Incremental Evolving Domain Adaptation

Adeleh Bitarafan, Mahdieh Soleymani Baghshah, and Marzieh Gheisari

Abstract—Almost all of the existing domain adaptation methods assume that all test data belong to a single stationary target distribution. However, in many real world applications, data arrive sequentially and the data distribution is continuously evolving. In this paper, we tackle the problem of adaptation to a continuously evolving target domain that has been recently introduced. We assume that the available data for the source domain are labeled but the examples of the target domain can be unlabeled and arrive sequentially. Moreover, the distribution of the target domain can evolve continuously over time. We propose the Evolving Domain Adaptation (EDA) method that first finds a new feature space in which the source domain and the current target domain are approximately indistinguishable. Therefore, source and target domain data are similarly distributed in the new feature space and we use a semi-supervised classification method to utilize both the unlabeled data of the target domain and the labeled data of the source domain. Since test data arrives sequentially, we propose an incremental approach both for finding the new feature space and for semi-supervised classification. Experiments on several real datasets demonstrate the superiority of our proposed method in comparison to the other recent methods.

Index Terms—Domain adaptation, evolving domains, semi-supervised learning, online learning

1 Introduction

In real world applications, the distribution of training data can be significantly different from that of test data. Therefore, in many cases, the performance of the learned classifier (on the training data) will be severely degraded on the test data. Domain adaptation techniques try to transfer information learned from the source (training) domain data to the target (test) domain data to diminish the performance degradation [1].

For example, domain shift in image classification occurs due to the difference in illumination, resolution, viewpoint, and background between samples of the source and the target domain. Domain adaptation in this application allows us to use (for example) labeled images taken under specific illumination conditions and having an uncluttered background to learn a classifier that is also useful for classifying test examples taken under different illumination conditions and background. Until now, many domain adaptation methods have been proposed to alleviate the domain shift problem for different applications such as event recognition [2], [3], video concept detection [4], [5], face recognition [6], object recognition [7], [8], [9], [10], [11], [12], content classification [13] and object detection [14], [15]. Recent methods for the classic domain adaptation problem [7], [8], [9], [10], [11] usually learn a new feature space in which domain-induced dissimilarity is minimized. In [7], both source and target data are first projected to their respective low-dimensional subspaces and then a domain-invariant kernel that minimizes the distances between these subspaces are found. Intuitively, in the new feature space, the source domain and the target domain have common characteristics and thus a classifier constructed for the source domain can also be used for the target domain.

Recently, the novel problem of adapting to continuously evolving domains is formulated in [1], [16] and domain adaptation for the evolving target has received attention. For example, [1] considers the scene classification in video stream where the appearance of the same scene class is continuously changing due to the factors such as illumination conditions (affected for example by time of day) and only the scene class of the start samples are available. Continuous Manifold domain Adaptation (CMA) method introduced in [1] tries to classify streaming data drawn from a continuously evolving visual domain, e.g., images of traffic scene taken at different time over the year. This method assumes labeled data are available for the source domain but data of the target domain are unlabeled and arrive sequentially. CMA assumes each target sample has been drawn from a distribution corresponding to a (lower dimensional) subspace on the Grassman manifold of subspaces. Since data arrive sequentially, the lower dimensional target subspace is changed incrementally and this subspace is found by a variant of the sequential Karhunen-Loeve method introduced in [17]. Therefore, CMA learns a time-varying kernel transformation between the source and the target subspace using unsupervised adaptation techniques [7], [10]. Then, classifiers such as KNN and SVM using the obtained kernel are learned using the labeled samples of the source domain. In addition to [1], a blind adaptation method has been proposed in [22] to predict the next step of a time-varying distribution by a given sequence of sample sets obtained for earlier time steps. It uses the predicted target distribution to learn a classifier for a target distribution from which no training examples are available (not even unlabeled ones). However, in the problem of adapting to continuously evolving domains that is introduced in [1], it

• *The authors are with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. E-mail: adele.bitarafan@gmail.com, soleymani@sharif.edu, gheisari@ce.sharif.edu.*

Manuscript received 24 Sept. 2015; revised 4 Jan. 2016; accepted 20 Mar. 2016. Date of publication 6 Apr. 2016; date of current version 5 July 2016. Recommended for acceptance by J. Ye.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2016.2551241

is assumed that target domain data are available but they are unlabeled.

In this paper, we address the problem of adaptation to a continuously evolving target distribution. We assume that the available data for the source domain are labeled but the examples of the target domain can be unlabeled (or labeled) and arrive sequentially. In the proposed method, we first find a new feature space in which the source domain and the current target domain are approximately indistinguishable. Since source and target data are similarly distributed in the new feature space, we use a semi-supervised classification method to use both the unlabeled data of the target domain and the labeled data of the source domain. Indeed, continuously evolving data distributions lie on a manifold and we can use the intrinsic structure revealed by all arriving data in our semi-supervised classifier. In the proposed method, an incremental approach is used both for finding the new feature space and for semi-supervised classification.

We can summarize the main contributions of our paper as:

- We first find a new representation space for the source and target domains that is an indistinguishable feature space for these domains. In the new feature space, we use a semi-supervised classification method to utilize both the unlabeled data of the target domain and the labeled data of the source domain. Indeed, in the previous domain adaptation methods like [1] only supervised classifiers on the samples of the source domain have been found and used for classifying data of the target domain. We utilize the property of the continuously evolving target domain that results in a manifold underlying the arrived samples at different times and use this intrinsic structure to propagate labels of the source domain in the subsequent samples properly.
- We use an incremental approach both for finding the new feature space and for semi-supervised learning.
- When new data arrive, the evolving source subspace is updated using the incremental PLS as a supervised dimensionality reduction method instead of the unsupervised SVD method. An advantage of our method is that it is applicable even when too much data arrive at time t as opposed to the CMA method that cannot work for big batches since updating the subspace in this method cannot be done for large number of samples.
- The supervised and semi-supervised versions of our method that can use the labels of all (or some of) the arriving data except to the data in the last batch are also proposed and they are appropriate for stream data classification. To handle concept drift, our method utilizes an incremental domain adaptation model and thus it can classify stream data in non-stationary environments.

We have organized this paper as follows. First, in Section 2, we review the related work about adaptation for evolving domains and stream classification with concept drift. We will introduce our proposed method in Section 3. The detailed experiment setup and results are summarized in Section 4. Section 5 concludes this paper.

2 Related Work

In the proposed method, we can use unlabeled evolving data to adapt the model without receiving any additional labels. However, the supervised (semi-supervised) version of our method that can also use labels of all (or some) arriving data except to the last batch has been proposed. These methods can be compared with stream classification methods. Since our work is related to both domain adaptation and stream classification methods, in this section, we review both of them.

2.1 Stream Data Classification

In many real applications, such as spam identification, climate monitoring, credit fraud, network intrusion detection, huge volume of data arrives continuously. Such applications are usually modeled as stream classification problems. Other than the huge data volume in data streams, a common challenge of stream classification is that streaming data are often generated by a non-stationary process where this happening is known as concept drift. In the recent years, a large number of classification methods for stream data with concept drift have been proposed. However, most of these methods such as [18], [19], [20], [21], [22], [23], [24], [25] are only appropriate for supervised tasks in which the labels of all the arriving data except to the data in the last batch are available. [26], [27] are methods proposed in the field of semi-supervised classification on non-stationary data streams that utilize only labels of a small proportion of data. In general, stream classification techniques can be subdivided into single model [18], [19] and ensemble ones [20], [21], [22], [23], [24], [26], [27].

The single model classification techniques [18], [19] incrementally update their model when new data arrives. These methods usually utilize only the most recent data to update their model and older instances are forgotten since new instances have larger effect than the older ones for the classification of current new data. By controlling how fast old instances are forgotten, the single model classification techniques enable us to handle concept drift.

Recently, ensemble techniques for stream data classification have received much attention. The principle behind these methods is to combine individual classifiers in order to find a more accurate classifier. Until now, several ensemble methods [20], [21], [22], [23], [24], [26], [27] that differ in the way of forgetting the data and selecting base classifiers have been proposed. The introduced methods in [21], [22], [23], [26] try to improve classifying data streams including concept drift by exploiting the existence of recurring concepts. These methods are based on maintaining a pool of classifiers that are updated continuously when consecutive batches of data are received. When a new batch of data is arrived, first the instances are classified using the existing classifiers and after receiving the true labels of these instances, an existing classifier in the pool is updated or a new classifier is added to the pool. Although the stream classification methods also work on sequentially arriving data, they need the true labels of arrived data. In fact, they cannot be carried out our unsupervised setting in which labels exist only for the source domain (i.e., the start batch or batches).

2.2 Domain Adaptation and Adapting to Evolving Domains

The goal of domain adaptation methods is to use labeled data of the source domain and unlabeled data of the target domain to find a classifier that can predict labels of the data obtained from the target domain whose underlying distribution differs from that of the source domain data. In the last years, domain adaptation has been widely studied and used in many different applications such as event recognition [2], [3], video concept detection [4], [5], face recognition [6], object recognition [7], [8], [9], [10], [11], [12], content classification [13] and object detection [14], [15].

