\mathcal{W}}_T$, $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^{N_T}$ is the set of parameters to be learned, and $K_\sigma(\cdot,\cdot)$ is a Gaussian kernel with kernel width σ, i.e., $K_\sigma(\boldsymbol{x},\boldsymbol{x}') = \exp\left\{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}\right\}$. The target sliding window instances, $\boldsymbol{\mathcal{W}}_T$, works as the Gaussian centers. We choose kernel width σ by *likelihood cross validation* following [8].

2) Learning parameters: The target distribution is estimated by weighted training distribution, $\hat{P}_T(\boldsymbol{x}) = \hat{\beta}(\boldsymbol{x})P_S(\boldsymbol{x})$. The objective of minimizing the Kullback-Liebler divergence from $P_T(\boldsymbol{x})$ to $\hat{P}_T(\boldsymbol{x})$ leads to the following convex optimization problem-

$$\underset{\{\alpha_i\}_{i=1}^{N_T}}{\text{maximize}} \left[\sum_{j=1}^{N_T} \log\left(\sum_{i=1}^{N_T} \alpha_i K_\sigma\left(\boldsymbol{\mathcal{W}}_T^{(j)}, \boldsymbol{\mathcal{W}}_T^{(i)}\right)\right)\right]$$
$$\text{subject to } \frac{1}{N_S}\sum_{j=1}^{N_S}\sum_{i=1}^{N_T} \alpha_i K_\sigma\left(\boldsymbol{\mathcal{W}}_S^{(j)}, \boldsymbol{\mathcal{W}}_T^{(i)}\right) = 1,$$
$$\text{and } \alpha_1, \alpha_2, \ldots, \alpha_{N_T} \geq 0. \tag{2}$$

The set of parameters $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^{N_T}$ in model (1) is learned by solving the above optimization problem.

3) Updating parameters online: Kawahara and Sugiyama have proposed an online update method for $\boldsymbol{\alpha}$ in [9]. However, unlike multistream classification scenario, this method assumes only one stream of data with a sliding window for defining the set of reference and test data instances. In this paper, we adapt this method for multistream classification scenario.

In case of a new instance in $\mathcal{S}$, only the constraints in the optimization problem in Eq. (2) need to be satisfied, as the instance does not affect the optimization problem directly. On the contrary, a new instance in $\mathcal{T}$ directly affects the problem in Eq. (2). Therefore, $\boldsymbol{\alpha}$ needs to be updated along with constraint satisfaction in this case.

The method for updating $\boldsymbol{\alpha}$ online is based on the online learning technique for kernel methods proposed in [10]. Assuming that $\beta(\cdot)$ is searched within a reproducing kernel Hilbert space $\mathcal{H}$, the following reproducing property holds-

$$\langle \beta(\cdot), K(\cdot, \boldsymbol{x}') \rangle = \beta(\boldsymbol{x}') \tag{3}$$

Considering the empirical error for the new instance in $\mathcal{T}$, and the reproducing property stated above, $\boldsymbol{\alpha}$ is updated as follows-

$$\begin{cases} \hat{\alpha}'_i \leftarrow (1-\eta\lambda)\hat{\alpha}_{i+1} & i = 1, \ldots, N_T - 1 \\ \hat{\alpha}'_i \leftarrow \frac{\eta}{\hat{\beta}\left(\boldsymbol{\mathcal{W}}_T^{(N_T+1)}\right)} & i = N_T \end{cases} \tag{4}$$

where λ and η denote the regularization parameter and the learning rate respectively.

B. Training and Classification

FUSION uses an ensemble classifier, denoted by $\mathcal{M}$. It trains the first model in the ensemble using the initial N_S and N_T instances in $\boldsymbol{\mathcal{W}}_S$ and $\boldsymbol{\mathcal{W}}_T$ respectively, referred to as the warm-up period instances. Importance weights for $\boldsymbol{\mathcal{W}}_S$ instances are estimated by the Density Ratio Estimation (DRM) module as follows-

$$\hat{\beta}\left(\boldsymbol{\mathcal{W}}_S^{(i)}\right) = \sum_{j=1}^{N_T} \alpha_j K_\sigma\left(\boldsymbol{\mathcal{W}}_S^{(i)}, \boldsymbol{\mathcal{W}}_T^{(j)}\right), i = 1, \ldots, N_S \tag{5}$$

Any learning algorithm that can incorporate importance weight of training instances can be used in FUSION. As new instances continue to arrive in $\mathcal{S}$ or $\mathcal{T}$, the ensemble classifier $\mathcal{M}$ is updated if there is a drift between distributions represented by weighted source stream and target stream data. FUSION predicts the majority voted class as the class of an incoming test instance in the target stream.

