Multistream Classification with Heterogeneous Feature Space

Yifan Li, Yang Gao, Hemeng Tao, Latifur Khan and Patrick Brandt
The University of Texas at Dallas
Richardson, TX
Email:(yli, yxg122530, hxt160430, lkhan, pbrandt)@utdallas.edu

Abstract

Under a newly introduced setting of multistream classification, two data streams are involved, which are referred to as source and target streams. The source stream continuously generates data instances from a certain domain with labels, while the target stream does the same task without labels from another domain. Existing approaches assume that domains for both data streams are identical, which is not quite true in reality, since data streams from different sources may contain distinct features. Indeed, they may even have different numbers of features. Furthermore, obtaining labels for every instance in a data stream is often expensive and time-consuming. Therefore, it has become an important topic to explore whether labeled instances from other related streams can be helpful to predict those unlabeled instances in a certain stream. Note that domains of source and target streams may have distinct feature spaces and data distributions. Our objective is to predict class labels of data instances in the target stream by using the classifiers trained by the source stream.

We propose a framework of multistream classification by using projected data from a common latent feature space, which is embedded from both source and target domains. Empirical valuation and analysis on both real-world and synthetic datasets are performed to validate the effectiveness of our proposed algorithm, comparing to state-of-the-art techniques. Experimental results show that our approach significantly outperforms other existing benchmarks.

Introduction

Data streams are crucial in the modern connected Internet world, and they have attracted the attention of researchers worldwide. Given important applications of data streams– such as IoT, social networks, and surveillance–mining them properly is becoming a more and more important topic to explore. However, data stream mining is also a challenging task due to its distinctive nature. For example, a data stream is theoretically infinite in length, therefore, its volume is very large. Meanwhile, it is possible that with high velocity of data arrivals, class labels are not available immediately after the arrival of new data instances, which makes it difficult to update the model. These properties have distinguished data stream mining significantly from traditional data mining problems (Hou, Zhang, and Zhou 2017; Haque, Khan, and Baron 2016).

Normally, one assumption of classification model is that source data with true labels are also representative to target data. However, since data streams are generated by non-stationary processes, it is possible that arbitrary changes may be introduced to data streams. Thus, using a model trained by old source data may introduce bias in prediction (Herlihy 1993), and will lead to poor performance in predicting future data labels. Thus, the model needs proper updates to adapt the current concept.

When it comes to the setting of multiple streams, this problem becomes even more challenging. The problem setting is described and motivated by a multistream classification (MSC) algorithm (Chandra et al. 2016). Two data streams are involved simultaneously under this setting. One is referred as the source stream, which contains data instances with class labels generated from one non-stationary process with labels. The other is referred as the target stream, which consists of data generated from another non-stationary process without labels.

In this paper, a domain adaptation setting is considered within the multistream classification framework. There is a need to address machine learning tasks of building models in a one domain by using information available from another related domain. Here, knowledge from the source domain (stream) can be transferred to the target domain (stream) to help get high prediction accuracy. This transfer process is important in many circumstances where labels in a certain domain are limited or too expensive to obtain (Arnold, Nallapati, and Cohen 2007), which fits the need of multistream by nature. Notice that here the feature representations (Number. of features) and distributions may not identical across domains.

In practice, it is assumed that training data are provided as a whole in advance (Pan and Yang 2010). However, under a data stream setting, all instances may arrive in a sequential manner. Take online spam email detection for example (Chen and Chen 2015). Normally, a classifier is built using a static training dataset, and it is trained in batches to detect spam emails as accurately as possible. However, the very definition of spam varies from person to person. In this case, the transfer of knowledge to personalize the spam detector for each individual becomes critical. Even for a spe-

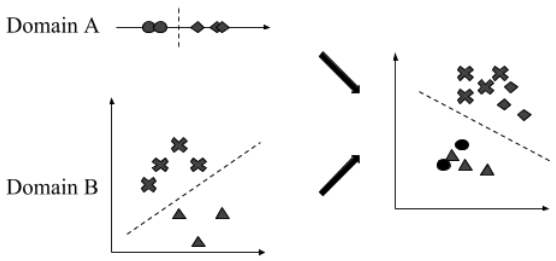

Figure 1: Feature Space Projection

cific person, the definition of spam could change over time. During a certain time period one may consider an advertisement from Zillow to be spam. When he starts to consider buying a house, Zillow ads are not spam for him anymore. Such problem can be formulated as a multistream learning task, the objective of which is to build an online prediction model for the target domain by using information available in the source domain with distinct feature space. Compared to the previous example, this task is more challenging, as the concept is evolving in the data streams simultaneously in heterogeneous source and target streams.

