Multistream Classification for Cyber Threat Data with Heterogeneous Feature Space

Yi-Fan Li
The University of Texas at Dallas
Richardson, Texas
yli@utdallas.edu

Yang Gao
The University of Texas at Dallas
Richardson, Texas
yxg122530@utdallas.edu

Gbadebo Ayoade
The University of Texas at Dallas
Richardson, Texas
gga110020@utdallas.edu

Hemeng Tao
The University of Texas at Dallas
Richardson, Texas
htx160430@utdallas.edu

Latifur Khan
The University of Texas at Dallas
Richardson, Texas
lkhan@utdallas.edu

Bhavani Thuraisingham
The University of Texas at Dallas
Richardson, Texas
bhavani.thuraisingham@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 real world scenario, since data streams from different sources may contain distinct 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 given stream. Note that domains of source and target streams may have distinct features 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. This framework is also crucial for enterprise system defenders to detect cross-platform attacks, such as Advanced Persistent Threats (APTs). Empirical evaluation 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 approaches.

CCS CONCEPTS

• **Computing methodologies** → **Transfer learning**; • **Information systems** → *Data stream mining*; • **Security and privacy** → Intrusion detection systems.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313572

KEYWORDS

Attack detection; Multistream classification; Domain adaptation

ACM Reference Format:
Yi-Fan Li, Yang Gao, Gbadebo Ayoade, Hemeng Tao, Latifur Khan, and Bhavani Thuraisingham. 2019. Multistream Classification for Cyber Threat Data with Heterogeneous Feature Space. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313572

1 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, e.g., theoretically infinite in length, heterogeneous domains, and lack of label for emerging data [12, 14]. These unique properties are discussed seperately in the following context.

Regarding *data stream mining*, concept drift had been a challenge problem due to its theoretical infinite length. Detecting it 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 [13] adopt the principle [11] 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 [8] to detect changes in distribution between two different time windows or always buffering the most recent concept [19]. Apart from working on a single stream, these methods explicitly detect change points.

For *domain adaptaion*, the problem setting described in this paper is motivated by a Multistream Classification framework [8]. 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 [25]. 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

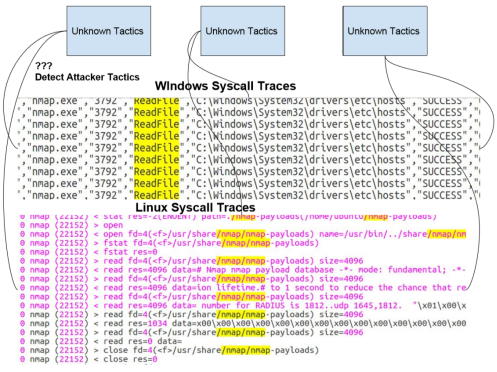

Figure 1: Sample Trace File showing syscall traces for an attacker scanning session with nmap network scanning tool

common feature representation for domain adaptation [2, 20, 23, 26]. The state-of-the-art methods, including deep learning, mostly address domain adaptation problems for stationary data except [26]. Thus, fitting domain adaptation frameworks to online data streams is yet to explore.

The multistream setting is very common in *cyber security area*. For example, organizations deploy threat monitoring tools that collect network system call events as data streams. Linking low-level traces to attacker tactics and intent is challenging. In Figure 1, a snapshot of a trace file is shown as scanning session of an attacker using the nmap tool to scan a victim's network. Also, data streams obtained from Linux machines cannot be used to predict attack events on a host with a Windows operating system. This is due to the difference in the way events are recorded on both systems (features may vary from Linux to Windows or vice versa). Machine learning algorithms have been deployed to learn attackers' behavioral patterns either using host-based [1, 6, 7, 24] or network-based [3, 10, 16, 17] techniques.

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. 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 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 in multistream setting, 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.

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

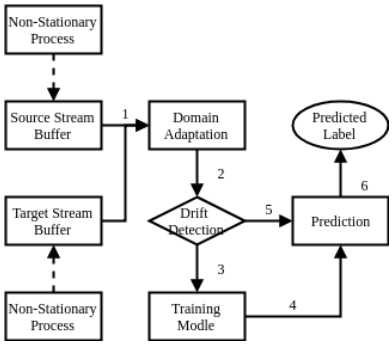

Figure 2: Multistream Classification with Domain Adaptation

(3) Our approach is empirically evaluated over several benchmark datasets, including security and non-security datasets. The results are compared with state-of-the-art methods. The performance of our proposed method is significantly better than the other three methods.

2 PROBLEM FORMULATION

2.1 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.

