COMC: A Framework for Online Cross-domain Multistream Classification

Hemeng Tao, Zhuoyi Wang, Mahmoud Zamani, Latifur Khan
Department of Computer Science, The University of Texas at Dallas, Richardson TX, USA
{hxt160430, zxw151030, mxz173130, lkhan}@utdallas.edu

***Abstract*—With the tremendous increase of the online data, training a single classifier may suffer because of the large variety of data domains. One solution could be to learn separate classifiers for each domain. However, this would arise a huge cost to gather annotated training data for a large number of domains and ignore similarity shared across domains. Hence, it leads to our problem setting: can labeled data from a related source domain help predict the unlabeled data in the target domain? In this paper, we consider two independent simultaneous data streams, which are referred to as the source and target streams. The target stream continuously generates data instances from one domain where the label is unknown, while the source stream continuously generates labeled data instances from another domain. Most likely, the two data streams would have different but related feature spaces and different data distributions. Moreover, these streams may have asynchronous concept drifts between them. Our problem setting, which is called Cross-domain Multistream Classification, is to predict the class labels of data instances in the target stream using a classifier trained on the labeled source stream. In this paper, we propose an efficient solution for cross-domain multistream classification by integrating change detection into online data stream adaptation. The class labels of data instances in the target stream are predicted using the sufficient amount of label information in the related source stream. And the concept drifts along the two independent streams are continuously being addressed at the same time. Experimental results on real-world data sets indicate significantly improved performance over baseline methods.**

I. Introduction

Data stream mining has attracted researchers due to its proliferation in today's social networks. Efficient extraction of knowledge from these streams may help in tackling important decisions in near-real-time opportunities. Recent studies in data stream classification [19], [14], [26] have proposed multiple techniques that adapt a prediction target function to changes in the data distribution over time. This change is referred to as Concept Drift [6]. However, the large variety of streaming data makes it difficult and costly to annotate training data for different domains. In travel inspiration task, for example, where the goal is to recommend the best destination for travelers according to their budget, length of trip and their historical cruise records, it may be infeasible to collect training data from all over the world because of calendar differences among countries. Specific vacations in different countries hold in a different period of time like Chinese new year ceremony around February while it is difficult to collect labeled stream data from other countries on February.

Identify applicable funding agency here. If none, delete this.

The cost of gaining true labels for streaming data leads us to a practically important problem: can the labeled data from another related data stream help predict the target task, even if they are generated from different feature domains and have different data distributions? To solve the difficulty mentioned above on obtaining labeled travelers information from all of the countries, we might want to adopt a learning system trained on some countries with excessive labels (the source domain) which is applicable for other countries (the target domain). However, the customer information such as currencies and destination distances may be vary from country to country or continent to continent, which will affect the accuracy of predication model. Most travelers prefer to visit closer destinations with less costs. For instance, Alaska is a very popular visiting place among Americans because of the short distance and low cost. If we directly apply the learning model trained from American travelers, it will recommend Asians to Alaska, which most of time is not reasonable.

In this paper, we perform classification in a setting involving two types of non-stationary data streams in different domains. A stream of labeled data instances is generated by a non-stationary process, which is referred to as the Source Stream. Another stream of data instances is assumed to be generated by an independent non-stationary process, which is referred to as the Target Stream. We call this Cross-domain Multistream Classification. Moreover, existing state-of-the-art techniques [7] can effectively detect drift and adapt the model to changes in data distribution over time. However, such approaches assume that the source and target streams share the same domain, i.e., features of the source stream exactly correspond to those in the target stream. In reality this condition often fails to hold. In this case, the existing approaches will suffer.

The main challenge in cross-domain multistream classification is to effectively unify the feature spaces and data distributions between the source and target streams, while continuously addressing concept drifts along the streams. Naïvely combining the two streams into a single stream and employing existing prediction techniques is not effective since the combined stream represents an overall data distribution, which attenuates the effect of individual drifts and may adversely affects prediction performance. Combining streams with different features affect accuracy more than distribution being attenuated. We address these challenges by utilizing Empirical Maximum Mean Discrepancy (EMMD) [21] to train an

adapted classifier. In addition, the proposed approach estimates the average mean discrepancy by the same EMMD model for detecting any change between distributions represented by adapted source and target stream data. The main contributions of our work are as follows. (1) We perform classification in a cross-domain multistream setting by utilizing the adapted source stream data to train a prediction model, which is used over the adapted target stream data. (2) We propose an online classification model which utilizes an efficient technique that simultaneously performs domain adaptation and asynchronous concept drift detection between source and target stream data. In particular, we utilize mean discrepancy for continuously adapting the source and target stream to a new feature representation, whose output is used for drift detection and model updating. (3) We evaluate our technique on both synthetic and real-world data sets, and compare the results with baseline methods.

The paper is organized as follows. We first discuss related studies and background in data stream mining and transfer learning, and then list the set of challenges in the cross-domain multistream classification problem. Next, we present our framework to address these challenges. Finally, we empirically evaluate the framework on numerous data sets and conclude.

II. Related Work

There are mainly two areas that relate to our work in this paper. The first is stream classification, and the other is domain adaptation. In the following sections, we will explore these two areas accordingly.

A. Stream Classification

Data stream classification is a challenging task due to its inherent properties, such as infinite length and concept drift. The infinite length problem is addressed by dividing the whole stream into fixed-size minibatches [15] or using a gradual forgetting mechanism [12], [22]. Recent approaches [7] address this problem by remembering only the instances within two consecutive concept drifts using a dynamically-sized minibatch.

Concept drift detection in multivariate data streams concentrates on tracking any changes in the posterior class distribution, $P(y|x)$ [17]. Instead of tracking changes in $P(y|x)$ directly, the approaches proposed in [5] adopt the principle [25] to detect this change indirectly by tracking drift in the error rate of the underlying classifier. However, tracking drift in the error rate requires true labels of test data instances, which are scarce in practice. Recent studies focus on a feedback mechanism using confidence score [3] to detect changes in distribution between two different time windows by detecting change points in the process. Apart from working on a single stream, these methods explicitly detect change points.

The problem setting described in this paper is motivated by a Multistream Classification framework [3]. It proposes a solution which focuses on the classification problems when there are asynchronous concept drifts on the source and target streams. The stream classification is a continuous process and data in the target stream is assumed to be generated with very few labels. There is a strong assumption made for the proposed solutions: features have a strict one to one correspondence from the source stream to the target stream. But this assumption may not always hold in real world scenarios. Considering the simple case of training a sentiment analyzer about two kinds of products: kitchen appliances and DVDs. One set of reviews would contain adjectives such as "reliable" or "sturdy", and the other "horrific" or "hilarious", resulting in incomparable feature spaces between training and test data. Thus, our proposed approach avoids this strong assumption.

