Multistream Classification with Relative Density Ratio Estimation

Bo Dong, Yang Gao, Swarup Chandra, Latifur Khan
Department of Computer Science
University of Texas at Dallas, Richardson TX
{bxd130630, yxg122530, swarup.chandra, lkhan}@utdallas.edu

Abstract

In supervised learning, availability of sufficient labeled data is of prime importance. Unfortunately, they are sparingly available in many real-world applications. Particularly when performing classification over a non-stationary data stream, unavailability of sufficient labeled data undermines the classifier's long-term performance by limiting its adaptability to changes in data distribution over time. Recently, studies in such settings have appealed to transfer learning techniques over a data stream while detecting drifts in data distribution over time. Here, the data stream is represented by two independent non-stationary streams, one containing labeled data instances (called source stream) having a biased distribution compared to the unlabeled data instances (called target stream). The task of label prediction under this representation is called *Multistream Classification*, where instances in the two streams occur independently. While these studies have addressed various challenges in the multistream setting, it still suffers from large computational overhead mainly due to frequent bias correction and drift adaptation methods employed. In this paper, we focus on utilizing an alternative bias correction technique, called relative density-ratio estimation, which is known to be computationally faster. Importantly, we propose a novel mechanism to automatically learn an appropriate mixture of relative density that adapts to changes in the multistream setting over time. We theoretically study its properties and empirically demonstrate its superior performance, within a multistream framework called MSCRDR, on benchmark datasets by comparing with other competing methods.

Introduction

When employing a supervised machine learning model for label prediction over a stream of data instances continuously generated from non-stationary processes, its prediction performance degrades with changes in data distribution over time. This problem, called concept drift (Gama et al. 2004), has been addressed in previous studies using model adaptation and drift detection techniques (Masud et al. 2011; Domingos and Hulten 2000; Gama et al. 2014). However, they assume that sufficient labeled data instances are readily available throughout the stream (Haque, Khan, and Baron 2016; Mu et al. 2017). Unfortunately, the existence of sufficient labeled instances is scarce in practice.

For instance, consider the scenario of predicting sentiment of Facebook posts (Ortigosa, Martín, and Carro 2014). Usually, the sentiment label is not provided along with a user's post. To train a classifier, true labels of a few posts from multiple users may be requested. However, if users responding with the labels are not a good representation of the population, then the classifier trained directly on such biased labeled data may not generalize well to all other user posts.

To address the challenge of limited labeled data over concept-drifting data streams, (Chandra et al. 2016) introduced a framework where a subset of independent processes continuously generate labeled data instances having a biased distribution compared to unlabeled instances generated from the population. They represent the data as two disjoint streams, one with biased labeled data and the other with unlabeled data, to leverage transfer learning techniques that aid in bias correction. This setting is called *Multistream Classification*, to emphasize the two-stream representation. The labeled stream is called *source* and the unlabeled stream is called *target*. Moreover, this representation allows for drifts in concepts to be observed independently over the two streams. The combination of concept drift and sampling bias provides a variety of unique challenges, such as data shift and asynchronous concept drifts between the source and target streams, for predicting labels of unlabeled instances in the target stream.

However, previous approaches that address the challenges of multistream setting suffer from lack of systematic model selection (Chandra et al. 2016) or have large computational time overhead (Haque et al. 2017). In a framework that perform label prediction over data streams, the speed of data consumption is typically limited by its slowest component. In existing approaches that perform multistream classification, the data processing time is limited by the bias correction technique employed. (Haque et al. 2017) uses an optimization procedure that has quadratic time complexity. This severely limits the speed of data consumption in the two streams, particularly in the target stream.

In this paper, we instead utilize a faster bias correction approach which is more suitable for multistream classification. Particularly, distribution bias is corrected by directly estimating the density ratio between source and target stream during classifier training, where the training is performed over a bias-corrected dataset. Recently, a technique that

estimates density ratio using Pearson divergence was proposed (Yamada et al. 2011) in a non-streaming environment. Here, the density ratio is a weighted proportion of target to source densities of a given fixed dataset. When applied to a data stream setting, it can be used over a fixed-size minibatch to estimate density ratio as new data arrives. However, applying a similar minibatch approach directly in a multi-stream setting introduces two main challenges. First, concept drifts in the source and target streams are asynchronous. A drift within a minibatch on either the source or target stream affects classification performance until the bias correction is updated. Second, more importantly, the weighted proportion for relative density between the source and target is unknown. This weight, denoted as α, depends on the dataset and is affected by drifts along the stream. Using a fixed α throughout the stream may not be appropriate and affects classification performance.

We address the above challenges by proposing a bias correction technique using relative density ratio estimation where weight proportion is automatically learnt and adapted along the stream. The key idea in the paper is to design a loss function useful for automatically learning α via an expectation-maximization technique from data observed in fixed-size sliding windows. We show that this technique, when used in a multistream classification framework for label prediction on the target stream called MSCRDR, not only increases computational speed, but also significantly improves prediction performance.

The main contributions are summarized as follows. (1) We utilize the α-relative density ratio to estimate importance weights associated with source data instances. (2) We propose an expectation-maximization technique to automatically learn model parameters for relative density ratio estimation from available data, which significantly improves the model performance. Moreover, we also study its theoretical properties useful to demonstrate its effectiveness. (3) We present a technique for detecting asynchronous concept drifts between source and target streams data using relative density ratios in the multistream setting. (4) We use benchmark real-world and synthetic datasets to empirically evaluate our approach, and compare the results with baseline methods.