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: $B_s, B_t \leftarrow readData(S, T)$
3: /* DA for initial buffer */
4: $W_s, W_t \leftarrow genProjectFunction(B_s, B_t, \beta)$
5: $L_s, L_t \leftarrow genProjectMatrix(B_s, B_t, W_s, W_t)$
6: $M \leftarrow buildModel(L_s, Y_s)$
7: **while** S or T *exists* **do**
8: $B_s, B_t \leftarrow readData(S, T)$
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: $W_s, W_t \leftarrow genProjectFunction(B_s, B_t, \beta)$
15: $L_s, L_t \leftarrow genProjectMatrix(B_s, B_t, W_s, W_t)$
16: $M \leftarrow buildModel(L_s, Y_s)$
17: /* Generate predictions */
18: $\hat{y_t} \leftarrow getPrediction(M, L_t)$

2.2 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 $D_s \neq D_t$. 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.

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

3.1 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 [23]. 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:

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

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 $\|B_t - L_t W_t\|_F^2$.

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

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

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:

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

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

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: $B_s^{(N_m+1)}, L_s^{(N_m+1)} \leftarrow x, xW_s^+$.
5: $\mu_S = \frac{1}{N_m} \sum_{i=1}^{N_m} L_s^{(i)} W_s^+$.
6: Go to line 11.
7: /* Instance is from target stream */
8: $B_t^{(N_m+1)}, L_t^{(N_m+1)} \leftarrow x, xW_t^+$.
9: $B_t^{(j)}, L_t^{(j)} \leftarrow B_t^{(j+1)}, L_t^{(j+1)}, j = 1, ..., N_m$.
10: $\mu_T = \frac{1}{N_m} \sum_{j=1}^{N_m} L_t^{(j)} W_t^{-1}$.
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** $s > -ln(\tau)$.

orthogonal matrices, $B_s^\top B_s = I$. Also, we are applying the cyclic permutation property of trace. Thus, Equation 3 can be expanded by the representation of trace.

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

3.2 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:

$$\begin{cases} L_s^{(i)} = B_s^{(i)} W_s^+ & B_s^{(i)} \in B_s \\ L_t^{(j)} = B_t^{(j)} W_t^+ & B_t^{(j)} \in B_t \end{cases} \tag{4}$$

where W_s^+ and W_t^+ are the Moore-Penrose inverse of the matrix W_s and W_t respectively, which can be represented as $(W_s^\top W_s)^{-1} W_s^\top$ and $(W_t^\top W_t)^{-1} W_t^\top$.

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.

3.3 Change Detection Module (CDM)

Previous work [8] 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.

Table 2: Datasets

Data type	Datasets	# features	# instances
APT detection	Win	28	10587
	Linux	105	10801
Dark web	BlackHat	38	32414
	Nulled	38	9930
	Darkode	67	100,000
	Hack	67	100,000
Digits	USPS	256	10,000
	MNIST	780	60,000
Amazon review	Video	100	5,000
	DVD	200	30,000
	Music	300	30,000
Synthetic data	Syn01	100	10,000
	Syn02	200	20,000

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:

$$Disc(L_s, L_t) = \left\| \frac{1}{N_m} \sum_{i=1}^{N_m} L_s^{(i)} W_s^+ - \frac{1}{N_m} \sum_{j=1}^{N_m} L_t^{(j)} W_t^+ \right\|^2 \tag{5}$$

where L_s, L_t are the set of projected data points in the source and target windows.

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. And details regarding online updating algorithm for change point detection can be found in Algorithm 2.

$$S = \log \frac{Disc_t(L_s^t, L_t^t)}{Disc_0(L_s^0, L_t^0)} > \tau \tag{6}$$

3.4 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 $O(N_m^2)$. The time complexity of classification depends on the learning algorithm used as the base model. Therefore, MSDA has total time complexity of $O(kN_m + N_m^2)$.

4 EXPERIMENT

4.1 Baseline Methods

To evaluate the effectiveness of our proposed our method MSDA, 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 MSDA-SVM and MSDA-LR. These

Table 3: Comparison of performance (Error %)

Data type	Dataset	OTL	HeMap-S	HeMap-L	MSDA-S	MSDA-L
Cyber security	Win → Linux	28.53 ± 0.88	30.55 ± 0.86	29.08 ± 0.58	23.73 ± 0.62	**21.33 ± 1.62**
	BlackHat → Darkode	22.42 ± 0.85	24.70 ± 1.00	25.36 ± 1.00	22.12 ± 1.29	**20.82 ± 0.64**
	Nulled → Hack	28.86 ± 1.54	26.78 ± 0.95	29.26 ± 0.50	25.47 ± 1.06	**24.75 ± 1.36**
Non cyber security	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
Synthetic data	Syn01 → Syn02	37.53 ± 1.37	41.83 ± 0.83	42.12 ± 1.06	**33.50 ± 0.97**	35.47 ± 0.92

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

OTL [26]. 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 [23]. 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.