TABLE I
List of Symbols

Symbols	Description
S, T	source / target stream
D_S, D_T	source / target domain
K, K'	original / projected feature bases
$\mathbf{P}$	projection matrix
$\mathcal{W}_S$, $\mathcal{W}_T$	source / target window
$\mathcal{W}_S^{(i)}$,$\mathcal{W}_T^{(j)}$	$i-$/ $j-$th instance in source / target window
$\mathbf{W}$	data matrix
N_m	window size
$\mathbf{M}_l$	MMD matrix, $l \in \{0, 1, ..., L\}$

B. Data Domain Adaptation

Direct access to vast quantities of labeled data for the task of the target domain is either costly or otherwise nonexistent, but labels are readily available for related source domain. Domain adaptation tackles this problem by transferring knowledge from a label-rich domain to a label-scarce domain. However, each domains data instances have different characteristics, which makes the problem difficult to solve. The goal of domain adaptation is to construct a model that works well on target domain despite the absence of their labels during training process.

Recently, several approaches have been proposed to learn a common feature representation for domain adaptation [1], [4]. [11] proposed a simple heuristic nonlinear mapping function to map the data from both source and target domains to a high-dimensional feature space. [20], [21] utilize a new dimensionality reduction method, called Maximum Mean Discrepancy Embedding (MMDE), for domain adaptation. They try to learn a set of common transfer components underlying both domains such that the difference in distributions of data in the different domains, when projected onto this subspace, can be reduced.

These state-of-the-art methods mostly address domain adaptation problems for stationary data. Thus, fitting domain adaptation frameworks to online data streams is a challenging problem yet to be explored.

III. Problem Formulation

In this section, we formalize the Cross-domain Multistream Classification problem and describe the challenges of performing classification over drifting data streams in this context.

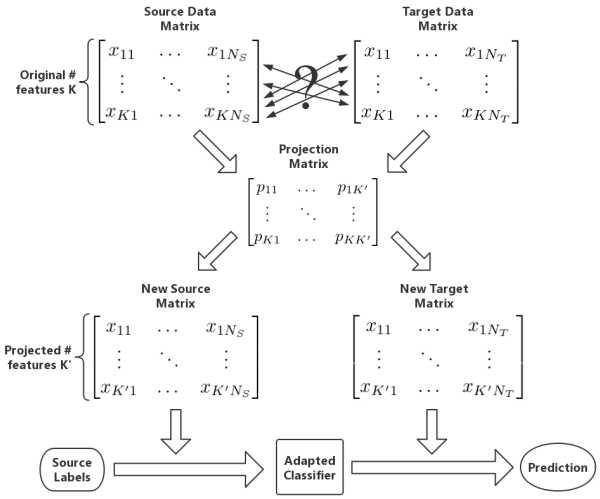


Fig. 1. Basic Idea of Domain Adaptation

A. Notation

Table 1 lists frequently used symbols in this paper. A bold symbol is used to denote a matrix, and a superscript to indicate the index of an element in the set. A calligraphic symbol is used to denote a set of elements, and a subscript to indicate the association of an element or a set to a type such as source or target stream.

B. Problem Statement

Let us consider two independent processes generating data continuously from different but related domains $D_S = \{\mathcal{X}, P_S(x)\}$ and $D_T = \{\mathcal{X}, P_T(x)\}$. A data instance, denoted by (x, y), is a vector of k covariates, and $y \in 1, ..., L$ is its corresponding class label. Here, L is the number of classes. The first process operates in a supervised environment, i.e., all the data instances that are generated from the first process are labeled. On the contrary, the second process generates unlabeled data from a different domain. The streams of data generated from the above processes are called the source stream and the target stream and are denoted by S and T respectively. The Cross-domain Multistream Classification is denoted as follows.

Definition III.1. Let $\{(x_S^{(i)}, y_S^{(i)})\}_{i=1}^{n_S}$ be a set of labeled instances from a non-stationary stream generated from the source domain D_S, where $x_S^{(i)} \in \mathcal{X}_S$ and $y_S^{(i)} \in \mathcal{Y}_S$. Similarly, let $\{(x_T^{(i)})\}_{i=1}^{n_T}$ be a set of unlabeled instances from another independent non-stationary stream generated from the target domain D_T, where $x_T^{(i)} \in \mathcal{X}_T$. $\mathcal{X}_S$ and $\mathcal{X}_T$ are different but related. The target task is assumed to be same as the source task. Construct a classifier that predicts the class label of $x_T^{(i)} \in \mathcal{X}_T$ using $\mathcal{X}_S$, $\mathcal{Y}_S$ and $\mathcal{X}_T$.

C. Challenge

The source stream is generated from a different feature space than the target stream. The problem of unifying different feature spaces is most challenging. As shown in Figure 1, even if the source and target instances have the same number of features, there need not be any strict correspondence between the two feature spaces.

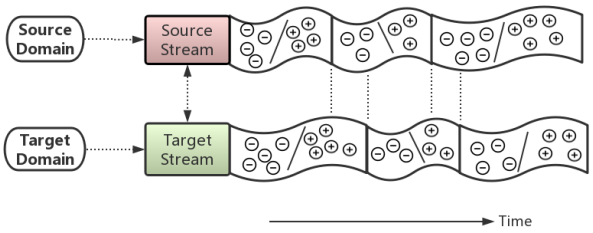


Fig. 2. Asynchronous drifts illustration

Previous works [8], [9], [3] on multistream classification only consider sampling bias between the source and target stream by assuming the two streams are generated from the same domain. Generally, a bias correction technique is applied by estimating the density ratios between the source and the target distributions. Then an importance weight is assigned to each source data point to train a corrected classifier, addressing the so-called covariate shift [24]. This paper involves domain adaptation on the multistream setting for the first time considering the scarcity of labeled data in a specific domain.

Moreover, since data is generated continuously from non-stationary processes, it is impossible to train a classifier in the traditional manner that requires storing the entire data stream in memory. Individually within each data stream, the conditional probability distribution may change over time due to concept drift, i.e. $P^{(t)}(y|x) \neq P^{(r)}(y|x)$. Similarly, a covariate drift, i.e. $P^{(t)}(x) \neq P^{(r)}(x)$, may also occur with time in each stream.

With two independent non-stationary processes generating data continuously, the effect of a drift may be observed at different times on these streams, referred to as asynchronous drift. Figure 2 illustrates asynchronous drift, this type of scenario may occur in a real-world application when the factor causing a drift affects the streams at different times. Asynchronous drifts could not only affect the accuracy of the prediction model but also affect the cross-domain adaptation process. In particular, the common feature representation learned between the target and source domain may change over time due to the covariate drift especially. Intuitively, this can be overcome by re-estimating a new feature representation and training a new cross-domain adapted classifier.