Background

Covariate Shift Adaptation When the distribution of labeled and unlabeled data are unequal, distribution matching techniques can be employed to correct such discrepancy. Typically, transfer learning techniques (Dai et al. 2007; Huang et al. 2007; Sugiyama et al. 2008; Zhu et al. 2011) are utilized to leverage available labeled data to perform prediction over unlabeled instances. Sampling bias or covariate shift (Shimodaira 2000) is one such setting where the data is occurring from the same domain with a bias in labeled data distribution compared to the population. A technique to perform distribution matching in such cases is to estimate importance weight associated with labeled data instances, given both labeled and unlabeled instances. Popular methods such as Kernel Mean Matching (KMM) (Huang et al. 2006), Kullback-Liebler Importance Estimation Procedure (KLIEP) (Sugiyama et al. 2008), and unconstrained Least-Squares Importance Fitting (uLSIF) (Kanamori, Hido, and Sugiyama 2009) aim to compute importance weights by directly estimating density ratio between the distributions represented by the given unlabeled and labeled data instances. Formally, if $P_S(\boldsymbol{x})$ and $P_T(\boldsymbol{x})$ denotes the distributions of source and target instances respectively, then the importance weight of an instance is given by $\beta(\boldsymbol{x}) = \frac{P_T(\boldsymbol{x})}{P_S(\boldsymbol{x})}$. Instead of estimating $P_T(\boldsymbol{x})$ and $P_S(\boldsymbol{x})$ individually, the density ratio can be directly estimated from instances.

In particular, KMM performs distribution matching by minimizing the Euclidean distance between the weighted source and unweighted target distribution. Whereas, KLIEP minimizes the KL distance between the distributions. However, both these approaches are known to have high degree of fluctuation in density ratio estimates. This affects classifier prediction. To smoothen out such fluctuations, (Yamada et al. 2011) introduced Relative Density Ratio that minimize Pearson Divergence between the distributions. Here, the relative density, denoted by $r_\alpha(\boldsymbol{x})$, for each source instance $\boldsymbol{x}$, is given by $r_\alpha(\boldsymbol{x}) = \frac{P_T(\boldsymbol{x})}{\alpha P_T(\boldsymbol{x}) + (1-\alpha) P_S(\boldsymbol{x})}$, where $\alpha \in (0, 1)$. Applying a Gaussian kernel model, $r(\boldsymbol{x})$ is modeled as:

$$\hat{r}_\alpha(\boldsymbol{x}) = \sum_{i=1}^{N} \theta_i(\alpha) K_\sigma\left(\boldsymbol{x}, \boldsymbol{W}_T^{(i)}\right) \tag{1}$$

where $\boldsymbol{\theta} = \{\theta_i(\alpha)\}_{i=1}^{N}, \alpha$ are the parameters to be learned from data. $\boldsymbol{W}_T^{(i)}$ is the i^{th} instance in target dataset $\boldsymbol{W}_T$. $K_\sigma(\boldsymbol{x}, \boldsymbol{x}') = \exp\left\{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}\right\}$ is a Gaussian kernel with kernel width σ. This mechanism is shown to be faster than KLIEP, and we adopt it to perform bias correction.

Multistream Classification In the case of a data stream, the difference between training and test distribution arises during a concept drift. Typically, concept drift over a single data stream is detected by observing changes in prediction behavior, i.e., either in prediction error (Gama et al. 2004) or classifier confidence (Haque, Khan, and Baron 2016) over the sequence of data occurring along the stream. The classifier is retrained using a newly collected training data occurring after the drift. However, this requires access to sufficient labeled data soon after drift detection.

Recent studies address a scenario where the distribution of labeled data available is biased compared to the unlabeled data along the data stream. Particularly, (Chandra et al. 2016) demonstrates the use of KMM along with drift detection in a stream setting where labeled and unlabeled data is represented as independent data streams. Their framework, called MSC, performs drift detection and bias correction over the two independent streams for data classification. Unfortunately, it suffers from the intrinsic limitation of KMM, i.e., there is no good model selection approach (Sugiyama et al. 2008). FUSION (Haque et al. 2017) addresses this issue by utilizing a version of KLIEP for bias correction. Moreover, it combines the mechanism for bias correction and drift detection by using a Gaussian kernel model for representing density ratio, with online updates. However, this frame-

work suffers from large execution overhead due to the optimization function of bias correction having quadratic time complexity. As mentioned earlier, recent studies have shown that relative density ratio estimation performs equivalent to KLIEP when the right set of parameters are known, while achieving low execution overhead for bias correction. Unfortunately, its parameter α is largely unknown. Even when naively used over a data stream, drifts along the stream over time may cause performance degradation if it is not appropriately adapted. Therefore, we propose a framework that leverages the advantages of relative density ratio with a mechanism to automatically evaluate α and adapt its value along the stream.

Preliminaries

Problem Let $\boldsymbol{x} \in \mathbb{R}^d$ be a vector representing the d-dimensional real-valued covariates of each observed data instance, and $y \in \mathcal{Y}$ be its class label, where $\mathcal{Y} = [1, k]$. Though we observe a data stream as a sequence of data instances, we only store the latest data in memory. For this purpose, we denote an observable window of size n by $\boldsymbol{W}$. Particularly for the multistream setting, we denote the current observed labeled data $(\boldsymbol{x}_S, y_S)$ by $\boldsymbol{W}_S$ and the unlabeled data $(\boldsymbol{x}_T)$ by $\boldsymbol{W}_T$. Here, the subscripts S and T denotes source and target streams respectively. Importantly, it is assumed that instances in $\boldsymbol{W}_S$ are sampled from the population in a biased manner, resulting in a biased data distribution compared to instances in $\boldsymbol{W}_T$. Formally, let $P_S(\boldsymbol{x}, y)$ denote the probability density function of source instances, and $P_T(\boldsymbol{x}, y)$ denote that of target instances. A typical form of sampling bias is covariate shift, i.e., $P_S(y|\boldsymbol{x}) = P_T(y|\boldsymbol{x})$ and $P_S(\boldsymbol{x}) \neq P_T(\boldsymbol{x})$. Under this covariate shift assumption, training a classifier directly on instances in S does not generalize well on instances in T. Therefore, a bias correction mechanism is necessary. Furthermore, a concept-drifting data stream is assumed, i.e., a drift in labeling function may occur along the stream over time. To reduce degradation of prediction performance, the classifier may need adaptation by reacting to such changes. The problem of multistream classification is to leverage the available labels in $\boldsymbol{W}_S$ to predict labels of instances in $\boldsymbol{W}_T$ by addressing the challenges of both bias correction and concept drift.