Many recent methods for domain adaptation try to construct data representation in which the discrepancy between source and target distributions is minimized by means of a linear projection [8], [9], [10] or a kernel function [7]. Some recent works [7], [8], [9] have shown the utility of considering discrete visual domains as fixed points embedded in a manifold of lower-dimensional subspaces. The manifold-based methods introduced in [8], [9] propose a common intermediate feature representation which is achieved by projecting the data to a sequence of subspaces sampled along the path between the source and the target domain on the Grassmann manifold of subspaces. Recently, GFK method [7] extended this approach to incorporate the infinite set of intermediate subspaces and defined a new kernel equivalent to integrating over all these intermediate subspaces that lie on the geodesic flow connecting the source and the target subspace [7]. However, all of these methods assume sufficient data from the source and the target domain are available for modeling the source and the target data distribution and none of them has been designed for evolving domains and online adaptation.

Until now some special purpose incremental adaptation methods have been proposed. The parameter adaptation-based method introduced in [14] focuses on the incremental domain adaptation of deformable part-based models (DPMs) for object detection using a little amount of labeled data available for the target domain. This method, similar to the introduced ones in [15], is an ensemble method which learns a classifier using the available data of the target domain, and combines this classifier with the trained classifier on the source domain data. The online domain adaptation method [15] has a similar setting to [14], while the employed Gaussian Process Regression requires a large number of labeled samples from the target domain. The online adaptation method introduced in [28] learns a single stationary target distribution from streaming unlabeled observations. Although these methods are online learning methods, they assume the target distribution is not evolving. Thus, they are not useful for object recognition applications where domains are continuously evolving.

Recently, the novel problem of adapting to continuously evolving domains is formulated in [1], [16] and the Continuous Manifold domain Adaptation method has been proposed for this purpose. As opposed to the traditional online learning methods and stream classification methods, in the problem of the adaptation to evolving domains, there is no need to access labels of arriving data in all the past times. Indeed, only data at some point in the past are labeled (as source domain) and data arrived in the next times are unlabeled. CMA method first learns a time-varying kernel transformation between the source and the evolving target subspace using unsupervised adaptation techniques [7], [10] and then finds a classifier (such as KNN or SVM) on the source domain data to classify the current test (or target domain) data.

3 Proposed Method

Assume that we have a continually evolving data and only the arrived data in the first time step are labeled. When the distribution of data is changing over time, a significant mismatch appears between the source domain containing training data and the target domain including test data. We use incremental domain adaptation approach to reduce disagreement between the source and the evolving target domain.

The proposed method has used the ideas of Geodesic Flow Kernel (GFK) method [7] introduced for the classic domain adaptation and also incremental semi-supervised classification to use all evolving data. Let $\boldsymbol{B}_i \in \mathbb{R}^{b\times D}$ denote the batch arrived in the time i that contains b samples in the D dimensional feature space. In addition, we assume that each batch includes a fixed number of instances. We show all the instances arrived until the time t as a vector:

$$\boldsymbol{D}_t \in \mathbb{R}^{t\times b\times D} = \{\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_{t-1}, \boldsymbol{B}_t\}. \quad (1)$$

For the unsupervised evolving domain adaptation, we have only the labels of the data in the first batch $\boldsymbol{B}_1$. However, for the supervised version, labels of the instances in the last batch aren't available while instances of the other batches are labeled. Thus, we can see the stream classification problem as a special case of the adaptation to evolving data. However, as opposed to the unsupervised adaptation to evolving data that only access to the labels of the examples in the first time step, in the stream classification problem the labels of all previously received data are available.

In the proposed method, we assign weights reflecting the arrival time of the instances to them. Indeed, the i -th data point that is arrived in the time t is weighted according to an exponential function $\lambda_i = \exp(-t/T)$. The parameter T controls how fast the weights are decreased. If $T = \infty$, all the examples have the same weight.

Let labeled batches be the source domain and the unlabeled batch be the target domain that can have a different distribution from the source domain. To reduce disagreements between the source and the target domain, we seek to learn a linear transformation $\boldsymbol{W}$ that maps instances into a new feature space in which the distribution of the source and the target domain is more similar. Recent work has shown the utility of considering discrete visual domains as fixed points embedded in a manifold of lower-dimensional subspaces. Adaptation can be achieved via transformations to provide invariant feature space for the source and the target domain. To find $\boldsymbol{W}$, we use Geodesic Flow Kernel [7] domain adaptation method (you can see Section 3.1 for more details). After finding a new feature space, we use an Incremental Semi-Supervised Learning (ISSL) method based on manifold regularization that will be introduced in Section 3.2 to predict labels of unlabeled data. The proposed algorithm has been shown in Fig. 1.

Input: Receive the first labeled batch $\boldsymbol{B}_1$.
Set requirement parameters such as T, γ_A, γ_B, size of the batches (b), the number of eigenvectors (k), and the subspace dimension d.
Output: Predict labels of instances of all unlabeled batches.

1. Initialization: source subspace $\boldsymbol{P}_s$ of the first labeled batch is computed by PLS.
   Source domain= $\{\boldsymbol{B}_1\}$, $t=1$.

Loop {Do this loop for each new batch arrived }

2. $t=t+1$.
3. Receive the t-th batch of unlabeled data $\boldsymbol{B}_t$ as the current target domain= $\{\boldsymbol{B}_t\}$.
4. Compute target subspace $\boldsymbol{P}_t$ of the target domain by the PCA method.
5. Construct a linear transformation $\boldsymbol{W}$ by the GFK algorithm and transform and normalize samples in $\boldsymbol{B}_t$ accordingly (Section 3.1).
6. Compute the weight of the instances in the batch $\boldsymbol{B}_t$ as $\exp(-t/T)$.
7. Predict the labels of the instances in the batch $\boldsymbol{B}_t$ by the proposed incremental semi-supervised learning method (Section 3.2).
8. In the unsupervised version of our algorithm, add instances of the last batch whose labels were predicted in Step 6 to the current source domain.

8′. In the supervised (or semi-supervised) version of our algorithm, the actual labels of all (or some of) the instances in the batch $\boldsymbol{B}_t$ are received and these instances are added to the source domain with the actual labels.

9. In the unsupervised version of our algorithm, update the source subspace $\boldsymbol{P}_s$ incrementally (Section 3.1.1.2) by incorporating the instances of the last batch whose labels are predicted using our ISSL method.

9′. In the supervised version of our algorithm, update the source subspace $\boldsymbol{P}_s$ incrementally by incorporating the instances of the last batch with their true labels.

End Loop

Fig. 1. Evolving domain adaptation algorithm.

In the above algorithm, after receiving the t-th unlabeled batch B_t, we first consider it as the target domain and compute its subspace using the PCA method and then find a transformation of the source and the target domain data to a new feature space (in the step 5). In the step 7, we predict the labels of the samples in the target domain. Then, we update the source domain and its subspace in the steps 8-9. In the following sections, we describe the details of these steps.

3.1 Finding a Linear Transformation

As mentioned above, each unlabeled batch arrived in the time t is first transformed into a new feature space by a linear transformation $\boldsymbol{W}$ constructed by Geodesic Flow Kernel as a classic domain adaptation method [7]. GFK is a new invariant feature representation method that exploits intrinsic low-dimension structures of data.

GFK assumes that the source and the target domain are approximately lying on the d-dimensional $(d \ll D)$ orthogonal subspaces P_s and $P_t \in \mathbb{R}^{D\times d}$ respectively, which are points on the Grassmann manifold. The final transformation is found by projecting and integrating over the infinite set of all intermediate subspaces between the source and the target domain. Therefore, after projecting and integrating over the infinite set of the intermediate subspaces $(\Phi: t \in [0,1])$, the dot product of the D-dimensional feature vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the new transformed space is computed as:

$$\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{G} \boldsymbol{x}_j = \int_0^1 (\boldsymbol{\Phi}(t)^{\mathrm{T}} \boldsymbol{x}_i)^{\mathrm{T}} \boldsymbol{\Phi}(t)^{\mathrm{T}} \boldsymbol{x}_j \, dt, \tag{2}$$

where $\boldsymbol{G} \in \mathbb{R}^{D\times D}$ is a positive semi-definite matrix that can be computed in a closed-form as in [7]. We use the $\boldsymbol{G}$ matrix as the linear transformation $\boldsymbol{W}$ in Step 5 of the proposed algorithm shown in Fig. 1.

To compute the target subspace corresponding to the target domain, i.e., P_t, we use the Principal Component Analysis (PCA) since there is no label for the target domain. However, since labels of all the training data are available for the source domain, we can find the subspace P_s for the source domain using the Partial Least Square (PLS) as a supervised linear dimensionality reduction method. Since the labeled data are growing over time, we propose the online PLS model in Section 3.2 to avoid high computational and storage requirements. Indeed, the proposed online PLS updates the subspace incrementally (without running PLS from scratch) when new data arrive or older data are removed.

3.2 Updating the Source Domain Subspace

In this section, we describe how to update the source domain (for the next time step) after receiving a new batch (Step 8-9 of Fig. 1). To this end, we first add the samples of the last batch to the source domain (Step 8 of Fig. 1). We have the actual labels of these samples in the proposed supervised method and their predicted labels (found in Step 7 of Fig. 1) in our unsupervised algorithm. Then, we update the subspace of the source domain in Step 9 required in the GFK algorithm (Step 5 of Fig. 1) in the next iteration. To this end, we use the PLS algorithm to compute the subspace P_s of the current source domain. The batch PLS algorithms [29], [30] require maintaining all the training data and retraining the model when new training data arrive or older ones are removed. Therefore, these methods are not practical for streaming data in real-world applications due to their high computational requirement.