To sum up, the problem that we are trying to solve involves two data streams with heterogeneous feature spaces, and this is the aspect which no researchers have been working on in the past.

This paper proposes a framework, called MultiStream Domain Adaptation (MSDA), to handle the issues described above. The main idea is to find a common feature space for two distinct data streams. This idea is demonstrated in Figure 1. Data in domain A are one-dimensional and in domain B are two-dimensional. By applying the proposed projection algorithm, we try to find a latent feature space that the distributions in original and latent feature space are similar. Meanwhile, the structure of data should be preserved, which means distinct classes are still far apart, and the decision boundaries for different categories should still be preserved. Thus the core problem here is to find a feature space that

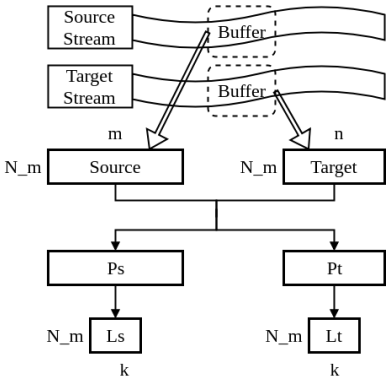

Figure 2: Feature Projection

maximize the similarity between original and projected latent feature space. Main contributions of this paper are as follows:

1. An embedding-based domain adaptation method is introduced to address the challenges of adapting source and target domains in a non-stationary environment.
2. A concept drift detection method is deployed under the new multistream setting with domain adaptation. Under this setting, true labels in the target stream is not required, while only true labels in the source stream is used.
3. Our approach is empirically evaluated over several benchmark datasets, and the results are compared with state-of-the-art methods. The performance of our proposed method is significantly better than the other three methods.

Related Work

We outline related work under two topics, namely domain adaptation and stream classification.

Domain Adaptation

One of fundamental assumptions in data mining, known as the "stationary distribution assumption", is that both the training and test data are generated from the same domain and thus represent the same data distribution (Zadrozny 2004). Therefore, common techniques normally cannot be directly deployed when training and test data are from different domains. The differences between domains can be normally considered as two aspects: 1. distinct number of features; 2. distinct feature distributions. Several approaches have been proposed to learn a common feature representation for domain adaptation (Ben-David et al. 2010; Pan and Yang 2010). (Shi et al. 2010) proposed an linear objective function to project a latent feature space for both source and target domains. This approach focuses on static data only. Recall that in stream feature space may evolve over times across streams, which may trigger heterogeneous feature space. (Zhao and Hoi 2010) uses a co-regularized method to project target domain to source domain in an online manner. However, this method also makes a strong assumption that features in source dataset is a subset of those in target dataset, which in reality, this is not always the case. However, the method works in an incremental manner which may be suitable for stream setting.

Similarly, as we stated in the previous section, it is challenging to derive an online method to reduce the distribution different between domains. The state-of-the-art methods, including deep learning, mostly address domain adaptation problems for stationary data except (Zhao and Hoi 2010). Thus, fitting domain adaptation frameworks to online data streams is yet to explore.

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 or using

a gradual forgetting mechanism. Recent approach (Bifet and Gavaldà 2007) handles 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, $P(y|x)$. Instead of tracking changes in $P(y|x)$ directly, the approaches proposed in (Haque et al. 2018) adopt the principle (Gama et al. 2004) 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 confidence (Chandra et al. 2016) 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 (Chandra et al. 2016). It proposes a solution which focuses on the classification problems when there are asynchronous concept drifts on source and target streams. Since stream classification is a continuous process, data in the target stream is assumed to be generated with very few labels. Yet, there is a strong assumptions made for this work: thenumber of features in both source and target streams is fixed. To overcome those obstacles, our proposed approach relaxs this assumption. features may not be exactly the same between domains, both in number and correspondence. Furthermore, other features in source stream may not have any mapping to features in target stream and vice-versa.