IV. Proposed Approach

We now describe our approach which integrates concept drift detection with online data distribution adaption among the two streams of data occurring over different domains. We refer this as COMC (CrOss-domain Multistream Classification). An overview of the proposed approach is illustrated in Figure 3. It has three modules, the Joint Distribution Adaptation Module

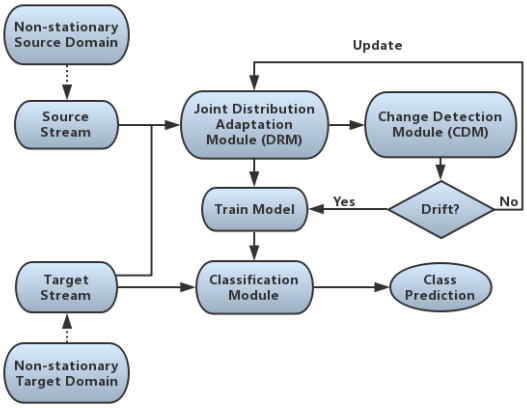


Fig. 3. Overview of COMC

(JDM), the Change Detection Module (CDM), and the Classification Module.

Algorithm 1 COMC: Cross-domain Multistream Classification

Input: Labeled source stream data S, unlabeled target stream T, The size of sliding window N_m.
Output: Labels predicted on T.

```
Read the first N_m instances from S and T into W_S and W_T respectively.
Learn projection matrix P ∈ R^{K×K'} using LearnMat.
Get the new presentation of instances in the two sliding window {W^(i)_newS = P^T W^i_S}_{i=1}^{N_m}, {W^(j)_newT = P^T W^j_T}_{j=1}^{N_m}.
Calculate initial mean discrepancy Disc_0 using left-hand side of Eq. (3).
Learn a initial classifier from W_newS.
repeat
    Receive a new instance x.
    if x ∈ T then
        Get the new representation x_new = P^T x.
        Predict the label of x using the latest classifier.
    end if
    Slide the corresponding window (W_S or W_T ) for
    including the new instance
    Check for any drift in data using ChangeDetection
    if DetectDrift returns True then
        Learn P using LearnMat.
        Update the classifier and Disc_0 by Eq. (2).
    end if
until T exists
```

Due to the theoretically infinite length of the data streams, we use two fixed-size sliding windows to observe recent instances, one window for source S, denoted by W_S, and the other for target T, denoted by W_T. The size of the sliding windows is denoted by N_m.

Data instances are continuously generated from processes in source and target domains. The pseudo-code is described in Algorithm 1. Initially, we perform joint distribution adaptation by iteratively learning an adaptation matrix for source and target data instances in the Joint Distribution Adaptation Module. As each new instance arrives, we transform the instance to the new feature representation using the adaptation matrix. The CDM detects a change point if there is a significant difference between the training and the test distribution, which is represented by data in the sliding windows using MMD. Once a change point is detected, a new classifier is trained on current source window W_S along with the updated adaptation matrix. The model is used to predict the labels of adapted data instances in T. We now present each module in detail.

A. Joint Distribution Adaptation Module (JDM)

The JDM uses a principled dimensionality reduction procedure to construct the feature representation. It jointly adapts distribution differences on the source and target streams. We now describe JDM in detail.

1) Dimensionality Reduction: Dimensionality reduction methods can learn a transformed feature representation of a data set consisting of many variables independent of each other while retaining the variation present in the dataset.

At a particular time, let $\mathcal{W}_S^{(i)}$ and $\mathcal{W}_T^{(i)}$ denote the i^{th} data instance in the source and target sliding window respectively, $\mathbf{W} = [\mathcal{W}_S^1, ..., \mathcal{W}_S^{N_S}, \mathcal{W}_T^1, ..., \mathcal{W}_T^{N_T}] \in \mathbb{R}^{K\times N}$ the K-dimensional data matrix where $N = N_S + N_T$. The covariance matrix $\mathbf{C}$ of $\mathbf{W}$ can be computed as $\frac{1}{N}\sum_{n=1}^{N}(W^{(n)} - \mu)(W^{(n)} - \mu)^T = \mathbf{WCW}^T$, where $\mathbf{C} = \mathbf{I} - \frac{1}{N}\mathbf{1}$ is a $N \times N$ centering matrix. The learning goal is to find a projection matrix $\mathbf{P} = [p_1, ..., p_{K'}]^T \in \mathbb{R}^{K\times K'}$ that maps each point to a low-dimensional space ($K' \leq K$). Here, we maximize the embedded data variance in orthogonal directions

$$\max_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} tr\big(\mathbf{P}^T\mathbf{WCW}^T\mathbf{P}\big) \tag{1}$$

where $tr(\cdot)$ denotes the trace of matrix.

2) Joint Distribution Adaptation: Even if the dimensionality-reduced representation can find a latent feature space, the distribution difference between source and target domains is still significantly large. To reduce the difference between joint distributions, we adopt the Maximum Mean Discrepancy (MMD) as the distance measure to compare different distributions. The distance between two distributions can be computed between the sample means of the two domains in the K'-dimensional embeddings:

$$Disc(\mathcal{W}_{newS}, \mathcal{W}_{newT}) = \left\| \frac{1}{N_S}\sum_{i=1}^{N_S}\mathbf{P}^T\mathcal{W}_S^{(i)} - \frac{1}{N_T}\sum_{j=1}^{N_T}\mathbf{P}^T\mathcal{W}_T^{(j)} \right\|^2 \tag{2}$$

where $\mathcal{W}_{newS}$, $\mathcal{W}_{newT}$ are the set of projected data points in the source and target windows.

However, reducing the difference in the marginal distributions does not guarantee a reduction in difference between conditional distributions of domains. Previous works

on multistream classification assume that at a specific time, the conditional probability would be similar between labeled source window and unlabeled target window, which is often not true in real world situations.