Challenges As mentioned above, we assume that at time t, data distributions of $\mathcal{S}$ and $\mathcal{T}$ are related by a covariate shift, i.e., $P_S^{(t)}(y \mid \boldsymbol{x}) = P_T^{(t)}(y \mid \boldsymbol{x})$ and $P_S^{(t)}(\boldsymbol{x}) \neq P_T^{(t)}(\boldsymbol{x})$. However, this assumption may not be hold at time $r > t$ due to the non-stationary nature of data streams. In addition, conditional probability distribution may change over time due to concept drift, i.e. $P^{(t)}(y \mid \boldsymbol{x}) \neq P^{(r)}(y \mid \boldsymbol{x})$.

The efficiency of the framework is critical for stream setting problem. Current multistream classification methods mainly suffer from expensive computation while handling covariate shift. Moreover, directly employing a faster bias correction technique such as relative density ratio estimation would not be appropriate since its unknown model parameters will have to be fixed initial. However, an adaptive model to concept drifts is more desirable for better prediction performance. We address the above challenges using our proposed framework.

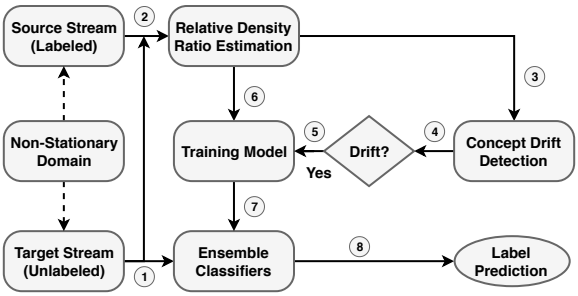

Figure 1: Overview of MSCRDR

The Proposed Approach

An overview of our framework, called MultiStream Classification using Relative Density Ratio (MSCRDR), is shown in Figure 1. Since the data instances occur continuously along the stream, we use an initial warm-up phase to begin classification, and the classifier model is subsequently adapted to changes in data distribution throughout its lifetime.

The warm-up phase consists of an initial set of instances in the source and target stream. Due to the assumption of covariate shift, we estimate α-relative density ratio associated with the source instances. Since the instance weights aid in correcting the distribution bias of source data instances, a classifier trained using the weighted labeled instances can generalize well on target instances. As a result, MSCRDR trains a classifier using the initial set of weighted source instances for label prediction on future target instances. However, concept drifts may occur along the stream. This may degrade the classifier performance over time. MSCRDR detects drifts by evaluating change points at which there exists significant difference between the relative density ratios across disjoint sequential time periods. Once a change point is detected, MSCRDR updates the bias correction model and retrains the classification model.

At the subsequent occurrence of every new instance along the target stream, the classifier prediction is used. As illustrated in Figure 1, we use an ensemble of classifiers to perform label prediction. This follows from previous studies (Street and Kim 2001; Min and Lin 2018) that demonstrate the variance reduction ability of ensemble classifiers to improve classification performance. Likewise, MSCRDR uses a fixed-size ensemble by maintaining the best performing classifiers. Consequent to drift detection, we update the ensemble by choosing the top E models based on performance. Since the labels of $\boldsymbol{W}_T$ are unavailable, the performance is measured by average confidence on $\boldsymbol{W}_T$ where the confidence score for each instance is evaluated using the probability output of SVM. If the ensemble contains less than E models currently, the new model is simply added to the ensemble. Otherwise, the least-performing model is replaced by the new model to maintain memory usage of the ensemble. These top E models employ majority voting to predict labels of instances.

Algorithm 1 Multistream Classification

Require: Labeled source stream S, unlabeled target stream T, the size of $\boldsymbol{W}_S$ and $\boldsymbol{W}_T$, N.
Ensure: Labels for target stream T.
1: $\boldsymbol{W}_S, \boldsymbol{W}_T \leftarrow readData(S, T, 1)$
2: Learn $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^{N}, \alpha$ by using Algorithm 2.
3: Estimate $r_\alpha(\boldsymbol{W}_S)$ by using Eq. 1.
4: Build initial model using $r_\alpha(\boldsymbol{W}_S)$ and $\boldsymbol{W}_S$.
5: **while** T $exists$ **do**
6: Receive a new instance $\boldsymbol{x}$
7: **if** $\boldsymbol{x}$ comes from T **then**
8: Predict $Label(\boldsymbol{x})$ by ensemble classifiers.
9: **end if**
10: Sliding $\boldsymbol{W}_S$ or $\boldsymbol{W}_T$ to include $\boldsymbol{x}$.
11: Check concept drift in stream using Eq.(9).
12: **if** There is concept drift in data stream **then**
13: Update $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^{N}, \alpha$ by Algorithm 2.
14: Update training model and ensemble classifiers.
15: Predict $Label(\boldsymbol{x})$ by ensemble classifiers.
16: **end if**
17: **end while**