Problem Formulation

Notations and Problem Statement

In Table 1, frequently used symbols in this paper are listed. There are two continuous stream of data instances generated from source domain D_s and target domain D_t. A data instance is denoted as (x, y), where x is a vector (m-dimensional in source stream and n-dimensional in target stream), and y is the corresponding class label. In the source stream, both x_s and y_s are available, while in the target domain only x_t is available. Therefore, a multistream classification with domain adaptation problem can be described as follows:

Suppose X_s is a set of m-dimensional vectors and Y_s is the corresponding labels in a source stream from a certain domain D_s, whereas X_t is a set of n-dimensional vectors in target stream from another domain D_t. In our problem setting, $m \neq n$. The objective is to construct a classifier M that predicts class labels of $x_t \in X_t$ using X_s, and Y_s.

Challenges

In the problem setting of multistream classification with domain adaptation, there are two major challenges at the same time. The first challenge is the adaptation of both source and target domain, and it can be represented as $D_s \neq D_t$. As shown in Figure 2, two streams may have different number of instances. However, without loss of generality, buffers (windows) from these streams representing contiguous data points will have the same number of instances N_m. Furthermore, each data point in source stream may have a different number of features compared to those in the target stream. Therefore, source data cannot be directly used as training data to learn the target task, and discovering a latent feature space with k dimensions is the key to handling the feature heterogeneity issue. The second challenge is the asynchronous concept drift in both source and target streams. This phenomenon means that the data pattern evolves, or more formally, the conditional probability distribution changes over time in both streams. Here, the problem can be described as $P_s^t(y \mid \mathbf{x}) \neq P_t^t(y \mid \mathbf{x})$ at time t. Furthermore, we assume that source and target data streams have asynchronous concept drifts, which means that the drifts in both streams occur independently.

Table 1: Notations

Symbol	Meaning
D_s	Domain of source stream
D_t	Domain of target stream
S	Source stream data
T	Target stream data
X_s, X_t	Feature space of source stream with dimension m and that of target stream with dimension n
Y_s, Y_t	Label space of source and target stream
x_s, x_t	Data instance of source and target stream
y_s, y_t	True and predicted label of a data instance in source stream
W_s, W_t	Projection function to the source and target space
L_s, L_t	Projected data from source and target domain to latent space
P_s, P_t	Probability distribution function of source and target data
N_m	Size of sliding window

Proposed Approach

In this paper, we propose a Multistream Domain Adaptation (MSDA) framework to address the major issues in our problem setting. To achieve this goal, we establish a framework with the following modules:

1. A domain adaptation module that helps find an optimized latent subspace for both source and target streams.
2. A concept drift detection module that detects concept drift in both source and target streams.

Applying these two modules together, once a concept drift is detected, we use data instances from both streams in the most recent window to update the feature mapping, so that the domain adaptation problem can be addressed. The diagram of this algorithm can be found in Figure 1.

First, a domain adaptation module is triggered to learn projection functions for both source and target data instances (step 1, 2 & line 2-6). Newly arriving instances are transformed to latent feature representation accordingly (line 8-11). Second, the change point detection module detects if there is a significant change in either source or target streams within the sliding windows Third, once a change point is detected, new classifiers are trained based on these two buffers from B_s and B_t (step 3 & line 12-16). Finally, the newly updated classifiers are used to predict the labels of adapted data

Algorithm 1 MSDA Algorithm

Require: Labeled source stream S, Unlabeled target stream T, The size of sliding window N_m. Similarity parameter β.
Ensure: Labels predicted on T.

```
1: /* Initialization */
2: B_s, B_t ← readData(S, T)
3: /* DA for initial buffer */
4: W_s, W_t ← genProjectFunction(B_s, B_t, β)
5: L_s, L_t ← genProjectMatrix(B_s, B_t, W_s, W_t)
6: M ← buildModel(L_s, Y_s)
7: while S or T exists do
8:     B_s, B_t ← readData(S, T)
9:     /* Concept drift detection and correction */
10:    Call ChageDetection /* Algorithm 2 */
11:    if ChangeDetection = True then
12:        /* Update prediction model */
13:        /* DA for stream buffer */
14:        W_s, W_t ← genProjectFunction(B_s, B_t, β)
15:        L_s, L_t ← genProjectMatrix(B_s, B_t, W_s, W_t)
16:        M ← buildModel(L_s, Y_s)
17:    /* Generate predictions */
18:    ŷ_t ← getPrediction(M, L_t)
```

instances from target stream T (step 4, 5, 6 & line 17-18). Details of our proposed method is as follows.