Since data are growing over time, we use the online PLS algorithm. In the supervised version of our method, we update the source subspace P_s using the arrived labeled data in the last time step incrementally (since we assume that after receiving a batch of data and predicting labels of its instances, the actual labels are received). Similarly the effect of the older data is decrementally removed. Indeed, during the time, we have more and more labeled data for the classification and very old data are discarded from the source domain. It must also be mentioned that in the unsupervised version of our method, the source subspace P_s is updated incrementally using the last batch of samples whose labels have been predicted by our ISSL method that will be introduced in Section 3.2.

Recently, methods have been proposed for incrementally updating a PLS model [31]. In the following sections, we first briefly explain the incremental PLS model proposed in [31] which can be updated when new data arrive. Then, we propose a decremental PLS model that is used when training data can be removed decrementally. Indeed, we introduce

this decremental method to remove old data when we intend to find the latent components of the current source domain.

3.2.1 Partial Least Squares

The basic idea of the PLS method is to find uncorrelated latent components using both the input matrix **X** including features and the output matrix **Y** containing response variables. The PLS method, similar to PCA, finds latent components, however it creates orthogonal weight vectors by maximizing the covariance between **X** and **Y**. Therefore, it can be considered as a supervised dimensionality reduction method.

Assume X is an $n\times p$ zero-mean matrix. The PLS methods try to find a linear decomposition from X into:

$$X = TP^T + E, \tag{3}$$

where T is an $n\times r$ matrix (of latent components) that the number of extracted latent vectors, i.e., r, depends on the rank of the matrix X. P is a $p\times r$ orthogonal loading matrix, and E shows the error terms. The decompositions of X is finalized so as to maximize the covariance between T and Y.

Some procedures exist to solve the PLS problem, such as PLS1 [29] and SIMPLS [30]. In this paper, we limit our discussion to the most frequently used PLS1 algorithm. The PLS1 constructs latent components T by projecting X on a set of weight vectors (or projection vectors) $W = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_r\}$ such that

$$\boldsymbol{t}_i = X\boldsymbol{w}_i\ , \qquad i = 1, \ldots, r, \tag{4}$$

where $\boldsymbol{t}_i$ is the i th column of the matrix T. The PLS1 aims to maximize the covariance between the response variable Y and the latent components by a sequential approach.

3.2.2. PLS Model and Its Incremental Updating

PLS algorithms (such as PLS1 [29] and SIMPLS [30]) are iterative and until now some methods [31] have been proposed to update a PLS model incrementally. In this paper, we limit our discussion to the online PLS1 algorithm which can be updated incrementally (without retraining) when a new datum arrives.

Assume that training data are available sequentially, $\{\boldsymbol{x}_1, y_1\}, \{\boldsymbol{x}_2, y_2\}, \ldots$ where $\boldsymbol{x}_i$ denotes the feature vector of the i -th instance and y_i denotes the corresponding class label. Let $\bar{\boldsymbol{x}}_n$, $\bar{y}_n$ and C_n respectively show the mean of feature vectors, the mean of class labels, and the covariance matrix obtained for the first n instances. Moreover, $X_n = [\boldsymbol{x}_1 - \bar{\boldsymbol{x}}_n, \ldots, \boldsymbol{x}_n - \bar{\boldsymbol{x}}_n]$ contains the mean removed feature vectors and $\boldsymbol{y}_n = [y_1, \ldots, y_n]$ includes the labels of the first n instances. To compute weight vectors, a new variable $\boldsymbol{v}_n = X_n^T\boldsymbol{y}_n$ that shows the leading projection direction has been defined in [31] and the projection direction $\boldsymbol{w}_n$ is found as $\boldsymbol{w}_n = \boldsymbol{v}_n/||\boldsymbol{v}_n||$. When a new datum arrives, we require to update incrementally the above mentioned elements $\bar{\boldsymbol{x}}_n$, $\bar{y}_n$, C_n, and $\boldsymbol{v}_n$ in the incremental PLS algorithm [31]. Incremental updating of these elements is straightforward:

$$\begin{aligned}\bar{\boldsymbol{x}}_n &= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i = \frac{1}{n}\left[\sum_{i=1}^{n-1}\boldsymbol{x}_i + \boldsymbol{x}_n\right] = \\ &\frac{1}{n}[(n-1)\bar{\boldsymbol{x}}_{n-1} + \boldsymbol{x}_n] = \frac{n-1}{n}\bar{\boldsymbol{x}}_{n-1} + \frac{1}{n}\boldsymbol{x}_n.\end{aligned} \tag{5}$$

Similarly, $\bar{y}_n$ can also be updated as:

$$\bar{y}_n = \frac{1}{n}\sum_{i=1}^{n} y_i = \ \frac{n-1}{n}\bar{y}_{n-1} + \frac{1}{n}y_n. \tag{6}$$

The covariance matrix C_n can be updated using the following equation:

$$\begin{aligned}C_n = X_n^T X_n &= \frac{1}{n}\sum_{i=1}^{n-1}[(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_n)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_n)^T] \\ &+ \frac{1}{n}[(\boldsymbol{x}_n - \bar{\boldsymbol{x}}_n)(\boldsymbol{x}_n - \bar{\boldsymbol{x}}_n)^T] \\ &= \frac{n-1}{n}C_{n-1} + \frac{n-1}{n}\boldsymbol{\Delta}_n\boldsymbol{\Delta}_n^T + \frac{1}{n}[(\boldsymbol{x}_n - \bar{\boldsymbol{x}}_n)(\boldsymbol{x}_n - \bar{\boldsymbol{x}}_n)^T],\end{aligned} \tag{7}$$

where $\boldsymbol{\Delta}_n = \bar{\boldsymbol{x}}_n - \bar{\boldsymbol{x}}_{n-1}$ shows the change of the mean vector $\bar{\boldsymbol{x}}_n$. The estimation of the leading projection direction at step n is given as:

$$\begin{aligned}\boldsymbol{v}_n = X_n^T\boldsymbol{y}_n &= \sum_{i=1}^{n} y_i(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_n) \\ &= \sum_{i=1}^{n-1} y_i(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_n) + y_n(\boldsymbol{x}_n - \bar{\boldsymbol{x}}_n) \\ &= \boldsymbol{v}_{n-1} - (n-1)\bar{y}_{n-1}\boldsymbol{\Delta}_n + y_n(\boldsymbol{x}_n - \bar{\boldsymbol{x}}_n).\end{aligned} \tag{8}$$

For additional details about incremental PLS method, you can refer to [31].

3.2.3 Decremental Updating of the PLS Model

When the number of arrived data is increased, it is common to remove some older samples. Now we need to update the PLS model after removing a training data. To remove the effect of the older data from the source subspace P_s, we'll propose the Decremental PLS (DecPLS) method for updating P_s decremetally. This is a straightforward extension of the incremental updating procedure discussed in the previous section. Let $\bar{\boldsymbol{x}}_{n\backslash r}$ be the mean of the first n data except to the rth datum. Similarly, $C_{n\backslash r}$ and $\boldsymbol{v}_{n\backslash r}$ respectively show the covariance matrix and the leading projection direction of the stream data in which the r-th arrived datum has been removed. We can update the above elements as follows:

$$\begin{aligned}\bar{\boldsymbol{x}}_{n\backslash r} &= \frac{1}{n-1}\sum_{\substack{i=1\\ i\neq r}}^{n}\boldsymbol{x}_i = \frac{1}{n-1}\left[\sum_{i=1}^{n}\boldsymbol{x}_i - \boldsymbol{x}_r\right] \\ &= \frac{1}{n-1}[n\bar{\boldsymbol{x}}_n - \boldsymbol{x}_r] = \frac{n}{n-1}\bar{\boldsymbol{x}}_n - \frac{1}{n-1}\boldsymbol{x}_r.\end{aligned} \tag{9}$$

$$\begin{aligned}C_{n\backslash r} &= \frac{1}{n-1}\sum_{i=1}^{n}[(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_n)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_n)^T] \\ &- \frac{1}{n-1}[(\boldsymbol{x}_r - \bar{\boldsymbol{x}}_{n\backslash r})(\boldsymbol{x}_r - \bar{\boldsymbol{x}}_{n\backslash r})^T] \\ &= \frac{n}{n-1}C_n + \frac{n}{n-1}\boldsymbol{\Delta}'_n\,\boldsymbol{\Delta}'^T_n - \frac{1}{n-1}[(\boldsymbol{x}_r - \bar{\boldsymbol{x}}_{n\backslash r})(\boldsymbol{x}_r - \bar{\boldsymbol{x}}_{n\backslash r})^T].\end{aligned} \tag{10}$$