In this paper, we employ EMMD [21] as the distance measure to simultaneously match both marginal distributions and conditional distributions. Following [13], letting $l \in 1, ..., L$ denote the class label, the MMD to measure the joint distribution can be written as:

$$\begin{aligned} &\left\| \frac{1}{N_S^{(l)}} \sum_{\mathcal{W}_S^{(i)} \in S^{(l)}} \mathbf{P}^T \mathcal{W}_S^{(i)} - \frac{1}{N_T^{(l)}} \sum_{\mathcal{W}_S^{(j)} \in T^{(l)}} \mathbf{P}^T \mathcal{W}_T^{(j)} \right\|^2 \\ &= tr\big(\mathbf{P}^T \mathbf{W} \mathbf{M}_l \mathbf{W}^T \mathbf{P}\big) \end{aligned} \tag{3}$$

where $S^{(l)} = \{\mathcal{W}_S^{(i)} : \mathcal{W}_S^{(i)} \in S \wedge y(\mathcal{W}_S^{(i)}) = l\}$ are the instances in the source window whose class labels are l, and $T^{(l)} = \{\mathcal{W}_T^{(j)} : \mathcal{W}_T^{(j)} \in T \wedge y(\mathcal{W}_T^{(j)}) = l\}$, correspondingly, are the instances in the target window whose class labels are l. $N_S^{(l)}$, $N_T^{(l)}$ denote the number of instances belong to l in the source and target window respectively. The EMMD matrix $\mathbf{M}_l$ for $l \in 1, ..., L$ can be computed as following:

$$(M_l)_{ij} = \begin{cases} \frac{1}{N_S^{(l)^2}}, & \mathcal{W}^{(i)}, \mathcal{W}^{(j)} \in S^{(l)} \\ \frac{1}{N_T^{(l)^2}}, & \mathcal{W}^{(i)}, \mathcal{W}^{(j)} \in T^{(l)} \\ \frac{-1}{N_S^{(l)} N_T^{(l)}}, & \begin{cases} \mathcal{W}^{(i)} \in S^{(l)}, \mathcal{W}^{(j)} \in T^{(l)} \\ \mathcal{W}^{(i)} \in T^{(l)}, \mathcal{W}^{(j)} \in S^{(l)} \end{cases} \\ 0, & otherwise \end{cases} \tag{4}$$

When $l = 0$, EMMD measures the marginal probability between source and target window $\mathcal{W}_S$, $\mathcal{W}_T$. $\mathbf{M}_0$ is defined as following:

$$(M_0)_{ij} = \begin{cases} \frac{1}{N_S^2}, & \mathcal{W}^{(i)}, \mathcal{W}^{(j)} \in S^{(c)} \\ \frac{1}{N_T^2}, & \mathcal{W}^{(i)}, \mathcal{W}^{(j)} \in T^{(c)} \\ \frac{-1}{N_S N_T}, & otherwise \end{cases} \tag{5}$$

3) Iterative Learning: To achieve cross domain adaptation between the two streams, the difference in both marginal and conditional distributions should be minimized such that the covariance of data matrix is maximized in Eq. (1). Then the optimization version of this problem is given by:

$$\min_{\mathbf{P}^T \mathbf{W} \mathbf{C} \mathbf{W}^T \mathbf{P} = \mathbf{I}} \sum_{l=0}^{L} tr\big(\mathbf{P}^T \mathbf{W} \mathbf{M}_l \mathbf{W}^T \mathbf{P}\big) + \lambda \|\mathbf{P}\|^2 \tag{6}$$

According to the constrained optimization theory, we obtain the following Lagrangian of (6):

$$\begin{aligned} L(\mathbf{P}, \mathbf{\Phi}) =& tr\Big(\mathbf{P}^T \Big(\mathbf{W} \sum_{l=0}^{L} \mathbf{M}_c \mathbf{W}^T + \lambda \mathbf{I}\Big) \mathbf{P}\Big) \\ &+ tr\big(\big(\mathbf{I} - \mathbf{P}^T \mathbf{W} \mathbf{C} \mathbf{W}^T \mathbf{P}\big) \mathbf{\Phi}\big) \end{aligned} \tag{7}$$

where $\mathbf{\Phi} = \{\phi_1, ..., \phi_{K'}\} \in \mathbb{R}^{K' \times K'}$ is a diagonal matrix whose diagonal elements are the Lagrange multipliers.

Let $\frac{\partial L}{\partial \mathbf{P}} = 0$, solving the optimization function (Eq. (6)) is identical to solving an generalized eigenvalue problem:

$$\Big(\mathbf{W} \sum_{l=0}^{L} \mathbf{M}_l \mathbf{W}^T + \lambda \mathbf{I}\Big) \mathbf{P} = \mathbf{W} \mathbf{C} \mathbf{W}^T \mathbf{P} \mathbf{\Phi} \tag{8}$$

Finding the optimal projection matrix P is to perform an eigendecomposition for the K' smallest eigenvectors.

We should note that the optimization function requires the labels of target instances when computing the EMMD matrix M_l. However, the target domain only generates unlabeled test data in our multistream classification problem. Since there are no labeled data in target stream, the conditional probability cannot be modeled directly. Thus, we follow [13] to iteratively learn the projection matrix $\mathbf{P}$ and gain the predicted label of target stream to improve the classification accuracy.

The pseudo-code for joint distribution adaptation is described in Algorithm 2. It will be used for initialization and updating purpose when a drift is detected. We discuss the drift detection technique later in this section. The optimum solution of the projection matrix $\mathbf{P}$ can be obtained by first employing an initial label for instances in target window and improving the labeling quality iteratively.

Algorithm 2 LearnMat: Learn Projection Matrix

Input: Source instances $\mathcal{W}_S = \{\mathcal{W}_S^{(i)}\}_{i=1}^{N_m}$, source labels $\{y_S^{(i)}\}_{i=1}^{N_m}$ target instances $\mathcal{W}_T = \{\mathcal{W}_T^{(j)}\}_{j=1}^{N_m}$, subspaces dimension K', regularization term λ.
Output: Projection matrix $\mathbf{P}$,adapted classifier f.

```
Generate data matrix:
    W = [W_S^1, ..., W_S^{N_m}, W_T^1, ..., W_T^{N_m}]
Calculate MMD matrix M_0 using Eq. (5).
Initialize {M_l}_{l=1}^L to 0.
repeat
    Solve the generalized eigenvalue problem with
    corresponding K' smallest eigenvectors to construct
    the projection matrix P.
    Train a classifier on the adapted source instances
    {P^T W_S^(i)}_{i=1}^{N_m}.
    Predict the target instances label {y_T^(j)}_{j=1}^{N_m}
    Update the MMD matrix M_l by Eq. (4).
until Convergence
Return Projection matrix P and adapted classifier f.
```

B. Classification Module

Initially, we train a classifier using a small set of data instances from both S and T, which are referred to as the warm-up period data. Here, we learn a $K \times K'$ projection matrix $\mathbf{P}$ to overcome joint distribution difference between the two streams. The new data representation is obtained by the JDM using warm-up period data from $\mathcal{W}_S$ and $\mathcal{W}_T$ as follows:

$$\begin{cases} \mathcal{W}_{newS}^{(i)} = \mathbf{P}^T \mathcal{W}_S^{(i)}, & \mathcal{W}_S^{(i)} \in \mathcal{W}_S \\ \mathcal{W}_{newT}^{(j)} = \mathbf{P}^T \mathcal{W}_T^{(j)}, & \mathcal{W}_T^{(j)} \in \mathcal{W}_T \end{cases} \tag{9}$$

Any learning algorithm can be used in COMC. As new instances arrive in S or T, the classifier is updated whenever there is a drift to ensure that it represents the current concepts. A new base model is trained using data in $\mathcal{W}_S$ and $\mathcal{W}_T$ at that time. Drift detection and the updating method used by COMC are discussed in the section. COMC predicts the class label of an incoming test instance from the target stream after projecting the instance into the new data format.