Initialization and Domain Adaptation

In our proposed framework, instances from source and target streams are stored in B_s and B_t respectively. Recalling our problem setting, data in B_s have true labels, while data in B_t come without true labels. The domain adaptation method is indeed a optimization problem solved by feature embedding (Shi et al. 2010). Also, for computational purposes, we design our approach in a way that buffer size in both source and target streams are the same, which means the numbers of data instances in processing window for source and target streams are identical. Thus, given a certain time t, data that we have are: source stream window matrix $B_s \in \mathbb{R}^{N_m \times m}$; source stream window labels vector $Y_s \in \mathbb{R}^{N_m \times 1}$, and; target stream window matrix $B_t \in \mathbb{R}^{N_m \times n}$. In this case, the best projection strategy of L_s and L_t in the latent feature space would be the minimization of following objective function:

$$\mathcal{O} = \min_{L_s, L_t} \ell(B_s, L_s) + \ell(B_t, L_t) + \beta \mathbf{D}(L_s, L_t) \quad (1)$$

where $\ell(\cdot,\cdot)$ is a distance function that evaluates the differences between the original data and projected data. $\mathbf{D}(\cdot,\cdot)$ is the co-regularizer that promotes the similarities between the two projected domains (L_s and L_t). β is a parameter that determines how desirable the two projected data are similar. The first two terms are expected to preserve the structure of the original data as much as possible. Therefore, we define the loss function $\ell(\cdot,\cdot)$ as the Frobenius norm, which can also be expressed as a matrix trace norm $\ell(B_s, L_s)$ equals $\|B_s - L_s W_s\|_F^2$, and $\ell(B_t, L_t)$ equals $\|B_t - L_t W_t\|_F^2$.

We factorize the original data into projections (L_s and L_t) by linear mapping functions (W_s and W_t). Also, note that here we are not applying the alternative definition such as $\ell(B_s, L_s)$ as $\|B_s W_s - L_s\|_F^2$, since this definition will always lead to a trivial solution $W_s = 0$ and $L_s = 0$, thus $B_s W_s = L_s = 0$ will always minimize the objective function. From Equation 1 the projected data should preserve the structures of original data.

Furthermore, we define $\mathbf{D}(Ls, L_t)$ as follows:

$$\mathbf{D}(L_s, L_t) = \frac{1}{2}\ell(B_s, L_t) + \frac{1}{2}\ell(B_t, L_s) \quad (2)$$

which is the mean value of cross-similarity between the original data and projected data. Finally, the parameter β controls the trade-off of importance of semantic similarity co-regularization by minimizing the differences between $\mathbf{D}(L_s, L_t)$.

Combining Equation 1 and 2 together, we obtain the overall optimization objective function as follows:

$$\begin{aligned}\mathcal{O} = &\|B_s - L_s W_s\|_F^2 + \|B_t - L_t W_t\|_F^2 \\ &+ \frac{1}{2}\beta\|B_s - L_t W_s\|_F^2 + \frac{1}{2}\beta\|B_t - L_s W_t\|_F^2\end{aligned} \quad (3)$$

Notice here the projection itself will perform rotation and scaling on the target matrix to minimize the difference. Since L_s and L_t are orthogonal matrices, $B_s^\top B_s = 1$. Also, we are applying the cyclic permutation property of trace. Thus, Equation 3 can be expanded by the representation of trace.

According to (Long, Yu, and Zhang 2008), setting partial derivatives to zero would generate the optimal solution based on KTT conditions. Consequently, the projection function for both source and target stream can be formulated as:

$$\begin{aligned}W_s &= \frac{\beta}{2+\beta} L_t^\top B_s + \frac{2}{2+\beta} L_s^\top B_s \\ W_t &= \frac{\beta}{2+\beta} L_s^\top B_t + \frac{2}{2+\beta} L_t^\top B_t\end{aligned} \quad (4)$$

After the steps described above, the optimal projection for both source and target stream to the latent feature space would be formed accordingly.

Classification Module

We start to train a classifier with a small set of data instances from both S and T, which are referred to as the initialization data. The new data representation is obtained by using initialization data from B_s and B_t as follows:

$$\begin{cases} L_s^{(i)} = B_s^{(i)} W_s^{-1} & B_s^{(i)} \in B_s \\ L_t^{(j)} = B_t^{(j)} W_t^{-1} & B_t^{(j)} \in B_t \end{cases} \quad (5)$$

As new instances arrive in S or T, the classifier is updated if there is a drift to ensure that it represents the current concepts. A new classification model is trained using data in B_s and B_t at that time. Concept drift detection and the updating method used by MSDA will be discussed in the next subsection. MSDA predicts the class label of an incoming target instance from the target stream after projecting the instance into the new data format.