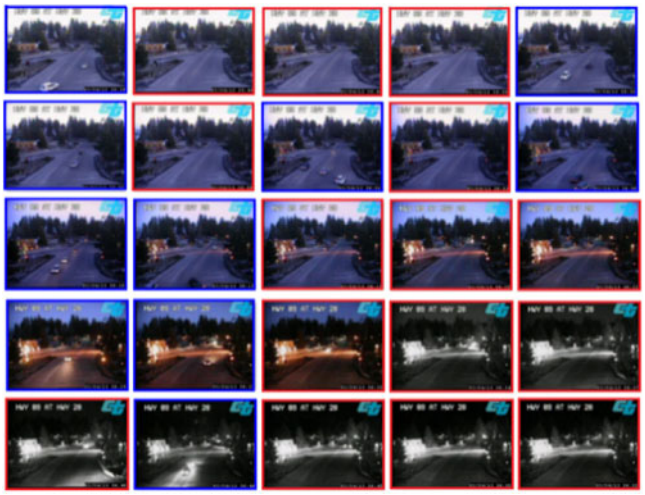

Fig. 2. Traffic scenes streaming from a traffic camera that must be classified as busy (blue border) or empty (red border) [1].

$$\begin{aligned}\boldsymbol{v}_{n\setminus r} &= \sum_{i=1, i\neq r}^{n} y_i\big(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_{n\setminus r}\big) = \sum_{i=1}^{n} y_i\big(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_{n\setminus r}\big) - y_r\big(\boldsymbol{x}_r - \bar{\boldsymbol{x}}_{n\setminus r}\big)\\ &= \sum_{i=1}^{n} y_i\big(\boldsymbol{x}_i - \big(\boldsymbol{\Delta}'_n + \bar{\boldsymbol{x}}_n\big)\big) - y_r\big(\boldsymbol{x}_r - \bar{\boldsymbol{x}}_{n\setminus r}\big)\\ &= \sum_{i=1}^{n} y_i(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_n) - \sum_{i=1}^{n} y_i\boldsymbol{\Delta}'_n - y_r\big(\boldsymbol{x}_r - \bar{\boldsymbol{x}}_{n\setminus r}\big)\\ &= \boldsymbol{v}_n - n\bar{y}_n\boldsymbol{\Delta}'_n - y_r\big(\boldsymbol{x}_r - \bar{\boldsymbol{x}}_{n\setminus r}\big),\end{aligned} \tag{11}$$

where $\boldsymbol{\Delta}_n{}' = \bar{\boldsymbol{x}}_{n\setminus r} - \bar{\boldsymbol{x}}_n$ shows the change of the mean vector.

In the proposed method, we'll remove the oldest datum from the source domain in front of receiving 10 recent data and update the source subspace decrementally using the above DecPLS method.

3.3 Prediction by Incremental Semi-Supervised Learning

In the proposed method, after finding a transformation to a new invariant feature space, we can use both labeled (source) data and unlabeled (target) data to learn a classifier in this new invariant space (Step 7 of Fig. 1). A continuously evolving target domain contains data that lie on a manifold (since closer data points in the sequence are usually more similar) and we can use this property to predict labels by semi-supervised classification methods based on manifold regularization. Indeed, we can propagate the labels of source domain data through the underlying structure and predict labels of arriving data during the time. For example, consider images taken from a fixed traffic camera observing an intersection (see Fig. 2). As can be illustrated in Fig. 2, data distribution is continuously and gradually changing due to sunlight/shadows, change to the night time IR mode and unexpected weather patterns in the same scenes types (class). In this case, we intend to classify traffic scenes streaming from a traffic camera as busy (blue border) or empty (red border).

The manifold regularization framework has demonstrated state-of-the-art performance for semi-supervised learning. However, more existing methods for this purpose cannot handle large number of examples and are not such scalable. In the proposed method, we use the ideas of semi-supervised learning method proposed in [32], [33] and propose an Incremental Semi-Supervised Learning method based on manifold regularization to solve the evolving domain adaptation problem.

Assume that the set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)\}$ shows labeled data and $\{\boldsymbol{x}_{l+1}, \ldots, \boldsymbol{x}_n\}$ shows unlabeled ones in the new feature space found using the GFK method (Step 5 of Fig. 1). Consider a graph whose vertices are data points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and edges are represented by an $n\times n$ weight matrix $\boldsymbol{S}$. Let S_{ij} be the weight between instances i, j that can be computed as $S_{ij}= \exp(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^2/2\sigma^2)$.

One of the most popular optimization problems for semi-supervised classification that is based on manifold regularization introduced in [33] is formulated as:

$$\begin{aligned}f^* &= \operatorname*{argmin}_f \sum_{i=1}^{l} \big(f(\boldsymbol{x}_i) - y_i\big)^2\\ &\quad + \frac{\gamma_A}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} S_{ij}\big(f(\boldsymbol{x}_i) - f\big(\boldsymbol{x}_j\big)\big)^2 + \gamma_B||f||_K^2\\ &= \operatorname*{argmin}_f \sum_{i=1}^{l} \big(f(\boldsymbol{x}_i) - y_i\big)^2 + \gamma_A\boldsymbol{f}^T\boldsymbol{L}\boldsymbol{f} + \ \gamma_B||f||_K^2,\end{aligned} \tag{12}$$

where f shows the function that predicts labels and $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)]^T$. Moreover, $\boldsymbol{L}$ denotes the graph Laplacian that is computed as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{S}$, where $\boldsymbol{D}$ is a diagonal matrix whose diagonal elements are $D_{ii} = \sum_{j=1}^{n} S_{ij}$. The above semi-supervised learning method tries to find a function f that agrees with labels on the labeled data while is smooth with respect to the similarity graph of data points. In the problem (12), the first term shows the training loss, the second term determines the smoothness of f on the manifold, and the third term penalizes the RKHS norm that imposes smoothness conditions on f in the associated RKHS $\mathcal{H}_K$.

Inspired by the above semi-supervised classification method, the optimization problem of our semi-supervised method is defined as follows:

$$\begin{aligned}f^* &= \operatorname{argmin}_f \sum_{i=1}^{l} \lambda_i(f(\boldsymbol{x}_i) - y_i)^2 + \gamma_A\boldsymbol{f}^T\boldsymbol{L}\boldsymbol{f} + \gamma_B||f||_K^2\\ &= \operatorname{argmin}(\boldsymbol{f} - \boldsymbol{y})^T\boldsymbol{\Lambda}(\boldsymbol{f} - \boldsymbol{y}) + \gamma_A\boldsymbol{f}^T\boldsymbol{L}\boldsymbol{f} + \gamma_B||f||_K^2,\end{aligned} \tag{13}$$

where λ_i shows the weight of the i th data point that is computed according to the arrival time of this data point and $\boldsymbol{\Lambda}$ is the diagonal matrix in which diagonal elements are $\Lambda_{ii} = \lambda_i$ if the ith data point is a labeled example and $\Lambda_{ii} = 0$ otherwise. Moreover, γ_A and γ_B are regularizer weights. Indeed, in the Problem (13), the first term is a weighted training loss in which older labeled data in the sequence of data get smaller weights, the second term is the manifold regularizer (intrinsic regularizer) which encourages the prediction smoothness over the graph (i.e., similar data tend to have same predictions) and the third term shows the complexity of the function f. Since the number of arrived labeled data is increasing in the supervised and semi-supervised scenarios and thus we have more labeled data in the later time steps, we use a forgetting mechanism to decrease the effect of older data points.

The solution of the problem (13) can be obtained using the equation $(\Lambda + \gamma_A L + \gamma_B I)\boldsymbol{f} = \Lambda \boldsymbol{y}$, where I is the identity matrix. Although this solution can be given in the closed form, it requires solving an $n \times n$ system of linear equations and so can be very expensive when n is large. We intend to use the ideas introduced in [32] to find highly efficient approximate solution to the proposed problem. Let $\boldsymbol{\Phi}_i$ and σ_i be the generalized eigenvectors and eigenvalues of L (i.e., $L\boldsymbol{\Phi}_i = \sigma_i D\boldsymbol{\Phi}_i$). Note that any vector in $\mathbb{R}^n$ can be written as:

$$\boldsymbol{f} = \sum_{i=1}^{n} \alpha_i \boldsymbol{\Phi}_i. \tag{14}$$

Moreover, eigenvectors corresponding to the smaller eigenvalues are smoother. Thus, as suggested in [32], [34], [35], the dimension can be reduced by using only a small number of the eigenvectors of the Laplacian matrix (corresponding to the smaller eigenvalues). Note that smooth vectors $\boldsymbol{f} \in \mathbb{R}^n$ will be linear combinations of the eigenvectors with small eigenvalues. Therefore, we can consider U as an $n \times k$ matrix whose columns are the k eigenvectors with the smallest eigenvalues and $\boldsymbol{f}$ can be found accordingly as $\boldsymbol{f} = U\boldsymbol{a}$. Thus, the proposed optimization problem can be reformulated as:

$$\boldsymbol{\alpha}^* = \arg\min_{\alpha} (U\boldsymbol{\alpha} - \boldsymbol{y})^T \Lambda (U\boldsymbol{\alpha} - \boldsymbol{y}) + \gamma_A \boldsymbol{\alpha}^T \boldsymbol{\Sigma} \boldsymbol{\alpha} + \gamma_B \boldsymbol{\alpha}^T \boldsymbol{\alpha}, \tag{15}$$

where $\boldsymbol{\Sigma}$ is a diagonal matrix whose diagonal elements are the smallest eigenvalues of L and $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_k]^T$. Finally, the solution can be obtained as:

$$\boldsymbol{\alpha}^* = (U^T \Lambda U + \gamma_A \boldsymbol{\Sigma} + \gamma_B \mathrm{I})^{-1} U^T \Lambda \boldsymbol{y}. \tag{16}$$