Algorithm 2 Learn Parameter

Require: Source instances $\boldsymbol{W}_S$, target instances $\boldsymbol{W}_T$, the size of $\boldsymbol{W}_S$ and $\boldsymbol{W}_T$, N.
Ensure: RDRM parameters $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^{N}, \alpha$.
1: Initialize α, the learning rate ϵ, the learning rate parameter k, the iteration step i
2: Learn $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^{N}$ by using Eq. 5.
3: $loss = [\alpha-1]_+ + [-\alpha]_+ + \lambda_1(\frac{1}{2}\boldsymbol{\theta}^T\hat{\boldsymbol{H}}\boldsymbol{\theta} - \hat{\boldsymbol{h}}^T\boldsymbol{\theta} + \frac{\lambda_2}{2}\boldsymbol{\theta}^T\boldsymbol{\theta})$
4: **repeat**
5: Learn $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^{N}$ by using Eq. 5.
6: $\alpha_{old} = \alpha$
7: $\epsilon = \epsilon \exp\{-ki\}$
8: $\alpha = \alpha - \epsilon\frac{\partial loss}{\partial \alpha}$
9: **if** $\alpha \notin (0, 1)$ **then**
10: $k = 2k$
11: $\alpha = \alpha_{old}$
12: **end if**
13: $i = i + 1$
14: **until** loss converges
15: **Return** $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^{N}, \alpha$

Relative Density Ratio Estimation

The Relative Density Ratio Estimation module (RDRE) of MSCRDR uses the Gaussian kernel model for density estimation, similar to Eq. 1. This is updated as concept drift occurs along the stream. In this section, we describe the RDRE and its parameter learning procedure.

Parameter Learning We now describe our method for automatically learning the model parameter based on an expectation-maximization procedure. The parameters $\boldsymbol{\theta}$ and α in Eq. 1 are learnt so that Pearson divergence from $r_\alpha(\boldsymbol{x})$ to $\hat{r}_\alpha(\boldsymbol{x})$ would be minimized. This is defined as:

$$PE[r_\alpha(\boldsymbol{x}), \hat{r}_\alpha(\boldsymbol{x})] = \frac{1}{2}\int (\frac{r_\alpha(\boldsymbol{x})}{\hat{r}_\alpha(\boldsymbol{x})} - 1)^2 \hat{r}_\alpha(\boldsymbol{x})\, dx \quad (2)$$

Algorithm 2 describes the procedure for learning the parameters $\boldsymbol{\theta}$ and α. In order to ensure $\alpha \in (0, 1)$ during the expectation-maximization procedure, we design a soft constraint for α in $loss$ as follows:

$$[\alpha - 1]_+ + [-\alpha]_+ = \begin{cases} -\alpha & \alpha \leq 0 \\ \alpha & \alpha \geq 1 \\ 0 & 0 < \alpha < 1 \end{cases} \quad (3)$$

When we take the derivative with respect to α in $loss$ during expectation-maximization procedure. This constraint gives penalty when $\alpha \notin (0, 1)$. As indicated in Algorithm 2, in each iteration, if α out of range $(0, 1)$, α is assigned to its previous value and the learning rate decreases correspondingly. This ensures α converges in range $(0, 1)$. These lead to the objective function as follows.

$$[\alpha - 1]_+ + [-\alpha]_+ + \lambda_1(\frac{1}{2}\boldsymbol{\theta}^T\hat{\boldsymbol{H}}\boldsymbol{\theta} - \hat{\boldsymbol{h}}^T\boldsymbol{\theta} + \frac{\lambda_2}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}) \quad (4)$$

where $\theta \in \mathcal{R}^n$, and $\frac{\lambda_2}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}$ is the penalty term for the regularization. For each iteration, the parameter $\boldsymbol{\theta}$ is computed based on currant α. It yields an analytical solution, given by,

$$\boldsymbol{\theta} = \left(\hat{\boldsymbol{H}} + \lambda \boldsymbol{I}_n\right)^{-1} \hat{\boldsymbol{h}} \quad (5)$$

where $\hat{\boldsymbol{H}}$ and $\hat{\boldsymbol{h}}$ are arrays whose elements are computed as follows. Here $\boldsymbol{W}_T^{(i)}$ is the i^{th} instance in target window $\boldsymbol{W}_T$ and $\boldsymbol{W}_S^{(j)}$ is the j^{th} instance in source window $\boldsymbol{W}_S$.

$$\begin{aligned}\hat{H}_{m,m'} = &\frac{\alpha}{N}\sum_{i=1}^{N} K\left(\boldsymbol{W}_T^{(i)}, \boldsymbol{W}_T^{(m)}\right) K\left(\boldsymbol{W}_T^{(i)}, \boldsymbol{W}_T^{(m')}\right) \\ &+ \frac{1-\alpha}{N}\sum_{j=1}^{N} K\left(\boldsymbol{W}_S^{(j)}, \boldsymbol{W}_T^{(m)}\right) K\left(\boldsymbol{W}_S^{(j)}, \boldsymbol{W}_T^{(m')}\right)\end{aligned} \quad (6)$$

$$\hat{h}_m = \frac{1}{N}\sum_{i=1}^{N} K\left(\boldsymbol{W}_T^{(i)}, \boldsymbol{W}_T^{(m)}\right) \quad (7)$$

For the whole EM procedure, we first initialize α and compute $\boldsymbol{\theta}$ by Eq. 5 based on instances in $\boldsymbol{W}_S$ and $\boldsymbol{W}_T$. Next, for each iteration, we recompute $\boldsymbol{\theta}$ based on current α. Then α is adjusted using the gradient of the loss function. We repeat this until convergence. Once the set of parameters $\boldsymbol{\theta}$ and α are learned from data instances, importance weight for any instance $\boldsymbol{x}$ is calculated using Eq. 1. By estimating the optimum parameter values, the α-relative density ratio estimate $\hat{r}_\alpha(x)$ can be obtained. We utilize this methodology for bias correction in multistream classification.