C. Change Detection Module (CDM)

Previous works [7], [3] of multistream classification use prediction confidence to detect changes in distribution between two different time windows. Also, since the source or target stream may have asynchronous concept drifts, an ensemble of classifiers is maintained and updated if a concept drift is detected in either of these streams of data. As a result, the overall ensemble management is complex and makes the algorithms extremely slow.

Algorithm 3 ChangeDetection: Change Detection

Input: Source instances $\mathcal{W}_S = \{\mathcal{W}_S\}_{i=1}^{N_m}$, target instances $\mathcal{W}_T = \{\mathcal{W}_T\}_{j=1}^{N_m}$,new instance x,initial mean discrepancy $Disc_0$,the change parameter τ.
Output: $True$ if drift is detected, else $False$.

1: **if** $x \in S$ **then**
2: $\quad \mathcal{W}_S^{(N_m+1)} \leftarrow x$.
3: $\quad \mathcal{W}_{newS}^{(N_m+1)} = \mathbf{P}^T x$.
4: $\quad \mathcal{W}_S^{(i)} \leftarrow \mathcal{W}_S^{(i+1)}, i = 1, ..., N_m$.
5: $\quad \mathcal{W}_{newS}^{(i)} \leftarrow \mathcal{W}_{newS}^{(i+1)}, i = 1, ..., N_m$.
6: $\quad \mu_S = \frac{1}{N_m} \sum_{i=1}^{N_m} \mathbf{P}^T \mathcal{W}_{newS}^{(i)}$.
7: $\quad$ Go to line 14.
8: **end if**
9: $\mathcal{W}_T^{(N_m+1)} \leftarrow x$.
10: $\mathcal{W}_{newT}^{(N_m+1)} = \mathbf{P}^T x$.
11: $\mathcal{W}_T^{(i)} \leftarrow \mathcal{W}_T^{(j+1)}, j = 1, ..., N_m$.
12: $\mathcal{W}_{newT}^{(j)} \leftarrow \mathcal{W}_{newT}^{(j+1)}, j = 1, ..., N_m$.
13: $\mu_T = \frac{1}{N_m} \sum_{j=1}^{N_m} \mathbf{P}^T \mathcal{W}_{newT}^{(j)}$.
14: Calculate the mean discrepancy $Disc_t$ at time t:
15: $\quad Disc_t = \|\mu_S^t - \mu_T^t\|^2$.
16: $s = ln \frac{Disc_t}{Disc_0}$.
17: **Return** $s > -ln(\tau)$**.**

In this section, we invoke a simple but efficient change detection method by monitoring significant changes in $\mathcal{W}_S$ and $\mathcal{W}_T$. Since data continuously arrives in the source and target streams, the MMD model in Eq. (2) needs to be updated also by updating mean of adapted data points in the two windows respectively. The online updating process is given by:

$$\begin{cases} \mathcal{W}_{newS}^{(i)} = \mathbf{P}^T \mathcal{W}_S^{(i+1)}, & i = 1, ..., N_m \\ \mathcal{W}_{newT}^{(j)} = \mathbf{P}^T \mathcal{W}_T^{(j+1)}, & j = 1, ..., N_m \end{cases} \tag{10}$$

Let $Disc_0$ be the initial value of mean discrepancy, and $Disc_t$ be the value of mean discrepancy at time t. The $Disc_t$ is updated online as new instances arrive in S or T. The difference between the distributions is quantized by taking the likelihood ratio. A drift is detected if it is more than a user-defined threshold τ, as follows.

$$S = ln \frac{Disc_t(\mathcal{W}_{newS}^t, \mathcal{W}_{newT}^t)}{Disc_0(\mathcal{W}_{newS}^0, \mathcal{W}_{newT}^0)} > \tau \tag{11}$$

Algorithm 3 outlines the online updating of parameters for change detection. As shown in line 1-6, if a new instance arrives in the source stream, we update the window $\mathcal{W}_S$ by discarding the oldest instance and storing the new instance. Then we project the newly arrived data point into the new sub-feature space and update the mean of source instances μ_S in current window. Otherwise, we update the target window $\mathcal{W}_T$ and target mean $\mu_T$instead (line 9-13). Finally, the mean discrepancy of two streams is calculate by line 15, and a change score calculated by line 16.

The efficiency of COMC stems from the fact that in addition to estimating the projection matrix among current fixed windows, it uses the same Mean Discrepancy model for drift detection. Therefore, COMC detects drift without adding any extra overhead.

D. Complexity Analysis

In this approach, a new model is trained from source to target stream for every iteration of multistream classification. Thus, the time complexity of this approach depends on the classification model we used. Within the classification model, the training process, which is the updating model determines the whole algorithm.

COMC has three modules, JDM, CDM, and Classification Module. JDM is to learn a projection matrix $\mathbf{P}$ from the instances stored in the source and target sliding window (Algorithm 2). The time complexity for JDM is $O(IK'K^2)$ where I is the number of iterations, and K and K' are the number of features before and after adaptation. Time complexity of CDM (Algorithm 3) is $O(N_m)$. Time complexity of classification depends on the learning algorithm used as the base model. Therefore, COMC has total time complexity of $O(IK'K^2) + f(N_m)$, where $f(N_m)$ is the time complexity for training a new classification model.

TABLE II
CHARACTERISTICS OF DATA SETS

Data set	# features	# classes	# instances
USPS	256	10	10,000
MNIST	256	10	60,000
PIE	1024	68	370,000
AmazonReview	300	4	65,427
News	200	2	64,629
CruiseTravel	59	12	161,370

V. EVALUATION

In this section, we evaluate the proposed approach using synthetic and real-world data sets. Then we compare the performance of the proposed approach with a couple of baseline methods.