Following the work in [32] that uses the convergence of the eigenvectors of the normalized graph Laplacian to eigenfunctions of weighted Laplace-Beltrami operators, we efficiently find accurate numerical approximations to the eigenvectors based on the density structure of the data. The work in [32] instead of attempting to compute the eigenvectors of the similarity matrix of the original data ($n \times n$), estimates the eigenfunctions $\boldsymbol{g}$ of a $B \times B$ histogram matrix, where B is the number of bins in a histogram. Estimating eigenfunctions is considerably faster than computing eigenvectors of the similarity matrix of the original data, since $B \ll n$ and $\boldsymbol{g}$ can be seen as an approximation of eigenvectors of the original data when $n \to \infty$ [36]. The eigenfunctions $\boldsymbol{g}$ can be estimated by solving the following equation [32]:

$$\big(\tilde{D} - P\tilde{S}P\big)\boldsymbol{g} = \sigma\ P\hat{D}\boldsymbol{g}, \tag{17}$$

where $\tilde{S}$ is the affinity between the B discrete points, P is a diagonal matrix whose diagonal elements given the density at the discrete points, $\tilde{D}$ is a diagonal matrix whose diagonal elements are sum of the columns of $P\tilde{S}P$, and $\hat{D}$ is a diagonal matrix whose diagonal elements are the sum of the columns of $P\tilde{S}$. After computing the eigenfunctions, U that is required in Eq. (16) is found by interpolating $\boldsymbol{g}$ at each of the data.

Moreover, we follow [37] to reach an incremental version of our semi-supervised method. The work in [37] is extended the introduced method in [32] to an incremental method. At first, it builds the histogram matrix ($B \times B$) of the training data, then derives the eigenfunctions $\boldsymbol{g}$. After computing $\boldsymbol{g}$, [37] extracts the eigenvectors of the training data (the matrix U_{train}) by interpolating $\boldsymbol{g}$ at each of the training data. The histogram matrix ($B \times B$) and $\boldsymbol{g}$ are then used to compute the eigenvectors of unlabeled data (the matrix U_{test}) for each batch by interpolating $\mathbf{g}$ at each of the unlabeled data. U_{train} and U_{test} are then concatenated to produce the matrix U that is required in Eq. (16).

4 Experiments

In this section, we conduct experiments to evaluate results of our proposed method and compare them with those of the recently introduced methods. In order to evaluate our method, we consider its three versions:

1. *Unsupervised version*: In this version, we assume that only the labels of the instances in the first batch are available and all the other data are unlabeled. This means that after predicting the labels of the instances in the other batches, the actual labels of their instances are not revealed.
2. *Semi-Supervised version:* In this version, only a small fraction of instances are considered as labeled. Thus, after classifying a batch of instances, the actual labels of only some instances in that batch are revealed.
3. *Supervised version:* In this version, not only the labels of the first batch are available but also after predicting the labels of the instances in the other batches, their actual labels will also be available.

4.1 Data Sets

In order to evaluate our proposed method, we used seven real datasets as follows:

- *Spam Dataset:* This dataset consists of 9,324 email messages extracted from the Spam Assassin Collection.[1] Each email is classified as either spam or ham and represented by 500 binary features indicating the existence of a set of words (derived using feature selection). The ratio of spam message to all messages is approximately 20 percent, thus this classification problem is imbalanced. The emails are sorted according to their arrival time and their distribution gradually changes as time passes, so this dataset represents gradual concept drift.[2]
- *Traffic Dataset*: This dataset was introduced in [1] consists of 5412 instances, 512 real attributes, and 2 classes as either busy or empty. This dataset includes images captured from a fixed traffic camera[3] observing an intersection over a 2 week period. The concept drift in this dataset is due to the changes that occur in the conditions such as illumination, shadows, fog, snow, light saturation from oncoming sedans, and etc. Some sample images of this dataset have been shown in Fig. 2.

1. The Apache Spam Assassin Project – http://spamassassin.apache.org
2. Available at: http://mlkd.csd.auth.gr/concept_drift.html
3. Available at: http://quickmap.dot.ca.gov

Fig. 3. Classifying sedans versus trucks across many decades as the design and shapes of them evolve [1].

- *Weather:* This dataset includes 18159 daily readings composed of features such as temperature, pressure, and wind speed. We use the same eight features as [38]. Each instance is labeled as either rain, or no rain. The dataset exhibits recurrent and gradual concept drift.
- *Waveform21*: This dataset which is available at UCI Archive[4] has 5000 instances with 21 attributes. All of the instances include noise and three classes of waves have been generated from a combination of two of the three base waves.
- *Waveform40*: This dataset is the second version of the above dataset in which the number of the attributes for each instance has been increased to 40. However, the added 19 attributes are all noise attributes with mean zero and variance one.
- *Bank Marketing*: The data is related with direct marketing campaigns (phone cells) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (binary: yes, no). The dataset is available at UCI Archive. This dataset contains 45,211 instances, 16 numeric attributes and two classes.
- *Car:* This dataset recently proposed by [1] to classify evolving data. The data contains images of automobiles manufactured between the years of 1950-1999 acquired from a freely available online database.[5] Some sample images of this dataset have been shown in Fig. 3. Car dataset includes 1,770 samples with two class as sedan or truck where each sample has 4,096 attributes using a deep learning method introduced in [39].[6]

4.2 Experimental Setup

In this section, the selected methods for comparison, the experimental setup for parameter tuning, and the evaluation metrics are discussed.

We compare our Evolving Domain Adaptation method with the following methods:

- Conceptual Clustering and Prediction (CCP) [21],
- Pool and Accuracy Stream Classification (PASC) [23] and its extended version [22],
- Non-Parametric Stream Classifier (NPSC) [20],
- Realistic Stream Classifier (ReaSC) [27],
- Semi-supervised PASC model (SPASC) [26],
- Continuous Manifold Adaptation (CMA) [1].

In order to evaluate the supervised version of our method, we consider the first three methods (i.e., CCP, PASC, and NPSC) that are the most recent works and state-of-the-art methods for stream classification. The semi-supervised version of our method is compared with the ReaSC and SPAC that are the recent semi-supervised stream classification methods. Finally, to evaluate the unsupervised version of our method, the CMA method as the state-of-the-art unsupervised method for the adaptation to evolving domains is considered.

The parameters of CCP, PASC, NPSC, SPASC, ReaSC and CMA methods are set to the specified values in the corresponding studies. There are some parameters used in our EDA method that are needed to be set in all the three unsupervised, semi-supervised, and supervised versions of our method. In all of these settings, we consider the first 50 samples as the training data (as in [1]). The following five parameters are set the same in all the three versions of our method for all the datasets:

- γ_A and γ_B *parameters*: the regularization parameter γ_A is set to one and γ_B is set to 0.1.
- σ *parameter*: the scaling term to find the weight (or similarity) matrix between the instances is set to $\sigma = 0.2$ (as in [32]).
- d *parameter*: the number of subspace dimension is selected as $d = 100$ for the high dimensional datasets (such as car, spam and traffic dataset) and $d = D/2$ for the other datasets.
- k *parameter*: the number of eigenvectors of the graph laplacian is set to 1,000 for spam dataset and 100 for the other datasets.

Some other parameters are selected as in similar studies to the corresponding versions of our algorithm. For example, the size of batch is set to 50 (as in [21], [23], [26]) for the supervised and semi-supervised versions and set to 2 (as in [1]) for the unsupervised version on all the datasets. The T parameter shows the forgetting factor that mitigates the impact of older data is set to 100 for the supervised and the semi-supervised versions of our method since we intend to give more attention to the recent labeled data, but set $T = \infty$ for our unsupervised method since only the first batch is labeled and we don't want to forget these labels.

We compared our method with the other methods mentioned above in terms of the accuracy defined as the number of correctly classified samples divided to the whole number of samples. Indeed, we report the accuracy on the target data that we intend to correctly classify. In order to evaluate the accuracy over time, we use the prequential accuracy defined in [40]:

$$A_\alpha(t) = \sum_{\tau=1}^{t} \alpha^{t-\tau} a(\tau) \Big/ \sum_{\tau=1}^{t} \alpha^{t-\tau}, \tag{18}$$

where $A_\alpha(t)$ is the accuracy of all the arrived data until time t and $a(\tau)$ is the ratio of correctly classified samples in batch τ. α is a fading factor which is set $\alpha = 1$ (as in [21], [26]) that means the selfsame average accuracy.

Our method is implemented in Matlab and experiments are conducted on a PC with seven Core 2.5 GHz CPU, 16 GB memory and Windows 8 OS.

4.3 Experimental Results

As mentioned above, we compare our EDA approach (i.e., all three unsupervised, semi-supervised, and supervised

4. http://archive.ics.uci.edu/ml/datasets.html
5. http://www.cadatabase.net
6. We use the vectorized output of layer six of the network.