4.2 Datasets

As shown in Table 2, we use both synthetic and real-world datasets to evaluate our methods [4]. Specifically, we incorperate security related datasets as well as non security related datasets here to validate our proposed framework.

Win and **Linux** This dataset consists of system-call events traces from windows host machine and linux host machines. We generated various attack instances which consist of attacks e.g. malicious exfiltration of data, password theft and scanning the system and services on a victim's network with nmap. For benign instances, we generated data which include web browsing activities and remote login using ssh. We collected the system call traces using sysdig trace tool for linux and procmon tool for windows. In total, we collected 10,587 and 10,801 trace instances. With Ngram, we extract features from contiguous sequences of system calls. Our goal is to classify whether a trace in the target stream is an attack.

BlackHat, Nulled, Darkode and **Hack** [21] are datasets extracted from dark web forums where illegal items are listed and sold . Note that all forums here are English forums. Our goal is to extract words that are products on the dark web forums. To satisfy the multistream setting with different domains, for BlackHat and Nulled, we extract the features using Penn Part of Speech Tags [22]. While, for Darkode and Hack datasets, features are extracted following the Stanford pos- tagger system [9]. Since forum posts are posted within a long time period, these streams satisfies the concept drift assumption.

USPS [15] and **MNIST** [18] contain gray-scale images of handwritten 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 "-". Therefore, the concept drift assumption of data stream is satisfied in this setting.

Music, **DVD**, and **Video** [5] are text datasets generated by Amazon reviews based on product categories. We process these datasets in the same way as described in [5]. Since data in this stream covers more than one year, we assume that there are concept drift along the change of time by nature in these data streams.

Syn01, **Syn02** [4] 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. Meanwhile, we artificially add abrupt concept drift at the half position in these data streams.

4.3 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.

4.4 Result Analysis

Figure 3 shows the distribution of features in the latent space. For a single window ($N_m = 400$), we plot the transformed source and target window together in one figure. Also, t-SNE is applied here to reduce the feature space to 2D. From this figure, we can see that binary classes from both source and target stream are well separated in two part of the latent space. This good separation makes it possible to find a hyper-plane in the space between two classes, which gives us confident and evidence to apply classifiers in this feature space.

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. For security related datasets, for instance, the APT attack detection datasets, our method has 21.33% error rate, which outperforms all other benchmark algorithms. It has a better performance as the projected latent feature space. Also, our proposed framework has the best performance on identifying items on dark web forum datasets. Regarding non security datasets, we can tell that MSDA-S outperforms competing methods on almost every datasets. Also,

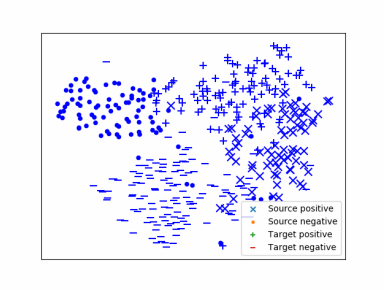

Figure 3: Latent Feature Embedding USPS→MNIST

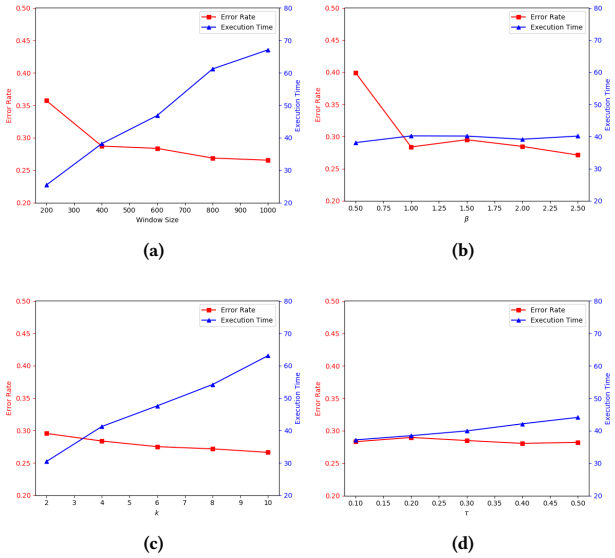

Figure 4: Sensitivity Analysis on USPS→MNIST

due to the fact that digits datasets are indeed very similar (all handwriting digits), which means that the source and target domain are pretty similar in this case, our algorithm shows best performance not only comparing to other benchmark methods, but also among all non security datastes. Furthermore, we can observe that our method works better when adapting from small feature space to larger feature space.