TABLE III
COMPARISON OF PERFORMANCE

Data Set	COMC		FUSION		MSC		SVM		TCA	
	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
USPS → MNIST	**42.37**	1.47	18.08	694.08	19.46	190.27	26.38	0.43	30.62	38.98
PIE@frontal→ left	**58.17**	8.40	31.39	2434.24	12.40	150.01	27.91	2.64	14.75	50.39
PIE@frontal → upward	**54.73**	8.53	42.23	2416.34	22.06	1434.11	40.96	2.66	32.12	53.85
PIE@frontal → downward	**52.71**	8.25	48.23	2430.19	20.47	1424.64	46.48	2.49	36.50	54.31
PIE@frontal → right	**44.28**	8.84	36.16	2438.96	16.12	143.29	32.11	2.23	16.25	48.59
AmazonReview@video → dvd	**46.68**	3.02	43.64	793.53	42.06	11.47	39.46	0.76	45.06	40.83
AmazonReview@video → music	**43.62**	3.38	37.06	807.80	38.66	11.58	36.25	0.83	41.00	39.11
AmazonReview@music → dvd	**46.82**	3.56	40.05	850.04	42.27	10.12	40.76	0.86	44.37	39.85
News@palestine → microsoft	**73.35**	1.16	67.03	633.15	71.34	343.66	65.61	0.32	72.54	37.64
News@palestine → economy	**78.75**	1.06	73.02	591.87	76.81	383.56	68.47	0.36	78.23	38.68
News@microsoft → economy	**78.26**	0.91	72.52	623.45	77.72	304.46	67.03	0.34	76.87	38.73
CruiseTravel@AS → EU	**22.29**	0.79	20.02	90.41	19.76	53.56	21.98	0.23	20.37	35.65
CruiseTravel@AU → EU	**25.84**	0.82	22.34	96.16	22.19	59.42	24.62	0.22	21.37	36.13
CruiseTravel@AS → US	**23.93**	0.74	20.65	87.26	21.34	51.92	22.25	0.26	20.62	39.99
CruiseTravel@AU → US	**25.41**	0.73	21.28	92.34	22.23	55.04	23.31	0.25	22.12	35.91
Average	**47.81**	3.44	39.58	1005.32	34.99	308.47	38.90	0.99	38.18	44.51

A. Data Sets

Table II lists the data sets used in the experiments. The first five data sets are publicly available real-world data sets. To evaluate our proposed cross-domain multistream framework, the last four data sets are divided into a couple of categories, each of which corresponds to a specific domain. Note the numbers of labeled instances in each source stream are much less than unlabeled instances in the corresponding target stream since in our problem setting, we assume labeled data is very scarce. Thus, the category containing the least instances in each data set is adopted as the source domain in our evaluation process and the rest are adopted as target domains.

USPS vs. MNIST contains gray-scale images of handwritten digits collected from different sources, they share 10 classes of digits [23]. The USPS dataset consists of 11,000 images of size 16×16, and MNIST dataset consists of 60,000 images of size 28×28. We uniformly resize all the images to 16×16 and construct the feature vectors using normalized pixel values.

AmazonReview@X is a multi-domain sentiment data set containing product reviews taken from Amazon.com from many product types [2], where X denotes the produce type. To adapt the review data for our streaming setting, we typically choose three product types–dvd, music, and video–which contain a relative large number of product reviews. In this paper, instead of implementing binary classification, we directly use the star ratings as 4 classes in the Review data set. To extract features from raw review text, we use the word2vec model that was pre-trained on 100 billion words from Google [16].

News@X is a news popularity data set obtained from the UCI repository as explained in [18]. This is a large data set of news items and their respective social feedback on multiple platforms: Facebook, Google and LinkedIn. The collected data accounts for about 60,000 news items on three different topics: economy, microsoft, and palestine. This data set is tailored for news popularity prediction in binary classification task, news with average popularity > 10 are labeled as popular news, those with average popularity < 10 are labeled as unpopular news. Each news in the original data set only contains it's title and headline. To extract features from those short texts, we directly convert the text document to a 200 length vector of token counts instead of using Sentence Embedding like mentioned above.

CruiseTravel@X are the real world travel data set populated by many companies like TripAdvisor. To predict the destination of the trip, we use traveler information such as budget, cabin location, age and so forth. Moreover, we utilize the historical records of customers like the number of reserved cruises. In order to adapt our algorithm, we divide the data set into four continents, which corresponds to the four domains based on travelers' permanent address. The above X represents different continents: Asian (AS), Australia (AU), Europe (EU) and United State (US) where total 161,370 records are included.

Following the strategy in [10], we simulate concept drifts in those real-world data sets using a rotating hyperplane. We first normalize all features within the range of [-1, 1]. Then, at a time instance, which is selected in the middle of every stream, we generate a weight vector $\mathbf{v} = \{v_k\}_{k=1}^{K}$ between -2 and -1 at random. The drifted data points are generated based on the equation $x_k = v_k x_k$, where x_k is the component of a data point.

B. Baseline Methods

The first and second baselines we use in this paper is MSC [3] and FUSION [9]. These methods trains classifiers using weighted instances from the source stream to address data shift adaptation. The weights are calculated by using kernel mean matching for MSC and by estimating density ratios for FUSION. To select the parameters, we use $S_{MSC} = 5000$, $E_{MSC} = 10$, $\alpha_{MSC} = 0.001$ and $L_{fusion} = 2$, $\tau_{fusion} = 0.0001$, $\lambda_{fusion} = 0.01$, $\eta_{fusion} = 1$, as suggested in [3] and [9] respectively. In the experiments, we note that FUSION detects drifts at every data point on most of

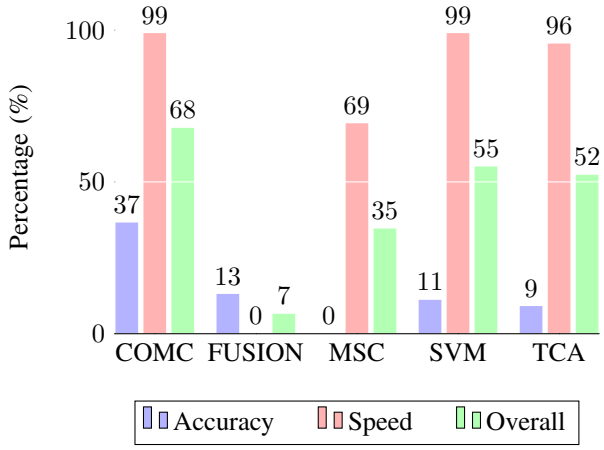


Fig. 4. Comparison of classification gain

the data sets. Thus we remove its drift detection function and periodically update the classifier, which gives an advantage to FUSION.

MSC and FUSION both use a Support Vector Machine (SVM) as the base classifier. For fair comparison, we design a second baseline employing a simple SVM model. The classification model is periodically updated on the latest data instances arrived in S and then used to predict the target labels of data instances in T.

The last baseline we choose to compare with is a well-known domain adaptation method, TCA [21]. Since it cannot be applied to our multistream framework, we divide the two independent streams into several disjoint chunks equal to the window size of COMC and implement the classification in batch manner. The sub-feature dimension $m = 50$ is selected the same as COMC. Evaluation is then performed repeatedly on the different chunks and the average accuracy and execution time is reported in Figure III.