TABLE 1
Classification Accuracy (%) of the Unsupervised Version

Method	Classifier	Traffic	Car	Waveform21	Waveform40	Weather
CMA+GFK	KNN	63.22	82.50	72.48	66.85	**69.77**
	SVM	68.87	82.73	69.15	68.77	63.81
CMA+SA	KNN	41.33	56.45	33.19	33.09	68.61
	SVM	41.33	56.45	33.84	33.05	31.40
EDA		**69.56**	**82.97**	**74.65**	**80.30**	67.34

versions of it) with the related methods introduced recently. Results of these three versions are demonstrated below.

Unsupervised version. In this version, all arriving samples (after the start batch) are considered as the target domain and their labels are not used in the classifier at all. We intend to correctly classify these data that are in the target domain. The performance of our unsupervised EDA method is only compared to the CMA [1] since the problem of unsupervised adaptation to a continuously evolving data distribution has been recently introduced in [1] and CMA is the only related method that we found. As discussed in [1], CMA can use each of the unsupervised adaptation techniques GFK [7] and Subspace Alignment method (SA) [10] and also can use either k-nearest neighbor (KNN with $k = 1$) or support vector machines (SVM) (trained with only the labeled data of the source domain) to classify data. Therefore, we can compare our method with all these versions of CMA.

Since the first 50 data samples (that form the first batch) in the bank and the spam dataset consist of the samples from only one class, we can't report results of the unsupervised methods on these two datasets. Results on the other datasets are presented in Table 1. In each column, the bolded accuracy is the first best accuracy and the underlined one shows the second best accuracy.

As shown in Table 1, EDA generally outperforms all the versions of CMA. Indeed, combining the domain adaptation and semi-supervised learning techniques makes the proposed method more powerful. Moreover, to compute the source subspace, in the proposed method, we take the available or predicted label information into consideration.

Since the weather dataset has a huge data volume compared to the other datasets, when we use a large number of eigenvectors in our semi-supervised classification method, higher accuracy will be achieved. However, the results presented in Table 1 for our method have been obtained when the number of eigenvectors for all the datasets has been set to 100. Indeed, we did not use higher number of eigenvalues for larger datasets such as Weather to get better results.

Fig. 4 depicts prequential accuracy in order to evaluate the accuracy over time on the datasets. According to this figure, those versions of the CMA method using the GFK adaptation technique have more comparable results to our method. However, generally our method performs better than all the versions of the CMA method. In the traffic dataset, it's obvious that a sudden drift happened to the data in the 1,100th batch and the accuracy is slightly reduced. However, as time goes our model and some of the other methods have adapted to the new target domain.

Semi-supervised version. In the semi-supervised version of our method, we assume that the first 50 instances are considered as the source data whose labels are available and

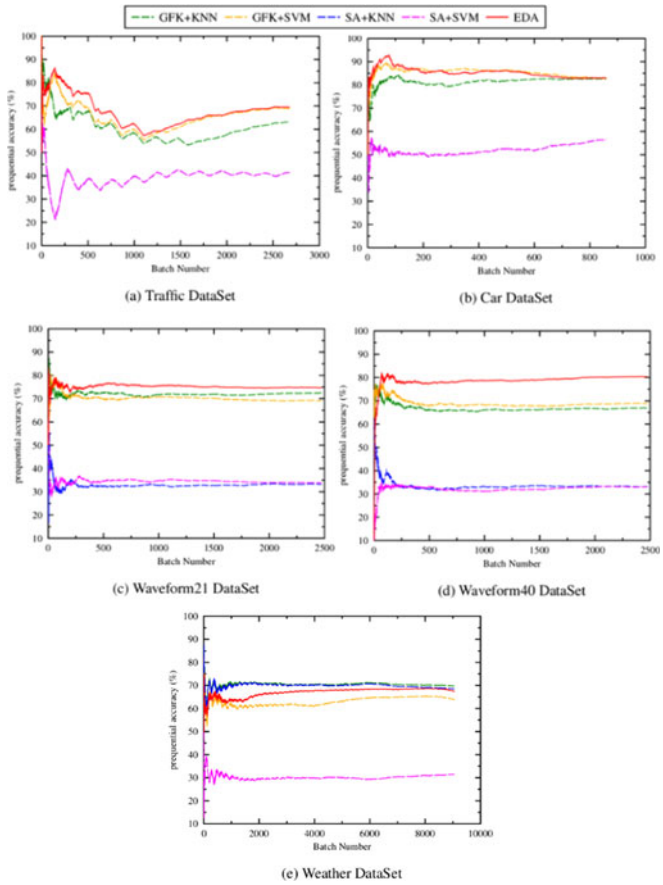

Fig. 4. Prequential accuracy of the unsupervised methods on (a) Traffic, (b) Car, (c) Waveform21, (d) Waveform40, and (e) Weather datasets.

other data are considered as the target data and after predicting their label, the true label of only 20 percent of the target data is revealed (as in [26]). The performance of our Semi-Supervised EDA (SemiEDA) method is compared with SPASC [26] and ReaSC [27] which are the recent methods proposed in the field of semi-supervised classification for non-stationary data streams. SPASC proposes two semi-supervised methods, i.e., Semi Bayesian and Semi Heuristic, and we compare our method with both of them. Results on whole target data of all the datasets are presented in Table 2 (in each column, the first best accuracy is bolded and the second best one is underlined). Since we have run all the methods 10 times (each time on a random subset of data containing 20 percent of the whole set of data that are selected to be labeled), the average and standard deviation of the accuracy have shown for each method.

According to Table 2, SemiPASC method does not perform well since it cannot adapt fast enough (using a few labeled data) when drift occurs. SemiPASC is capable to detect recurring contexts, but isn't able to detect gradual and incremental concept drifts. The results on all the datasets show the superiority of our method by a large margin compared to the second best one (ReaSC). Fig. 5 depicts the prequential accuracy on all the datasets that is useful to evaluate the accuracy over time.

It could be inferred from Fig. 5 that our method outperforms the other methods on all the batches. We also change the proportion of labeled instances of the whole data (percentage of labeled instances is changed from 10 to 100

TABLE 2
Classification Accuracy (%) of the Semi-Supervised Version

Method	Traffic	Car	Waveform 21	Waveform 40	Bank	Weather	Spam
SPASC+ Heuristic	61.79 ± 0.36	60.18 ± 0.75	40.39 ± 0.46	40.30 ± 0.45	26.65 ± 0.11	37.93 ± 0.26	77.12 ± 0.49
SPASC+ Bayesian	61.87 ± 0.49	60.16 ± 0.69	40.52 ± 0.42	40.39 ± 0.56	26.84 ± 0.15	38.14 ± 0.21	77.07 ± 0.52
ReaSC	72.23 ± 1.92	76.90 ± 1.79	67.20 ± 1.31	66.28 ± 0.51	84.45 ± 0.15	62.08 ± 0.64	84.80 ± 0.70
SEDA	$\mathbf{85.33 \pm 0.67}$	$\mathbf{88.62 \pm 1.18}$	$\mathbf{83.65 \pm 0.30}$	$\mathbf{82.74 \pm 0.47}$	$\mathbf{89.72 \pm 0.08}$	$\mathbf{74.52 \pm 0.30}$	$\mathbf{94.10 \pm 0.38}$

percent) and show the obtained results in Fig. 6. As it is expected, the accuracy of all these methods is improved by increasing the proportion of labeled data according to Fig. 6. SemiEDA outperforms SemiPASC and ReaSC for all percentage of labeled instances on all the datasets. Therefore, we can conclude that SemiEDA method is able to use unlabeled instances effectively.

One of the advantages of our proposed method (compared to SemiPASC and ReaSC) is that SemiEDA method can classify test data with highest accuracy, even if the percentage of labeled instances is low.

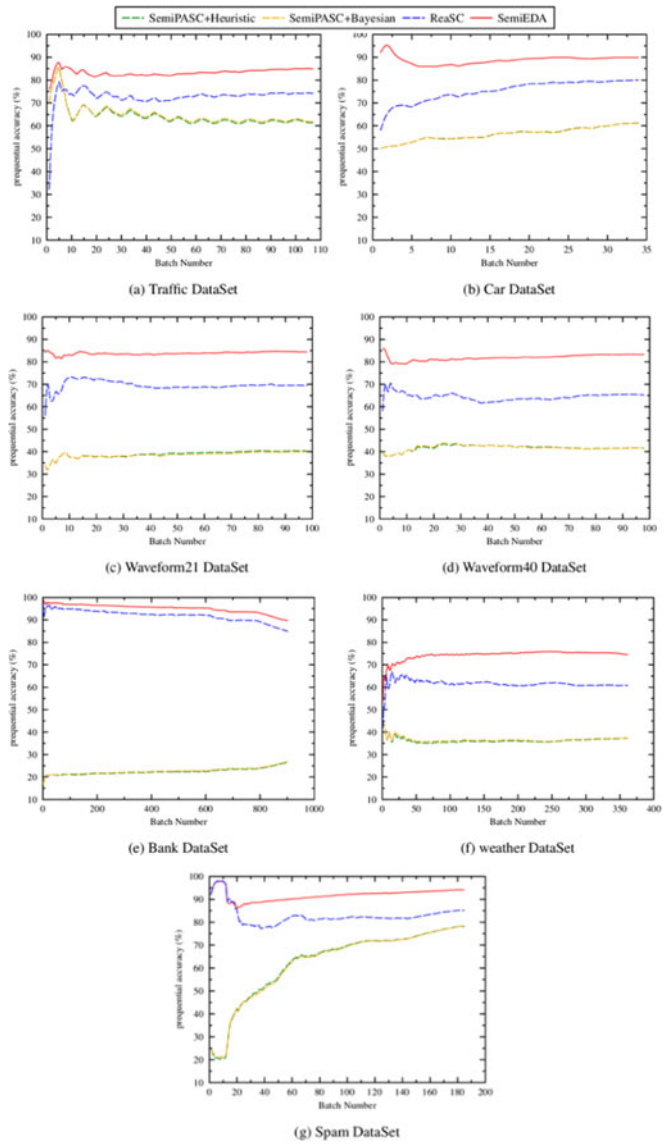

Fig. 5. Prequential accuracy of the semi-supervised methods on (a) Traffic, (b) Car, (c) Waveform21, (d) Waveform40, (e) Bank, (f) Weather, and (g) Spam datasets.