Algorithm 2 ChangeDetection: Change Detection

Require: Source instances $B_s = \{B_s\}^{i=1}$, target instances $B_t = \{B_t\}^{j=1}$, new instance x, initial mean discrepancy $Disc$, the change parameter τ.
Ensure: $True$ if drift is detected, else $False$.
1: /* Instance is from source stream */
2: **if** $x \in S$ **then**
3: $B_s^{(i)}, L_s^{(i)} \leftarrow B_s^{(i+1)}, L_s^{(i+1)}, i = 1, ..., N_m$.
4: $B_s^{(N_m+1)}, L_s^{(N_m+1)} \leftarrow x, xW_s^{-1}$.
5: $\mu_S = \frac{1}{N_m} \sum_{i=1}^{N_m} L_s^{(i)} W_s^{-1}$.
6: Go to line 16.
7: /* Instance is from target stream */
8: $B_t^{(N_m+1)}, L_t^{(N_m+1)} \leftarrow x, xW_t^{-1}$.
9: $B_t^{(j)}, L_t^{(j)} \leftarrow B_t^{(j+1)}, L_t^{(j+1)}, j = 1, ..., N_m$.
10: $\mu_T = \frac{1}{N_m} \sum_{j=1}^{N_m} \mathbf{L}_t^{(j)} W_t^{-1}$.
11: /* Calculate the mean discrepancy $Disc_t$ at time t: */ $Disc_t = \|\mu_s^t - \mu_t^t\|^2$.
12: $s = ln \frac{Disc_t}{Disc_0}$.
13: **Return** $s > -ln(\tau)$**.**

Change Detection Module (CDM)

Previous work (Chandra et al. 2016) of multistream classification uses prediction confidence to detect changes in distribution between two different time windows. Due to possible asynchronous concept drifts between source and target streams, an ensemble of classifiers is maintained and updated if a concept drift is detected in either of these streams of data. Therefore, complex ensemble algorithms may lead to very slow execution.

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:

$$Disc(L_T, L_{T'}) = \left\| \frac{1}{N_m} \sum_{i=1}^{N_m} L_s^{(i)} W_s^{-1} - \frac{1}{N_m} \sum_{j=1}^{N_m} L_t^{(j)} W_t^{-1} \right\|^2 \quad (6)$$

where L_T, $L_{T'}$ are the set of projected data points in the source and target windows.

We invoke a simple but efficient change detection method by monitoring significant changes in B_s and B_t. Since data continuously arrives in the source and target streams, the MMD model in Equation needs to be updated also by updating the mean of adapted data points in the two windows respectively. The online updating process is given by:

$$\begin{cases} L_s^{(i)} = B_s^{(i+1)} W_s^{-1} & B_s^{(i)} \in B_s \\ L_t^{(j)} = B_t^{(j+1)} W_t^{-1} & B_t^{(j)} \in B_t \end{cases} \quad (7)$$

Here, we let $Disc_0$ to initialize the mean discrepancy. $Disc_t$ at time t is updated online as new instances arrive in S or T. The difference between the distributions is determined by the log ration between MMD. Therefore, a concept drift point is detected if it is more than a user-defined threshold τ, as follows.

$$S = ln \frac{Disc_t(L_s^t, L_t^t)}{Disc_0(L_s^0, L_t^0)} > \tau \quad (8)$$

Algorithm 2 demonstrates an online updating algorithm for change point detection. If a new instance arrived in the source stream, we update the window L_s by eliminating the oldest instance and storing the newest instance (line 1-6). Then, new arrived data are projected into the new subspace, and the mean of source instances μ_s is updated as well in current window. Otherwise, we update the target window L_t and target mean μ_t instead (line 9-13). At last, the mean discrepancy of two streams and a change sore are calculated (line 15, 16).

Table 2: Datasets

Data set	# features	# instances
USPS	256	10,000
MNIST	780	60,000
VIDEO	100	5,000
DVD	200	30,000
MUSIC	300	30,000
SYN01	100	10,000
SYN02	200	20,000

Complexity Analysis

In our proposed approach, an update model is trained from source to target stream anytime when a concept drift point is detected. Hence, the time complexity of classification module is significant in determining the overall model complexity. Within the classification model, the training process, which is the updating model determines the complexity of whole algorithm.