C. Setup

We implemented the proposed approach and the baseline methods using Python version 2.7.6. All the methods have been evaluated using a Linux machine with 2.40 GHz core and 16 GB of main memory. The approach also involves multiple parameters. We use $N_m = 800$ as our default setting in the experiments. Also, $K' = 50$, $\lambda = 0.1$ and $I = 10$.

D. Classification Performance

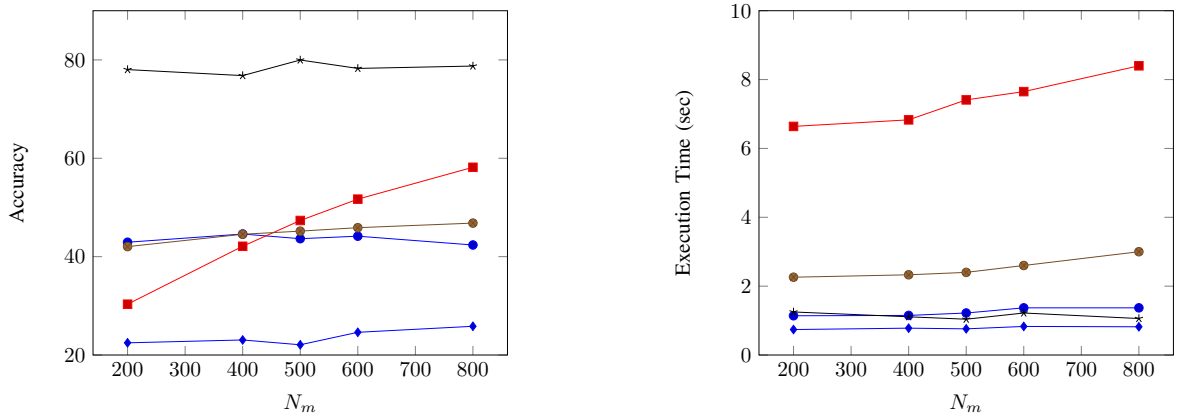
The first set of experiments are designed for comparing classification accuracy and execution time of the approaches considered in this paper on all the data sets mentioned in Table 1.

1) Classification Accuracy: Classification accuracy on different data sets have been shown in Table III. The proposed approach clearly outperforms all the other baseline approaches considered in this paper. As stated before, we apply MSC, FUSION and SVM on our multistream setting and periodically update the classification model for examining if simply retraining the classifier without unifying the feature space is useful. Also, to compare the performance between our method and the most popular domain adaptation method, we implement TCA in batch manner on the disjoint data stream chunks. For a fair comparison, we consider that true labels of only the source stream instances are available.

It can be observed that SVM performs poorly compared to the other four baseline methods on most of the data sets. Since one of the assumptions of SVM is that the distribution of training and testing data are similar, the SVM model suffers in the multistream setting due to not handling data distribution shift. The proposed approach also outperforms FUSION and MSC by a large margin. For instance, the average accuracy in the four PIE@X data sets from COMC is about 52.33%, 39.50% from FUSION and 21.27% from MSC. It can be seen that the accuracy has improved around 32% using COMC. This indicates that only applying data distribution shift correction techniques among the two cross-domain data streams is not sufficient. The proposed COMC can effectively learn common components underlying two different domains while matching the data distribution difference. Also, COMC significantly outperforms the state-of-art domain adaptation method TCA which only addresses the reduction of marginal distribution difference. Instead, COMC constructs new feature representation by jointly adapting both the marginal distribution and conditional distribution, achieving much better results.

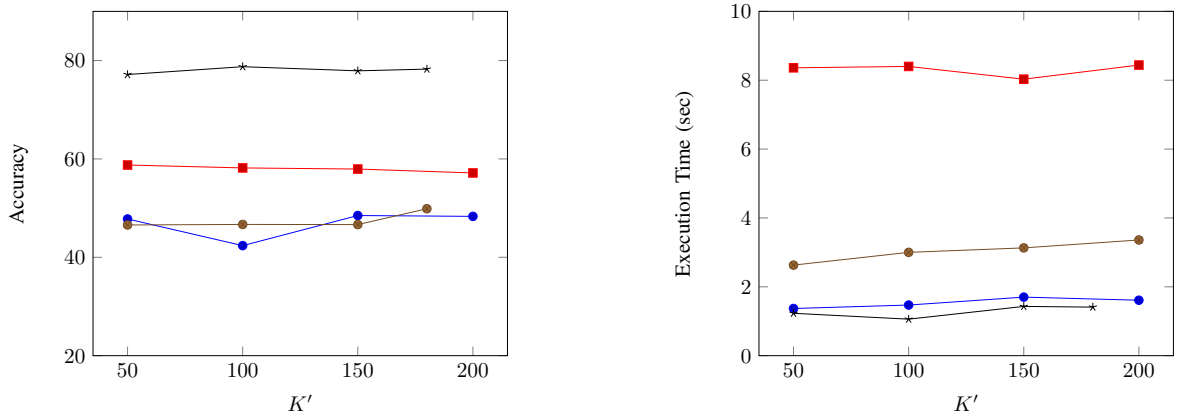
2) Execution Time: Table III shows the average time to process 1000 instances (in seconds) by the approaches on different data sets. As discussed in Section 4.4, the time complexity of COMC is $O(IK'K^2) + f(N_m)$, therefore the data dimension K is the most significant factor for the performance of our approach. We observe that those data sets with higher dimension like PIE have longer execution time compared to other data sets in the experiment. However, COMC still achieves much better execution time compared to FUSION, MSC and TCA. As shown in the four PIE@X data sets, it takes about 2430, 1434 and 53 seconds for FUSION, MSC and TCA respectively running on the high-dimensional streaming data, whereas COMC only takes 8.5 seconds in average. This is expected since FUSION uses a kernel model to estimate and update the importance weights or density ratios of source instances at every data point, which is extreme time-consuming when performing on the high-dimensional data stream. In contrast, COMC efficiently learns a projection matrix on the data points stored in $\mathcal{W}_S$, $\mathcal{W}_T$ and maps them into a new sub-feature space by simply applying a linear transformation. Moreover, since TCA can only be implemented in batch manner on the streaming data, the classifier is forced to update at each chunk, which makes the time complexity extremely high when processing continuously arriving instances, whereas COMC directly utilizes a pre-trained classifier and projection matrix to adapt the incoming data and update the training model only if a change is detected. Also, the time complexity of COMC is very close to the other baseline SVM especially on low-dimensional data sets.