Concept Drift Detection

We now present a technique for detecting asynchronous concept drifts between source and target streams data using relative density ratios in the multistream setting. The classifier is updated following a concept drift, i.e. when there is a significant difference between $P_T(\boldsymbol{x})$ and $\hat{r}_\alpha(\alpha P_T(\boldsymbol{x}) + (1 -$

$\alpha)P_S(\boldsymbol{x}))$. Let $\hat{r}^0_\alpha$ and $\hat{r}^t_\alpha$ be relative density ratios at initial time and time t respectively. Following Eq. 1, we can get,

$$\hat{r}^0_\alpha(\boldsymbol{W}^{(i)}_T) = \sum_{j=1}^{N} \theta^0_j K_\sigma(\boldsymbol{W}^{(i)}_T, \boldsymbol{W}^{(j)}_T) \tag{8}$$

where $i = 1, \ldots, N$. Similarly, we can compute $\hat{r}^t_\alpha(\boldsymbol{W}^{(i)}_T)$ by using current parameters $\left\{\theta^t_j\right\}_{j=1}^N$ at time t. Following likelihood ratio measures the deviation of the weighted source distribution from the target distribution at time t. We calculate the sum of log likelihood ratios as follows

$$S = \sum_{i=1}^{N} \ln \frac{P_T}{\hat{r}^0_\alpha(\alpha P_T + (1-\alpha)P_S)} = \sum_{i=1}^{N} \ln \frac{\hat{r}^t_\alpha(\boldsymbol{W}^{(i)}_T)}{\hat{r}^0_\alpha(\boldsymbol{W}^{(i)}_T)} \tag{9}$$

Using Eq.9, the maximum score S can be calculated. Then S is compared with threshold (defined as $-ln(\tau)$, τ is a user defined parameter). If S is equal or larger than this threshold, a change point is detected. Otherwise, there is no concept drift in data stream.

Theoretical Analysis

In this section, we analyze the convergence rate of Algorithm 2, and the overall time complexity of MSCRDR.

Convergence Rate

Lemma 1. *$loss$ is a convex function, i.e., $loss((\alpha_1 + \alpha_2)/2) \leq \frac{1}{2}(loss(\alpha_1)+loss(\alpha_2))$ for all α on interval (0,1).*

Proof. For any $\alpha \in (0,1)$, $[\alpha - 1]_+ + [-\alpha]_+ = 0$ as we defined in Eq. 3. Then, the function $loss$ is given by

$$loss = \lambda_1(\frac{1}{2}\boldsymbol{\theta}^T \hat{\boldsymbol{H}} \boldsymbol{\theta} - \hat{\boldsymbol{h}}^T \boldsymbol{\theta} + \frac{\lambda_2}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}) \tag{10}$$

The first derivative with respect to α, results in the following:

$$\frac{\partial loss}{\partial \alpha} = \frac{1}{2}\lambda_1 \boldsymbol{\theta}^T \frac{\partial \hat{\boldsymbol{H}}}{\partial \alpha} \tag{11}$$

From Eq. 6, we can see, the first derivative $\frac{\partial \hat{\boldsymbol{H}}}{\partial \alpha}$ is a constant with respect to α. Thus, the second derivative $\frac{\partial^2 loss}{\partial \alpha^2}$ becomes to 0 in each iteration. Thus, $loss$ is convex. □

Theorem 1. *Let the gradient of $loss$ function be Lipschitz continuous with constant $L > 0$, i.e. we have that $||\nabla loss(\alpha_1) - \nabla loss(\alpha_2)||_2 \leq L||\alpha_1 - \alpha_2||_2$ for any α_1, α_2. Then if we run EM for l iterations with adaptive gradient step size of $\epsilon_l \leq \frac{1}{L}$, it will yield a solution $loss_l$ which satisfies*

$$loss(\alpha_l) - loss(\alpha^*) \leq \frac{||\alpha_0 - \alpha^*||_2^2}{2\epsilon_{min} l} \tag{12}$$

where $loss(\alpha^*)$ is the optimal value.

Proof. In our approach, when $\alpha \in (0,1)$, $loss$ function is convex and differentiable, as shown in Lemma 1. In i^{th} iteration of the gradient process, let

$$\alpha' = \alpha - \epsilon_i \nabla loss(\alpha) \tag{13}$$

Combining the Taylor expansion of $loss$ with Eq. 13 we have:

$$\begin{aligned} loss(\alpha') &\leq loss(\alpha) + \nabla loss(\alpha)(\alpha' - \alpha) + \frac{1}{2}L||\alpha' - \alpha||_2^2 \\ &= loss(\alpha) - (1 - \frac{1}{2}L\epsilon_i)\epsilon_i ||\nabla loss(\alpha)||_2^2 \end{aligned} \tag{14}$$

Since we assume $\epsilon_i \leq \frac{1}{L}$, we have

$$-(1 - \frac{1}{2}L\epsilon_i) \leq -\frac{1}{2} \tag{15}$$

Using Eq. 15 and Eq. 14, we can get

$$loss(\alpha') \leq loss(\alpha) - \frac{1}{2}\epsilon_i ||\nabla loss(\alpha)||_2^2 \tag{16}$$

We can bound $loss(\alpha')$, i.e. the objective value at the next iteration, in terms of $loss(\alpha^*)$, i.e., the optimal objective value.

$$loss(\alpha') - loss(\alpha^*) \leq \frac{1}{2\epsilon_i}(||\alpha - \alpha^*||_2^2 - ||\alpha' - \alpha^*||_2^2) \tag{17}$$