MSDA has three modules, Domain Adaptation (DA) module, Change Point Detection (CDM) module, and classification module. The DA module learns projection matrix W_s W_t respectively, and from the instances stored in the source and target sliding window. Since it is a sequence of matrix transformation, the processing time is $O(kN_m)$. The time complexity of CDM module is $O(k)$. The time complexity of classification depends on the learning algorithm used as the base model. Therefore, MSDA has total time complexity of $O(kN_m + k) + f(k)$, where $f(k)$ is the time complexity for training a new classification model.

Experiment

Baseline Methods

To evaluate the effectiveness of our proposed our method MSDA, which uses embedding based methods for domain adaptation, we compare it with several state-of-the-art domain adaptation or online learning frameworks. Also, we applied two different variations of our method, which are

Table 3: Comparison of performance

Error Rate	OTL	HeMap-S	HeMap-L	MSDA-S	MSDA-L
USPS → MNIST	32.31 ± 0.94	38.19 ± 0.75	39.04 ± 0.68	**28.72 ± 0.56**	29.96 ± 0.64
MUSIC → DVD	33.54 ± 0.69	36.06 ± 0.51	35.85 ± 0.77	32.91 ± 0.49	**31.64 ± 0.57**
VIDEO → MUSIC	38.01 ± 1.03	40.41 ± 0.65	39.72 ± 0.85	**31.76 ± 0.72**	32.02 ± 0.75
VIDEO → DVD	35.88 ± 0.88	40.28 ± 0.52	40.09 ± 0.61	**32.35 ± 0.49**	33.84 ± 0.65
SYN01 → SYN02	37.53 ± 1.37	41.83 ± 0.83	42.12 ± 1.06	**33.50 ± 0.97**	35.47 ± 0.92

MSDA-SVM and MSDA-LR. These two methods are different in a way that classifiers applied are SVM and Logistic Regression. The following paragraphs describe details of baseline methods.

OTL (Zhao and Hoi 2010). Online Transfer Learning (OTL) uses a co-regularized method to project target domain data into the source domain. This method is applied in a supervised manner, which means that data in source domain are offline, while data in target domain are online. Furthermore, there is a strong assumption that features in source stream are a subset of those in target stream.

HeMap (Shi et al. 2010). Heterogeneous Mapping (HeMap) projects data in two domains with correspondence onto a common latent space. This method is designed as a batch training, which means both source and target data are offline. Also, this method requires class labels in the target dataset. We adapt HeMap into our problem by applying sliding windows. Meanwhile, two classifiers, SVM and LR, are also applied in this method as variations, which are referred as HeMap-S and HeMap-L respectively.

Datasets

We use both synthetic and real-world datasets to evaluate our methods. As Table 2 shows, the first five datasets are publicly available, and the latter two synthetic datasets are generated by MOA (Bifet et al. 2010).

USPS (Hull 1994) and **MNIST** (LeCun et al. 1998) contain gray-scale images of hand-written digits collected from different sources. In order to satisfy the concept drift assumption in this paper, we shift the concept of positive and negative classes in the middle of datasets. That is, in the first half of both USPS and MNIST, labels 0-4 represent "-" and labels 5-9 represent "+", while in the second half class labels 3-7 mean "+" and the rest class labels mean "-".

MUSIC, **DVD**, and **VIDEO** (Blitzer, Dredze, and Pereira 2007) are text datasets generated by Amazon reviews based on product categories. Features are extracted from raw review text by implementing word2vec model proposed by Google (Mikolov et al. 2013). We label scores with ratings greater than 3 is defined as"+", and those smaller than 3 are defined as "-". Scores that equals to 3 is discarded due to ambiguous polarity. Meanwhile, weighted average sentence embedding (Arora, Liang, and Ma 2016) is applied to represent word vectors, which provide more weight to uncommon words.

SYN01, **SYN02** (Bifet et al. 2010) are synthetic datasets that are generated by MOA framework. These datasets are generated in a way that the number of features in SYN01 is 100 while that in SYN02 is 200, so that our domain adaptation assumption can be satisfied.

Experiment Setup

Our MSDA approach involves multiple parameters. We use $N_m = 400$ as our default setting in the experiment. Meanwhile, $\beta = 1$, $k = 4$, and $\tau = 0.1$ are selected to conduct our initial experiment here. The sensitivity of parameters will be discussed later in this section.