Supervised version. In this version, similarly we use the first 50 data instances as the source domain data whose labels are available and other data are considered as the target data whose true labels are revealed only after predicting their labels by the system. The performance of our Supervised EDA (SEDA) method is compared with CCP [21], PASC [23] and extended PASC method [22] which are the most successful methods proposed for stream

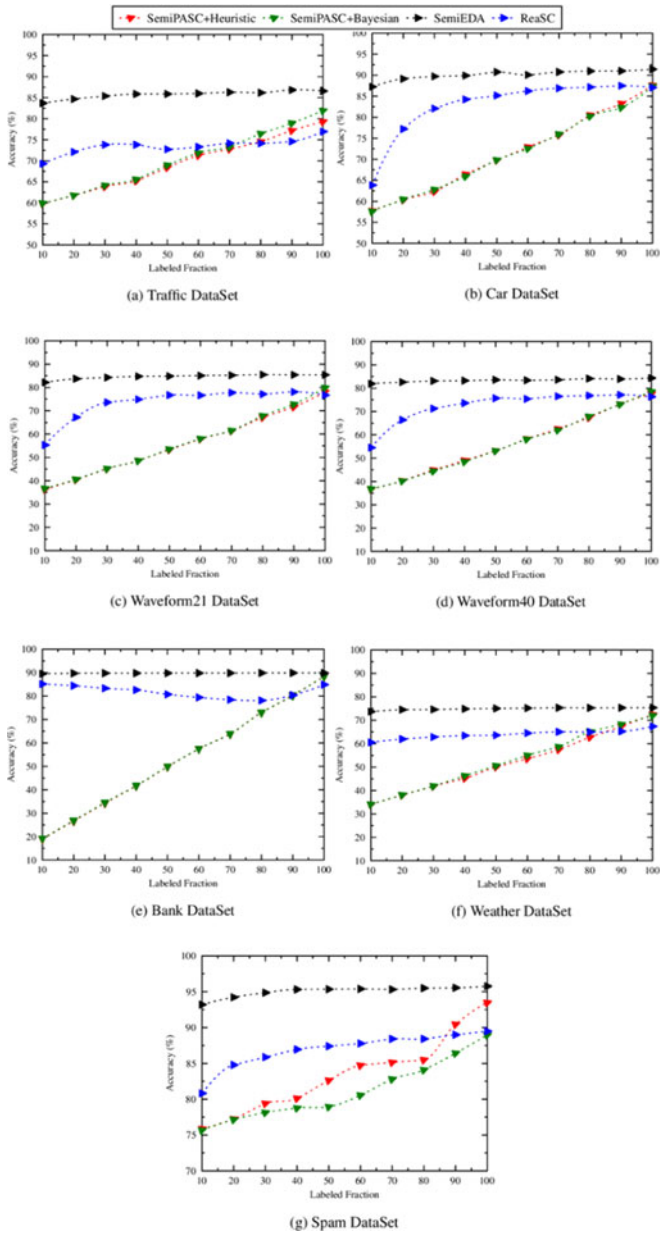

Fig. 6. Accuracy of different semi-supervised methods for different percentage of labeled data on (a) Traffic, (b) Car, (c) Waveform21, (d) Waveform40, (e) Bank, (f) Weather, and (g) Spam datasets.

TABLE 3
Classification Accuracy (%) of the Supervised Version

Method	Classifier	Traffic	Car	Waveform 21	Waveform 40	Bank	Weather	Spam
CCP	Active	73.36	90.23	81.01	79.81	78.68	72.73	91.93
	Weighted	79.48	87.44	80.58	77.09	86.51	71.96	90.61
PASC+ Heuristic	Active	76.33	89.65	81.37	79.57	85.72	71.82	89.29
	Weighted	78.86	90.11	81.37	79.51	86.28	72.32	90.26
PASC+ Bayesian	Active	76.29	90.34	81.09	79.67	87.16	68.92	89.59
	Weighted	80.02	88.72	80.28	78.36	86.66	71.64	89.67
Extended PASC	Active	77.32	90.41	80.86	79.67	82.45	73.72	90.24
	Weighted	80.32	88.25	80.62	78.20	86.46	72.45	91.20
NPSC		81.47 ± 0.55	89.16 ± 0.11	83.78 ± 0.49	82.30 ± 0.21	88.89 ± 0.08	**75.39 ± 0.81**	94.36 ± 0.83
SEDA		**86.03**	**91.10**	**85.39**	**84.40**	**89.86**	**75.39**	**95.69**

classification, and NPSC [20] which is the state-of-the-art method proposed in the field of supervised stream classification with the concept drift. Since NPSC method uses from the Gibbs sampling technique for inference, we run it 10 times and compute the average and standard deviation of its accuracy. PASC proposes two supervised methods: Bayesian and Heuristic, and uses two classification approaches: active classifier and weighted classifiers. We compare our method with all of these methods. Results of our experiments are presented in Table 3 (in which the bolded accuracies show the best accuracies and the underlined ones are the second best accuracies on the corresponding datasets).

As it can be seen in Table 3, supervised EDA outperforms all the other methods and it can better handle concept drift since it observes all the data and forgets older ones gradually. Ensemble method NPSC perform better than PASC and CCP, because it doesn't assume that all the data in a batch belong to the same concept and can handle multiple concepts. Fig. 7 shows the prequential accuracy in order to evaluate the accuracy over time on all the datasets.

A disadvantage of the CCP, PASC, and NPSC methods (in classifying test data) is that they need to access enough data to update their models. An important advantage of our method is the ability to classify each test example with highest accuracy, even if the size of the batch is small.

Results show that although EDA is a single model, all supervised, semi-supervised and unsupervised versions of it provide better results than the other methods on most of the datasets.

4.4 Sensitivity Analysis of Parameters

In this section, we intend to analyze the effect of the values of the parameters used in our EDA method such as γ_A, γ_B, σ, d, T, the size of the batches, and the number of eigenvectors. We show the results of sensitivity analysis of the parameters used in the supervised version of our EDA method (i.e., SEDA) on the spam dataset. The effect of T showing forgetting time, γ_A and γ_B that are regularization parameters, and σ that is used to compute the similarity matrix are shown in Fig. 8.

It could be inferred from the above figures that changes of the parameter values, don't lead to major variations in performance. As it can be seen, when the forgetting factor becomes very small, the accuracy will be decreased since the effect of some useful (but older) data has been decreased.

For demonstrating the effect of the number of subspace dimensions, we change d parameter from 20 to 140 with the ascending steps of 20 and find the accuracy of SEDA method that is shown in Fig. 9.

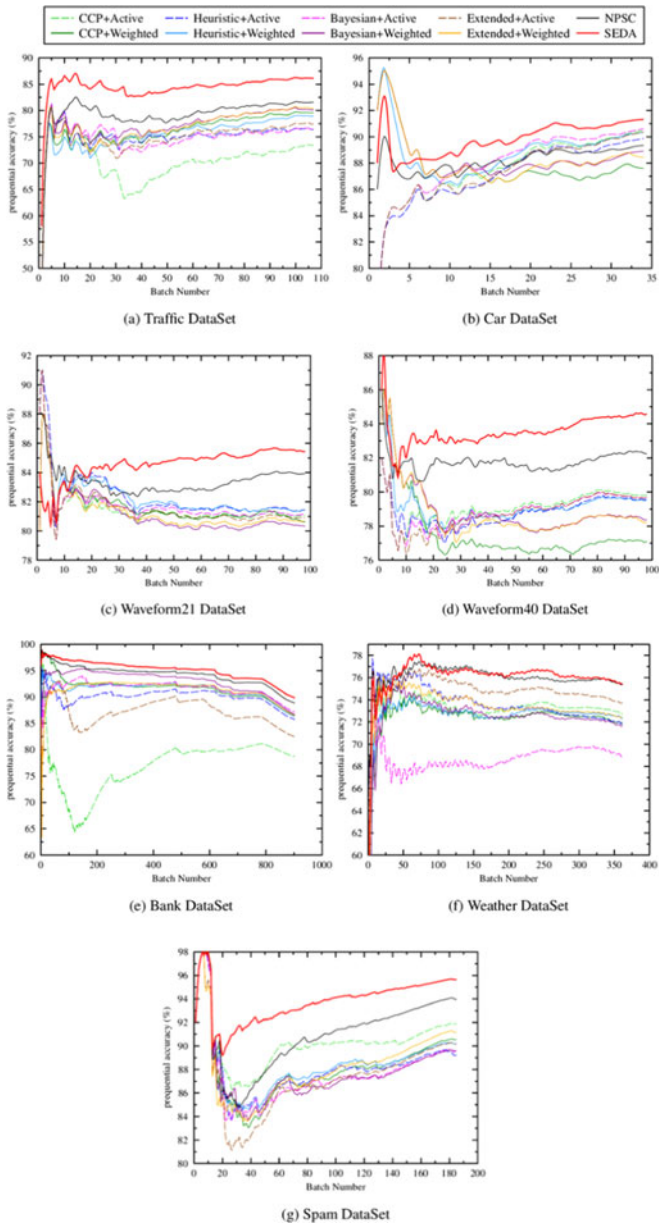

Fig. 7. Prequential accuracy of supervised methods on (a) Traffic, (b) Car, (c) Waveform21, (d) Waveform40, (e) Bank, (f) Weather, and (g) Spam datasets.