3) Classification Gain: We further show the superiority of COMC by reporting classification accuracy and speed



(a) Accuracy (b) Execution Time

Fig. 5. Sensitivity of COMC to difference window size (N_m): *USPS→MNIST*; *PIE@frontal→left*; *AmazonReview@video→dvd*; *News@palestine→economy*; *CruiseTravel@AU→EU*;



(a) Accuracy (b) Execution Time

Fig. 6. Sensitivity of COMC to difference # sub-feaures (K'): *USPS→MNIST*; *PIE@frontal→left*; *AmazonReview@video→dvd* *News@palestine→economy*;

gain in Figure 4. The accuracy gain of method m is calculated as $AG_m = (a_m - a_{min})/a_{min}$. a_m represents the average classification accuracy of approach m over all data sets and a_{min} the lowest average accuracy among all competing approaches. Similarly, the speed gain of method m is calculated as $SG_m = t_{max} - t_m/t_{max}$ and the overall gain as $(AG + SG)/2$. The overall gain of COMC, TCA, FUSION, MSC and SVM is 67.81%,6.55%,34.65%,55.08% and 52.34% respectively, which indicates that our method significantly improves classification performance compared to existing baseline methods. Although SVM has a similar speed gain with COMC, but the accuracy gain is very low compared to COMC. Also, we can notice that FUSION has the lowest overall gain due to it's very high computation complexity.

E. Parameter Sensitivity

The next set of experiments are designed to examine parameter sensitivity of COMC on both real-world and synthetic data sets. For simplicity, we select four data sets from different catergories respectively, USPS→ MNIST, PIE@frontal→ left, AmazonReview@video→ dvd, News@palestine→ economy and CruiseTravel@AU→ EU. In these experiments, we have used $N_m = 800$, $K' = 50$, $I = 10$ and $\lambda = 0.1$ as the default setting if not mentioned otherwise.

1) Window Size: We study the sensitivity of COMC to window size or N_m. Fig. 5 shows the accuracy and execution time when using different values of N_m. We observe that the accuracy of most data sets remains similar with little fluctuations as the size of the sliding window increases. However, the accuracy of PIE decreases from 58.17% to 30.33% as the size of window size decreases from 800 to 200. It is understandable since PIE contains 68 individual classes, a smaller window will only contain very few data instances of each class, which can negatively affects the prediction accuracy on the target stream. Moreover, it can be observed that the execution time slightly goes up as the sliding window size increases. Just like we discussed in Section 4.4, N_m is one of those factors deciding the time complexity. Therefore, choosing a larger window size for the data set that contains a great number of distinct classes

is vital for getting better prediction accuracy.

2) No. of Sub-feaure: The performance of COMC with different number of sub-feature K' is shown in Fig. 6. Here, the value of K' varies between 50 and 200. The result of CruiseTravel is not included since the data set only contains 59 features. It can be seen that under most scenarios, changing the number of sub-features does not significantly affect the accuracy. It indicates that a low dimensional subspace is already a sufficient representation even for a 1024-dimensional data set. However, the change of execution time over increasing K' has a similar trend to N_m as we discussed above since K' also partly affect the time complexity. Thus, it is appropriate to chose a smaller K' considering the cost of learning projection matrix multiple times on the streaming data.

VI. Conclusion

In this paper, we perform classification in a cross-domain multistream setting involving two independent yet related data streams generated from different domains. We address the main challenges of domain adaptation by using EMMD and utilizing available ground truth labels on the source stream to iteratively refine the prediction accuracy on the target stream and learn the adapted classifier. We then present an online mechanism to update the average mean discrepancy of adapted data arrived in the two sliding windows, and utilize it for change detection. Our empirical evaluation on real-world data sets demonstrate that our method performs significantly better than the baseline approaches.

References

[1] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[2] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*, 2007.

[3] S. Chandra, A. Haque, L. Khan, and C. C. Aggarwal. An adaptive framework for multistream classification. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 1181–1190, 2016.

[4] O. Day and T. M. Khoshgoftaar. A survey on heterogeneous transfer learning. *J. Big Data*, 4:29, 2017.

[5] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings*, pages 286–295, 2004.

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

[7] A. Haque, L. Khan, M. Baron, B. M. Thuraisingham, and C. C. Aggarwal. Efficient handling of concept drift and concept evolution over stream data. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 481–492, 2016.

[8] A. Haque, H. Tao, S. Chandra, J. Liu, and L. Khan. A framework for multistream regression with direct density ratio estimation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

[9] A. Haque, Z. Wang, S. Chandra, B. Dong, L. Khan, and K. W. Hamlen. FUSION: an online method for multistream classification. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 919–928, 2017.

[10] S. Ho and H. Wechsler. On the detection of concept changes in time-varying data stream by testing exchangeability. *CoRR*, abs/1207.1379, 2012.

[11] H. D. III. Frustratingly easy domain adaptation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*, 2007.

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

[13] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer feature learning with joint distribution adaptation. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 2200–2207, 2013.

[14] M. M. Masud, T. Al-Khateeb, L. Khan, C. C. Aggarwal, J. Gao, J. Han, and B. M. Thuraisingham. Detecting recurring and novel classes in concept-drifting data streams. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pages 1176–1181, 2011.

[15] M. M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham. A practical approach to classify evolving data streams: Training with limited amount of labeled data. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 929–934, 2008.

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

[17] L. L. Minku. Transfer learning in non-stationary environments. In *Learning from Data Streams in Evolving Environments*, pages 13–37. Springer, 2019.

[18] N. Moniz and L. Torgo. Multi-source social feedback of online news feeds. *CoRR*, abs/1801.07055, 2018.

[19] S. A. Noghabi, K. Paramasivam, Y. Pan, N. Ramesh, J. Bringhurst, I. Gupta, and R. H. Campbell. Samza: stateful scalable stream processing at linkedin. *Proceedings of the VLDB Endowment*, 10(12):1634–1645, 2017.

[20] S. J. Pan, J. T. Kwok, and Q. Yang. Transfer learning via dimensionality reduction. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 677–682, 2008.

[21] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Trans. Neural Networks*, 22(2):199–210, 2011.

[22] O. Papapetrou, M. Garofalakis, and A. Deligiannakis. Sketching distributed sliding-window data streams. *The VLDB JournalThe International Journal on Very Large Data Bases*, 24(3):345–368, 2015.

[23] S. Roweis. Handwritten digits. Accessed: 2018-05-22.

[24] M. Sugiyama, M. Krauledat, and K. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005, 2007.

[25] V. Vapnik. *Statistical learning theory*. Wiley, 1998.

[26] X. Zhou and L. Chen. Event detection over twitter social media streams. *The VLDB JournalThe International Journal on Very Large Data Bases*, 23(3):381–400, 2014.