Result Analysis

Table 3 shows the average prediction error % on the target stream T: $\frac{A_{wrong}}{m}$, where A_{wrong} is the total number of instances identified incorrectly, and m is the number of instances in the target stream. From this table, we can tell that MSDA-S outperforms all other competing methods on almost every dataset,except MUSIC → DVD. The reason here is because in other datasets information are transferred from low-dimensional to high-dimensional feature space while this dataset is reversed. Since the performance of the model can not be simply reversed, our method works better when adapting from small feature space to larger feature space.

For example, the error rate of OTA, HeMap-S and MSDA-S on USPS → MNIST are 32.31%, 38.19% and 28.72% respectively. We can see that our proposed method has better performance by a significant margin compared to baseline methods. The reason is that for OTL, the algorithm has a strong assumption that features in the source data are a subset of those in the target stream, which is not quite the case here. Text dataset used in this paper, such as reviews for VIDEO and DVD, would have overlapping features in terms of data distributions. However, not all features in VIDEO would exist in the DVD dataset in our settings. When it comes to HeMap, it also has its own shortcomings. HeMap is an offline method, which makes it not able to adapt the concept drift assumption in our problem. In this dataset specifically, the concept in the second half of data shifts from the first half of data (described in datasets section). Since we applied periodic updates every 4000 instances for the HeMap method, there is a delay on updating the model comparing our proposed MSDA-S approach.

Sensitivity Analysis

The results of our proposed method are further analyzed by tuning defined parameters N_m, β, k, and τ. All experiments are conducted on USPS → MNIST dataset. In this section, we vary N_m by setting it to {200, 400, 600, 800, 1000}, β to {0.5, 1, 1.5, 2, 2.5}, k to {2, 4, 6, 8, 10}, and τ to {0.1, 0.2, 0.3, 0.4, 0.5} respectively.

First, the parameter that controls window size (N_m) is set to different values, and results are shown regarding how they affect MSDA approach in Figure 3. We can see that the average error decreases along with window size increases, while

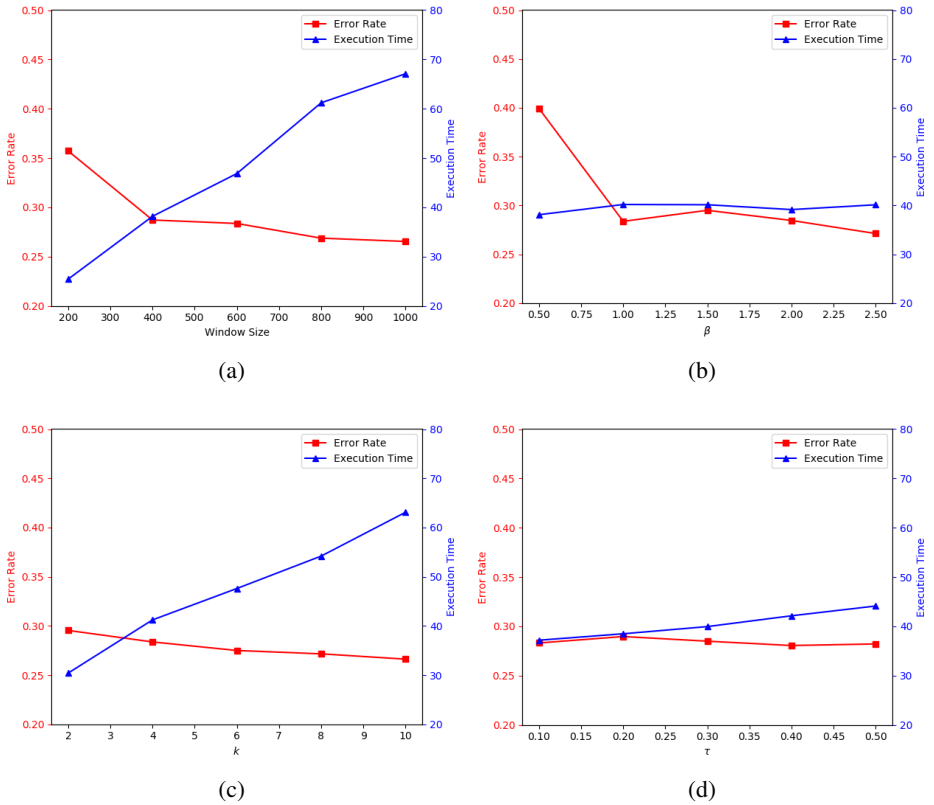

Figure 3: Sensitivity Experiment Results of MSDA

the execution time increases with increasing window size. This is close to our expectation, as we state that the execution time of MSDA depends on N_m. Based on this observation, we should choose a medium value so that both error rate and execution time could be balanced.