C. Drift Detection Module (DDM)

As mentioned before, P_T is estimated by $\hat{P_T} = \hat{\beta}(\boldsymbol{x})P_S(\boldsymbol{x})$. The classifier is updated following a drift, i.e., a significant difference between P_T and $\hat{\beta}(\boldsymbol{x})P_S(\boldsymbol{x})$. Let $\boldsymbol{\alpha}^o$ be the set of initial parameters. As mentioned before, these parameters are updated online as new instances arrive in $\mathcal{S}$ or $\mathcal{T}$. Let $\boldsymbol{\alpha}^t$ be the set of parameters at time t. A drift is detected at time t if-

$$S = \sum_{i=1}^{N_T} \ln \frac{P_T\left(\boldsymbol{\mathcal{W}}_T^{(i)}\right)}{\hat{\beta}_o P_S\left(\boldsymbol{\mathcal{W}}_T^{(i)}\right)} = \sum_{i=1}^{N_T} \ln \frac{\hat{\beta}_t\left(\boldsymbol{\mathcal{W}}_T^{(i)}\right)}{\hat{\beta}_o\left(\boldsymbol{\mathcal{W}}_T^{(i)}\right)} > \tau$$

where $\hat{\beta}_o$ and $\hat{\beta}_t$ are density ratios defined by $\boldsymbol{\alpha}^o$ and $\boldsymbol{\alpha}^t$ respectively, and τ is a user defined threshold.

D. Classifier Update

Once a drift is detected, or the sliding windows reach the maximum capacity, first the Gaussian kernel model of DRM (specified in Eq. (1)) is updated by re-evaluating $\boldsymbol{\alpha}$ as discussed in Section III-A2. Then, a new model is trained based on instances from $\boldsymbol{\mathcal{W}}_S$ along with weights calculated by the updated DRM. Next, the ensemble classifier $\mathcal{M}$ is updated using the newly trained model. The maximum number of models $\mathcal{M}$ can contain is L. If $\mathcal{M}$ contains less than L models currently, the new model is simply added to $\mathcal{M}$. Otherwise, the worst model in the ensemble is replaced by the new model. Finally, $\boldsymbol{\mathcal{W}}_S$ and $\boldsymbol{\mathcal{W}}_T$ are re-initialized.

IV. Evaluation

Dataset	# features	# classes	# instances
ForestCover	54	7	150,000
KDD	42	23	200,000
PAMAP	53	19	150,000
SynRBF@002	70	7	100,000
SynRBF@003	70	7	100,000

TABLE I: Characteristics of Data Sets

A. Data sets

Table I lists the data sets used in the experiments. The first three data sets are from real-world. *SynRBF@X* are synthetic data sets generated using *RandomRBFGeneratorDrift* from MOA [11] framework, where X is the Speed of change of centroids in the model. We generate two such data sets using $X = \{0.002, 0.003\}$ to evaluate the approaches on concept drifts having various intensities and frequencies. Each of these synthetic data sets contain data having 70 attributes and from 7 classes. We simulate source and target streams from each data set following the method mentioned in [4].

B. Setup

We have used the only available method for multistream classification, i.e., *Multistream Classifier (MSC)* [4] as the baseline approach. MSC uses *Support Vector Machine (SVM)* as the base classifier. To implement SVM, we used weighted LibSVM library [12] with RBF kernel as suggested in [4]. For a fair comparison, we have also used SVM as the base classifier in the proposed approach (FUSION). The kernel width (σ) for the Gaussian kernel model in FUSION is selected by likelihood cross validation. We have used $N_m = 500$ and $L = 6$ as the default setting, if not mentioned otherwise. Moreover, we use $\lambda = 0.01$ and $\eta = 1$ in the experiments following [9]. We have implemented both approaches using *Python* version 2.7.6. We have executed all the experiments in a *linux* machine with *2.40 GHz* core and *16 GB* of main memory.

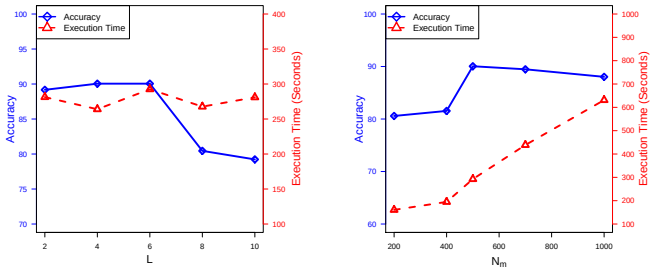
Data Set	FUSION		MSC	
	Accuracy	Time (Sec)	Accuracy	Time (Sec)
ForestCover	**76.23**	**305.82**	59.62	443.46
KDD	**98.16**	**271.24**	94.36	765.18
PAMAP	**90.05**	**292.51**	83.79	499.60
SynRBF@002	**54.03**	**319.72**	37.98	465.36
SynRBF@003	**53.87**	**336.99**	38.89	513.14