This inequality holds for α' on every iteration of gradient descent. Summing over iterations, we have,

$$\sum_{i=0}^{l}(loss(\alpha_i) - loss(\alpha^*)) \leq \frac{1}{2\epsilon_{min}}(||\alpha_0 - \alpha^*||_2^2) \tag{18}$$

where $\epsilon_i = \epsilon_0 e^{-\frac{i(i+1)}{2}k}$ and $\epsilon_{min} = \min\{\epsilon_i\}$, $i = 0, 1, ..., l$. Since $loss$ function decreases on every iteration, we can get,

$$\begin{aligned} loss(\alpha_l) - loss(\alpha^*) &\leq \frac{1}{l}\sum_{i=1}^{l}(loss(\alpha_i) - loss(\alpha^*)) \\ &\leq \frac{||\alpha_0 - \alpha^*||_2^2}{2\epsilon_{min} l} \end{aligned} \tag{19}$$

Therefore, based on above derivation, the convergence rate for Algorithm 2 is $\mathcal{O}(\frac{1}{\epsilon_{min} l})$. □

Time Complexity

Time complexity of MSCRDR mainly depends on RDRE and CDD components. RDRE aids in learning the parameters of α relative density ratio (Algorithm 2). N is denoted as source (and target) window size. For each iteration of EM process in Algorithm 2, the running time is $\mathcal{O}(N^2)$ due to the computation of the Gaussian kernel matrix which is centered by instances in target window. The running time of the whole EM procedure is $\mathcal{O}(lN^2)$, where l is the iteration times of EM procedure. Time complexity of CDD is $\mathcal{O}(N)$. Time complexity of classification and update depends on the learning algorithm used as the base model. Therefore, MSCRDR has a total time complexity of $\mathcal{O}(lN^2) + f(N)$, where $f(N)$ is the time complexity for training a new model.

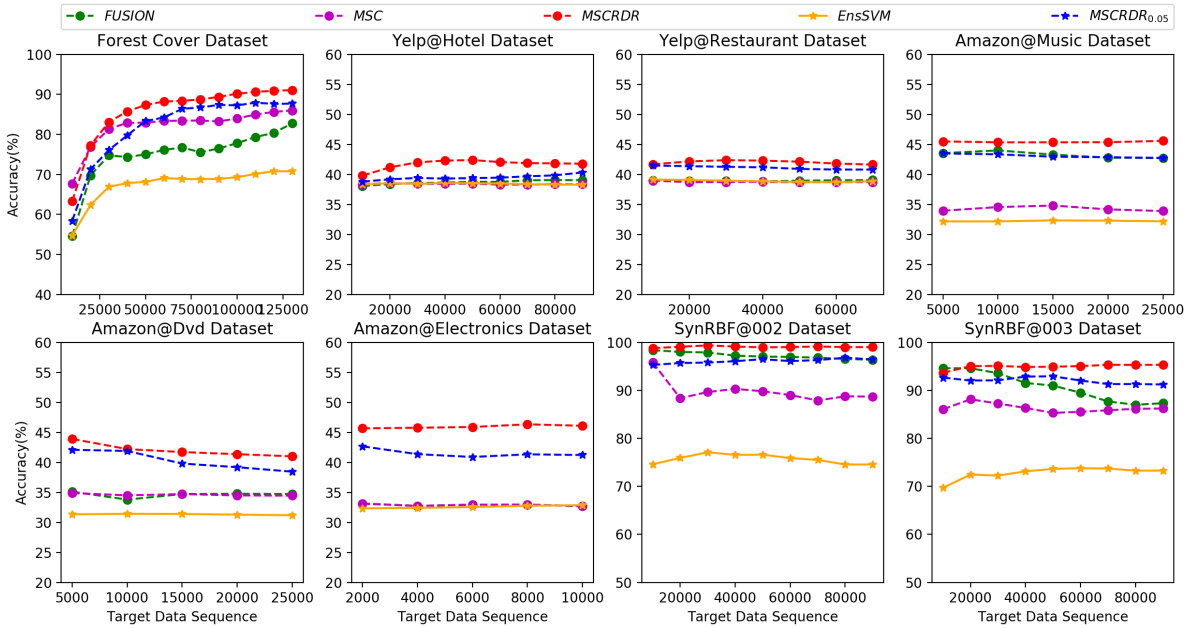

Figure 2: Accuracy along T for datasets by competing methods on multistream classification.

Table 1: Characteristics of Data Sets

Dataset	#features	#classes	#instances
Forest Cover	54	7	150,000
Yelp@Hotel	300	5	109,548
Yelp@Restaurant	300	5	80,938
Amazon@Music	300	5	29,877
Amazon@Dvd	300	5	31,411
Amazon@Electronics	300	5	11,667
SynRBF@002	70	7	100,000
SynRBF@003	70	7	100,000

Experimental Evaluation

Datasets

Table 1 lists the real-world and synthetic datasets used to evaluate our approach.

Real World Datasets *Forest Cover* is a benchmark dataset from the UCI repository[1] containing geospatial descriptions of different forest types. The labeling task is to predict the actual forest cover type for each given instance. *Yelp@X, Amazon@X* are real-word datasets containing customer reviews taken from Yelp.com and Amazon.com. Here, X denotes the category of the dataset. *Yelp@X* dataset (Shrestha, Mukherjee, and Solorio 2018) contains the customer id, reviews and ratings for different Hotel/Restaurant on Yelp website. *Amazon@X* dataset (Blitzer, Dredze, and Pereira 2007) contains the timestamps, reviews and ratings for different Music/Dvd/Electronics products on Amazon website. The task is to predict ratings of each review.