For example, the error rate of OTL, 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. 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.

In synthetic datasets, we observe similar phenomenon. Since the MOA framework can help generate concept drifts in the data stream, the two variations of HeMap don't perform well on this dataset. On the other hand, OTL seems to track the concept drift along the stream, however, since the dataset doesn't quite satisfy its assumption, the overall performance of MSDA-S beats other benchmarks by a significant margin.

4.5 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, we can see in Figure 4 that the average error decreases along with window size increases, while 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 4, 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, parameter k determines how large the feature space is to apply our classifiers. From Figure 4, 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.

5 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 to find a common latent space for both source and target streams, and an online update mechanism using MMD for concept drift correction. Additionally, the prediction model is updated if a concept drift is detected. 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.

ACKNOWLEDGMENTS

This material is based upon work supported by NSF award numbers: DMS-1737978, DGE 17236021. OAC-1828467; ARO award number: W911-NF-18-1-0249; IBM faculty award (Research); NSA awards; ONR award N00014-17-1-2995; and a gift from the Eugene McDermott family.

REFERENCES

[1] Khaled Al-Naami, Swarup Chandra, Ahmad Mustafa, Latifur Khan, Zhiqiang Lin, Kevin Hamlen, and Bhavani Thuraisingham. 2016. Adaptive encrypted traffic fingerprinting with bi-directional dependence. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*. ACM, 177–188.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning* 79, 1-2 (2010), 151–175.

[3] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. 2014. Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials* 16, 1 (2014), 303–336.

[4] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. 2010. Moa: Massive online analysis. *Journal of Machine Learning Research* 11, May (2010), 1601–1604.

[5] John Blitzer, Mark Dredze, and Fernando Pereira. 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.

[6] João BD Cabrera, Lundy Lewis, and Raman K Mehra. 2001. Detection and classification of intrusions and faults using sequences of system calls. *Acm sigmod record* 30, 4 (2001), 25–34.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 15.

[8] Swarup Chandra, Ahsanul Haque, Latifur Khan, and Charu Aggarwal. 2016. An adaptive framework for multistream classification. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 1181–1190.

[9] Marie-Catherine De Marneffe and Christopher D Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*. Association for Computational Linguistics, 1–8.

[10] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. 2002. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*. Springer, 77–101.

[11] João Gama, Pedro Medas, Gladys Castillo, and Pedro Pereira Rodrigues. 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.

[12] Ahsanul Haque, Latifur Khan, and Michael Baron. 2016. SAND: Semi-Supervised Adaptive Novel Class Detection and Classification over Data Stream.. In *AAAI*. 1652–1658.

[13] Ahsanul Haque, Hemeng Tao, Swarup Chandra, Jie Liu, and Latifur Khan. 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*.

[14] Bo-Jian Hou, Lijun Zhang, and Zhi-Hua Zhou. 2017. Learning with Feature Evolvable Streams. In *Advances in Neural Information Processing Systems*. 1417–1427.

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

[16] Jungwon Kim, Peter J Bentley, Uwe Aickelin, Julie Greensmith, Gianni Tedesco, and Jamie Twycross. 2007. Immune system approaches to intrusion detection–a review. *Natural computing* 6, 4 (2007), 413–466.

[17] Christopher Kruegel, Darren Mutz, William Robertson, and Fredrik Valeur. 2003. Bayesian event classification for intrusion detection. In *null*. IEEE, 14.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[19] Mohammad M Masud, Tahseen M Al-Khateeb, Latifur Khan, Charu Aggarwal, Jing Gao, Jiawei Han, and Bhavani Thuraisingham. 2011. Detecting recurring and novel classes in concept-drifting data streams. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 1176–1181.

[20] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.

[21] Rebecca S Portnoff, Sadia Afroz, Greg Durrett, Jonathan K Kummerfeld, Taylor Berg-Kirkpatrick, Damon McCoy, Kirill Levchenko, and Vern Paxson. 2017. Tools for automated analysis of cybercriminal markets. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 657–666.

[22] Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision). *Technical Reports (CIS)* (1990), 570.

[23] Xiaoxiao Shi, Qi Liu, Wei Fan, S Yu Philip, and Ruixin Zhu. 2010. Transfer learning on heterogenous feature spaces via spectral transformation. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 1049–1054.

[24] Xiaokui Shu, Danfeng Yao, and Naren Ramakrishnan. 2015. Unearthing stealthy program attacks buried in extremely long execution paths. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 401–413.

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

[26] Peilin Zhao and Steven C Hoi. 2010. OTL: A framework of online transfer learning. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 1231–1238.