Second, β determines the similarities between projected domains from both source and target stream. From Figure 3, we can tell that the error rate decreases significantly when β increases from 0.5 to 1. If $\beta > 1$, the decreasing trend of error rate is becoming not significant. Also, we can see from the figure that the execution time regarding different β remains similar in the experiment. Thus, we choose $\beta = 1$ as our recommended parameter here.

Third, considering the parameter that defines the dimension of latent feature space k. This parameter influence the model in both embedding and classification. In other words, this parameter determines how large the feature space is to apply our classifiers. From Figure 3, we can see that the performance of the model has improved marginally while the execution time increases dramatically as k increases. Thus, we need to choose as small k as possible, meanwhile we need to make sure that performance doesn't degrade. Thus, we choose $k = 4$ for our approach.

At last, we can see that for τ, which controls the threshold for concept drift detection. In this case, the performance of model doesn't quite change when τ varies. Therefore, based on performance of execution time, we need to choose a smaller τ. Hence, we select $\tau = 0.1$ in our model.

In all, sensitivity experiments indicate that MSDA is sensitive to N_m and k, while not quite sensitive to β and τ.

Conclusions

In this paper, a multistream classification framework that incorporates domain adaptation techniques is proposed. Two major challenges, namely heterogeneous domain and concept drift, are addressed simultaneously in two data streams. Our solution involves an embedding-based mapping approach for domain adaptation, and an online update mechanism using average mean discrepancy for concept drift correction. More specifically, the mapping approach helps to find a common latent space for both source and target streams, which preserves the structure of data and maximizes the similarities between source and target data. Additionally, the prediction model is updated if a concept drift is detected, which happens when a likelihood ratio is greater than the user-defined threshold τ. Extensive experiments with both real-world and synthetic data show that our approach has significantly better performance in terms of error rate on various datasets, compared to existing state-of-the-art solutions.

References

Arnold, A.; Nallapati, R.; and Cohen, W. W. 2007. A comparative study of methods for transductive transfer learning. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, 77–82. IEEE.

Arora, S.; Liang, Y.; and Ma, T. 2016. A simple but tough-to-beat baseline for sentence embeddings.

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

Bifet, A., and Gavaldà, R. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*, 443–448.

Bifet, A.; Holmes, G.; Kirkby, R.; and Pfahringer, B. 2010. Moa: Massive online analysis. *Journal of Machine Learning Research* 11(May):1601–1604.

Blitzer, J.; Dredze, M.; and Pereira, F. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, 440–447.

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.

Chen, Y.-R., and Chen, H.-H. 2015. Opinion spam detection in web forum: a real case study. In *Proceedings of the 24th International Conference on World Wide Web*, 173–183. International World Wide Web Conferences Steering Committee.

Gama, J.; Medas, P.; Castillo, G.; and Rodrigues, P. P. 2004. 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*, 286–295.

Haque, A.; Tao, H.; Chandra, S.; Liu, J.; and Khan, L. 2018. 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*.

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

Herlihy, M. 1993. A methodology for implementing highly concurrent data objects. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 15(5):745–770.

Hou, B.-J.; Zhang, L.; and Zhou, Z.-H. 2017. Learning with feature evolvable streams. In *Advances in Neural Information Processing Systems*, 1417–1427.

Hull, J. J. 1994. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence* 16(5):550–554.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Long, B.; Yu, P. S.; and Zhang, Z. 2008. A general model for multiple view unsupervised learning. In *Proceedings of the 2008 SIAM international conference on data mining*, 822–833. SIAM.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22(10):1345–1359.

Shi, X.; Liu, Q.; Fan, W.; Philip, S. Y.; and Zhu, R. 2010. Transfer learning on heterogenous feature spaces via spectral transformation. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, 1049–1054. IEEE.

Zadrozny, B. 2004. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, 114. ACM.

Zhao, P., and Hoi, S. C. 2010. Otl: A framework of online transfer learning. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 1231–1238.