To extract features from raw review text data, we use word2vec (Mikolov et al. 2013) model that was pre-trained on 100 billion words from Google to convert the words to vectors with size 300. For each text, we follow sentence embedding (Arora, Liang, and Ma 2017) to perform a weighted sum of word vectors generated by word2vec, and remove principal components from the weighted vector computed in previous step to yield the corresponding sentence vector.

[1]http://archive.ics.uci.edu/ml/datasets/covertype

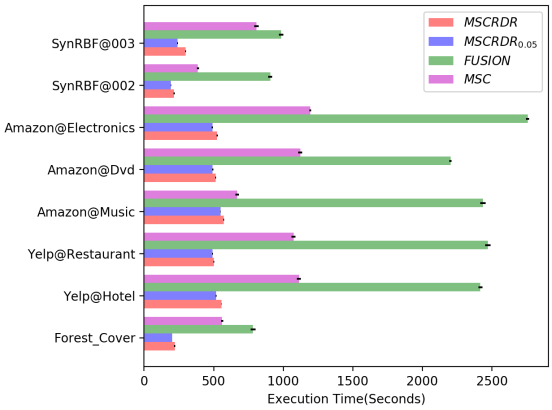

Figure 3: Execution time of multistream classification.

Synthetic Datasets *SynRBF@X* are synthetic datasets generated using *RandomRBFGeneratorDrift* of MOA (Bifet et al. 2010) framework, where X is the speed of change of centroids in the model. We generate two such datasets using $X = \{0.002, 0.003\}$ to evaluate the approaches on concept drifts having various intensities and frequencies. We normalize all the datasets used, and reshuffle the instances from dif-

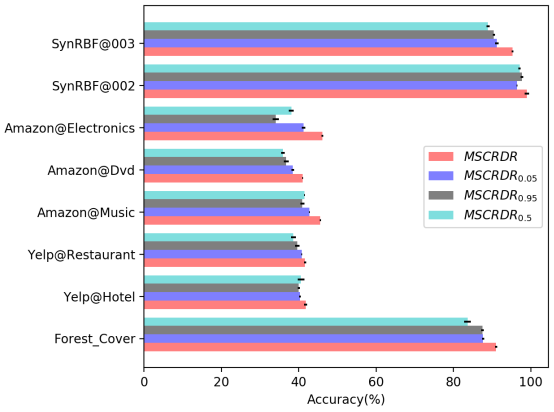

Figure 4: Accuracy comparison.

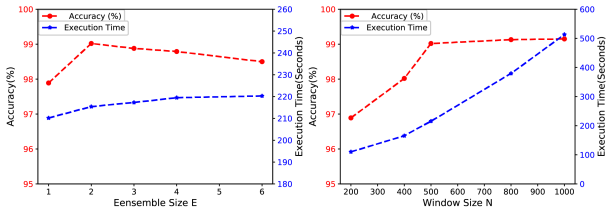

Figure 5: Parameter sensitivity of MSCRDR on SynRBF@002 dataset as an example.

ferent classes randomly to remove novel classes from them.

Generating Stream For each dataset, we assume that each feature follows a beta distribution. For all features, the distribution is evaluated by a joint beta distribution. We divide the data with a ratio of 1:9, into two disjoint sets of instances based on the value of probability density function (pdf). One set with relatively smaller pdf is considered as the source stream, the other is the target stream. Thus, the source and target streams have covariate shift caused by sampling bias.

Baseline Methods

The first two baseline methods include the multistream classification frameworks (MSC (Chandra et al. 2016) and FUSION (Haque et al. 2017)). Concretely, MSC and FUSION are competing multistream classification frameworks that leverages KMM (Huang et al. 2006) and KLIEP (Sugiyama et al. 2008) for bias correction and drift detection. These are useful to show the effectiveness of using α-relative density ratio estimation to handle covariate shift and concept drift.

Since we propose an algorithm which performs parameter learning while computing the relative density ratio, we design the third baseline, denoted as MSCRDR$_X$, where $X \in [0.05, 0.5, 0.95]$. Here we remove the learning parameter algorithm in our approach and fix the parameter α with values 0.05, 0.5, 0.95 respectively. It is designed for showing the effectiveness of Algorithm 2.

All the above baseline methods and our method use weighted SVM for classification since it provides confidence value for each prediction. In addition, all methods are designed with ensemble models. Therefore, we use an ensemble of Support Vector Machines (SVM) as the fourth baseline method where training is performed over unweighted source instances. Denoted as EnsSVM. It demonstrates the effects of bias in limited labeled data and concept drift with regard to classification performance. Parameters of all baselines were set based on cross validation on the initialization set (warm up phase) of each dataset.

Results

Performance As shown in Figure 2, our proposed approach MSCRDR has the highest accuracy compared to other four baseline methods on all datasets. Meanwhile, EnsSVM have relatively low accuracy both on real world and synthetic datasets. For example, the accuracy of EnsSVM on *ForestCover* dataset is 70.77%. Whereas the accuracy of MSCRDR, MSCRDR$_{0.05}$, FUSION and MSC are 91.04%, 87.69%, 82.73% and 85.98% respectively. This clearly shows that covariate shift correction is necessary for better label prediction under sampling bias. Since EnsSVM do not consider covariate shift and concept drift in data stream, its classification performance is directly affected. MSC lacks good model selection process which affect the classification performance. FUSION apply direct density ratio for bias correction which may not converge to global maximum. MSCRDR has better performance since it overcome the above limitations.

Figure 4 shows that MSCRDR performs better on all datasets compared to MSCRDR$_X$, $X \in [0.05, 0.5, 0.95]$. This is due to the parameter learning method (Algorithm 2), which is superior to a method that used a fixed parameter value. When updating the classification model, we learn the optimum value of parameter α each time, which ensure classification performance. Thus, for overall evaluation, MSCRDR out performs MSCRDR$_X$ with fixed parameter.