TABLE II: Classification Accuracy and Execution Time

C. Performance

Classification accuracy of FUSION on different data sets have been shown in Table II. We observe that FUSION achieves much better accuracy compared to the existing baseline approach (MSC). The difference in performance is more evident in case of the synthetic data sets, where we alter degree of concept drifts intentionally to test the adaptability of the approaches.

We have reported average time to process 1000 instances (in seconds) by the approaches on different data sets in Table II. The major contributing factor in the time complexity of FUSION is learning $\boldsymbol{\alpha}$ in the Gaussian kernel model. Since $\boldsymbol{\alpha}$ is learned only once at the beginning, and then updated online onward, FUSION is expected to be faster than MSC. Results from Table II shows that FUSION exhibits much better performance in terms of execution time compared to MSC.



(a) Ensemble Size (b) Maximum Window Size

Fig. 2: Parameter Sensitivity of FUSION on PAMAP

D. Sensitivity

In this section, we examine the sensitivity of FUSION to the ensemble size (L), and the maximum size of the sliding windows (N_m). We plot the average accuracy and execution time taken to process 1000 data instances by FUSION with different values of the parameters in Fig (2). It can be observed from Fig. (2a) that FUSION is not much sensitive to the size of the ensemble. As mentioned before, unlike MSC, ensemble management is much lightweight in FUSION compared to MSC. Therefore, changing ensemble size does not affect the execution time of FUSION significantly. Sensitivity of FUSION to N_m is shown in Fig. (2b). The accuracy slightly increases with increasing N_m. Importantly, we observe that the execution time of FUSION increases gradually with increasing N_m. This is expected since the execution time of FUSION depends on N_m. Overall the experiment results indicate that FUSION is not too much sensitive to the parameters.

V. Conclusion

In this paper, we have proposed FUSION, a framework for efficient multistream classification. The main challenges of multistream classification are data shift, and asynchronous data drifts between source and target stream data. To address these challenges, FUSION uses an ensemble classifier, where each model is trained using weighted instances from the source stream. The weights are estimated using a Gaussian kernel model by estimating density ratios. The same model is also used for addressing asynchronous drifts between source and target stream. Experiment results show the effectiveness of the proposed approach.

Acknowledgments

This material is based upon work supported by The Air Force Office of Scientific Research under award no. FA9550-14-1-0173 and FA9550-12-1-0077, NSF award no. DMS-1322353, and IBM faculty award (Research).

References

[1] A. Haque, L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal, "Efficient handling of concept drift and concept evolution over stream data," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, May 2016, pp. 481–492.

[2] A. Haque, L. Khan, and M. Baron, "Sand: Semi-supervised adaptive novel class detection and classification over data stream," in *Thirteenth AAAI Conference on Artificial Intelligence*, Feb 2016, pp. 1652–1658.

[3] M. M. Masud, T. M. Al-Khateeb, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham, "Detecting recurring and novel classes in concept-drifting data streams," in *2011 IEEE 11th International Conference on Data Mining*, Dec 2011, pp. 1176–1181.

[4] S. Chandra, A. Haque, L. Khan, and C. Aggarwal, "An adaptive framework for multistream classification," in *The 25th ACM International Conference on Information and Knowledge Management (CIKM)*, Oct 2016.

[5] E. Kouloumpis, T. Wilson, and J. D. Moore, "Twitter sentiment analysis: The good the bad and the omg!" *Icwsm*, vol. 11, pp. 538–541, 2011.

[6] B. Z. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *In International Conference on Machine Learning (ICML)*, 2004, pp. 903–910.

[7] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting sample selection bias by unlabeled data," in *Advances in neural information processing systems*, 2006, pp. 601–608.

[8] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *Advances in neural information processing systems*, 2008, pp. 1433–1440.

[9] Y. Kawahara and M. Sugiyama, "Sequential change-point detection based on direct density-ratio estimation," *Stat. Anal. Data Min.*, vol. 5, no. 2, pp. 114–127, Apr. 2012.

[10] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online Learning with Kernels," *IEEE Transactions on Signal Processing*, vol. 52, pp. 2165–2176, August 2004.

[11] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "Moa: Massive online analysis, a framework for stream classification and clustering," in *Journal of Machine Learning Research*, 2010, pp. 44–50.

[12] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.