Importantly, Figure 3 indicates that MSCRDR requires less execution time compared to FUSION and MSC. FUSION takes the longest execution time on all datasets compare to other methods in Figure 3 due to (1) the optimization procedures for FUSION have high non-linearity of the objective functions to be optimized. (2) when FUSION build the Gaussian kernel model, it uses cross validation to search the optimal parameters, which make the procedure slow. MSC suffers from its concept drift detection component which has cubic time complexity. Our proposed approach is competitive to FUSION and MSC. Since the optimization procedure not only introduce non-linearity of the objective function compare with FUSION, but also largely reduces the time complexity of bias correction. We utilize relative density ratios for drift detection which has linear time complexity. The result indicates that MSCRDR adapts to covariate shift and data drifts more efficiently than all baselines.

Parameter Sensitivity Figure 5 indicates that the accuracy increases with the window size. However, the execution time also increases with increasing maximum window size. This is expected since the execution time of MSCRDR is related to window size N. Figure 5 demonstrates the number

of classifiers in ensemble E affect the accuracy and execution time on *SynRBF@002* dataset. When $E = 2$, the accuracy achieves the highest value. After that, the accuracy does not change much. The execution time increases when the ensemble size increases. For this dataset, we choose $E = 2$.

Conclusion

We propose an efficient approach to perform data classification over a multistream setting. Particularly, we propose an algorithm which automatically learn parameters for relative density ratio estimation to address covariate shift and asynchronous concept drift in source and target streams. Further, we study its theoretical properties. Empirical results indicate MSCRDR performs significantly better than the baselines.

Acknowledgment

This material is based upon work supported by NSF award number DMS-1737978, AFOSR award number FA9550-14-1-0173, IBM faculty award (Research), HP grant and NSA.

References

Arora, S.; Liang, Y.; and Ma, T. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

Bifet, A.; Holmes, G.; Kirkby, R.; and Pfahringer, B. 2010. Moa: Massive online analysis. *J. Mach. Learn. Res.* 11:1601–1604.

Blitzer, J.; Dredze, M.; and Pereira, F. 2007. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *In ACL*, 187–205.

Chandra, S.; Haque, A.; Khan, L.; and Aggarwal, C. 2016. An adaptive framework for multistream classification. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 1181–1190. ACM.

Dai, W.; Xue, G.-R.; Yang, Q.; and Yu, Y. 2007. Transferring naive bayes classifiers for text classification. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1*, AAAI 2007, 540–545. AAAI Press.

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

Gama, J.; Medas, P.; Castillo, G.; and Rodrigues, P. P. 2004. Learning with drift detection. In Bazzan, A. L. C., and Labidi, S., eds., *SBIA*, volume 3171 of *Lecture Notes in Computer Science*, 286–295. Springer.

Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; and Bouchachia, A. 2014. A survey on concept drift adaptation. *ACM computing surveys (CSUR)* 46(4):44.

Haque, A.; Wang, Z.; Chandra, S.; Dong, B.; Khan, L.; and Hamlen, K. W. 2017. Fusion: An online method for multistream classification. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, 919–928. New York, NY, USA: ACM.

Haque, A.; Khan, L.; and Baron, M. 2016. Sand: Semi-supervised adaptive novel class detection and classification over data stream. In *AAAI*, 1652–1658.

Huang, J.; Smola, A. J.; Gretton, A.; Borgwardt, K. M.; and Scholkopf, B. 2006. Correcting sample selection bias by unlabeled data. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, 601–608. Cambridge, MA, USA: MIT Press.

Huang, J.; Gretton, A.; Borgwardt, K. M.; Schölkopf, B.; and Smola, A. J. 2007. Correcting sample selection bias by unlabeled data. In Schölkopf, B.; Platt, J. C.; and Hoffman, T., eds., *Advances in Neural Information Processing Systems 19*. MIT Press. 601–608.

Kanamori, T.; Hido, S.; and Sugiyama, M. 2009. A least-squares approach to direct importance estimation. *J. Mach. Learn. Res.* 10:1391–1445.

Masud, M.; Gao, J.; Khan, L.; Han, J.; and Thuraisingham, B. M. 2011. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering* 23(6):859–874.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Min, Z., and Lin, D. 2018. Probabilistic ensemble of collaborative filters. In *AAAI*. AAAI Press.

Mu, X.; Zhu, F.; Du, J.; Lim, E.; and Zhou, Z. 2017. Streaming classification with emerging new class by class matrix sketching. In *AAAI*, 2373–2379. AAAI Press.

Ortigosa, A.; Martín, J. M.; and Carro, R. M. 2014. Sentiment analysis in facebook and its application to e-learning. *Comput. Hum. Behav.* 31:527–541.

Shimodaira, H. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference* 90(2):227–244.

Shrestha, P.; Mukherjee, A.; and Solorio, T. 2018. Large scale authorship attribution of online reviews. In Gelbukh, A., ed., *Computational Linguistics and Intelligent Text Processing*, 221–232. Cham: Springer International Publishing.

Street, W. N., and Kim, Y. 2001. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, 377–382. New York, NY, USA: ACM.

Sugiyama, M.; Suzuki, T.; Nakajima, S.; Kashima, H.; von Bünau, P.; and Kawanabe, M. 2008. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics* 60(4):699–746.

Yamada, M.; Suzuki, T.; Kanamori, T.; Hachiya, H.; and Sugiyama, M. 2011. Relative Density-Ratio Estimation for Robust Distribution Comparison. *ArXiv e-prints*.

Zhu, Y.; Chen, Y.; Lu, Z.; Pan, S. J.; Xue, G.; Yu, Y.; and Yang, Q. 2011. Heterogeneous transfer learning for image classification. In *AAAI*. AAAI Press.