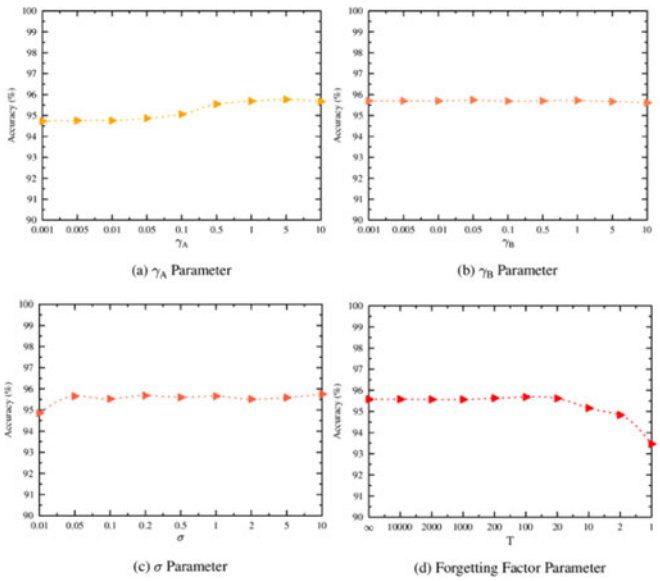

Fig. 8. Effect of the parameters such as (a) γ_A, (b) γ_B, (c) σ, and (d) T on spam dataset in supervised EDA method.

As it can be seen, changes of subspace dimensions, don't lead to major variations in performance. For demonstrating the effect of the size of batches, we tested Supervised EDA method on fifteen different sizes of batches (like 2, 5, 10, 20, 30, 50, 60, 80, 100, 150, 200, 250, 300, 400 and 500) that are shown in Fig. 10. Since this parameter is common between NPSC and SEDA methods, we show the sensitivity to the value of this parameter for both of these methods. As it can be seen, SEDA method outperforms NPSC method for all of the batch sizes. Another evidence that can be obtained from this experiment is that when the size of batches increases, SEDA shows a decline in accuracy. However, NPSC method shows a more intensive decrease.

As shown in Fig. 10 the accuracy of Supervised EDA method changes when the number of eigenvectors is increases. For the spam dataset, when the number of eigenvectors increases the accuracy will be improved. Generally, for datasets with large number of data, the performance of our method improves when the number of eigenvectors increases. However, we have use a fixed small number of eigenvectors (i.e., 100 as mentioned in Section 4.2) for all the

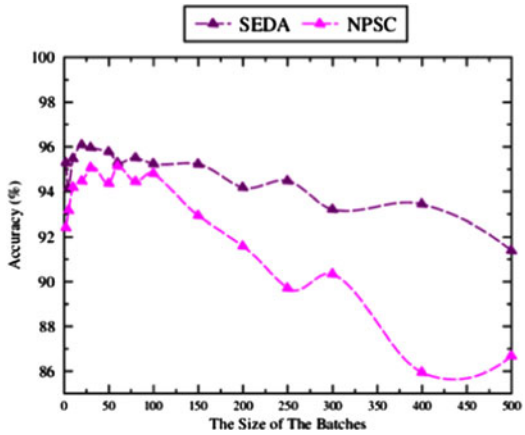

Fig. 10. Effect of the size of batches on the spam dataset.

datasets to avoid high computational complexity in the above experiments.

For demonstrating the effect of the number of eigenvectors, we tested SEDA method for twenty different numbers of eigenvectors that are shown in Fig. 11.

5 Conclusion

In this paper, we addressed the problem of adaptation to continuously evolving target domains and proposed the Evolving Domain Adaptation method for classification of evolving data. The proposed method first incrementally learns a time-varying kernel and then used a semi-supervised classification method to utilize both the unlabeled data of the target domain and the labeled data of the source domain for classifying current test data. Indeed, EDA uses the property of continuously evolving data distributions to find a structure to propagate labels via it and the labeling function is enforced to be sufficiently smooth with respect to the intrinsic structure revealed by all arriving data that include both labeled and unlabeled data. The proposed EDA method is an incremental method and its supervised and semi-supervised versions could also be used to classify stream data in non-stationary environments. We experimentally show that our method outperforms other recent methods of the three unsupervised, semi-supervised, and supervised scenarios on several datasets which are dynamic over time.

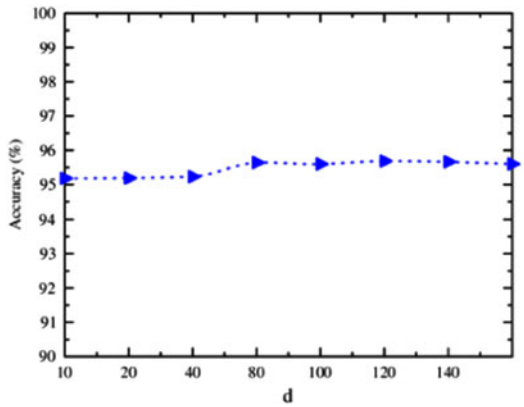

Fig. 9. Effect of the number of the subspace dimensions on the spam dataset for the supervised EDA method.

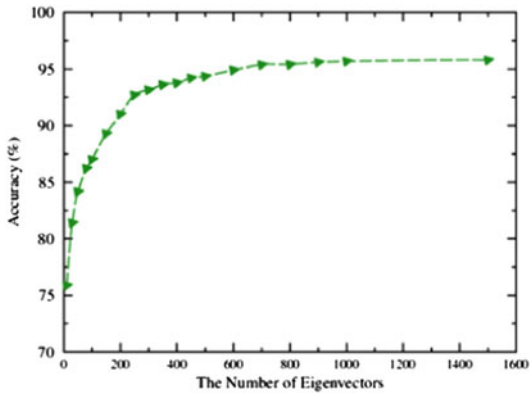

Fig. 11. Effect of the number of eigenvectors on the spam dataset for the supervised EDA method.

REFERENCES

[1] J. Hoffman, T. Darrell, and K. Saenko, "Continuous manifold based adaptation for evolving visual domains," in *Proc. 27th IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 867–874.

[2] L. Duan, D. Xu, I. W. Tsang, and J. Luo, "Visual event recognition in videos by learning from web data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1667–1680, Sep. 2012.

[3] L. Duan, D. Xu, and S. Chang, "Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach," in *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, 2012, pp. 1338–1345.

[4] L. Duan, I. Tsang, and D. Xu, "Domain transfer multiple kernel learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 465–479, Mar. 2012.

[5] L. Duan, I. W. Tsang, D. Xu, and T.-S. Chua, "Domain adaptation from multiple sources via auxiliary classifiers," in *Proc. 26 th Int. Conf. Mach. Learn.*, 2009, pp. 289–296.

[6] M. Kan, J. Wu, S. Shan, and X. Chen, "Domain adaptation for face recognition: Targetize source domain bridged by common subspace," *Int. J. Comput. Vision*, vol. 109, no. 1, pp. 94–109, Aug. 2014.

[7] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. 25th IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 2066–2073.

[8] R. Gopalan, R. Li, and R. Chellappa, "Unsupervised adaptation across domain shifts by generating intermediate data representations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2288–2302, Nov. 2014.

[9] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *Proc. 12th IEEE Int. Conf. Comput. Vis.*, 2011, pp. 999–1006.

[10] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Proc. 14th IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2960–2967.

[11] M. Gheisari and M. S. Baghshah, "Unsupervised domain adaptation via representation learning and adaptive classifier learning," *Neurocomputing*, vol. 165, pp. 300–311, Oct. 2015.

[12] J. Hoffman, E. Rodner, J. Donahue, K. Saenko, and T. Darrell, "Efficient learning of domain-invariant image respresentations," in *Proc. Int. Conf. Learning Representations*, 2013.

[13] S. Qian, et al., "Transfer learning for bilingual content classification," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 2147–2156.

[14] J. Xu, S. Ramos, D. Vázquez, and A. M. López, "Incremental domain adaptation of deformable part-based models," in *Proc. 25th Brit. Mach. Vis. Conf.*, 2014.

[15] V. Jain and E. Learned-Miller, "Online domain adaptation of a pre-trained cascade of classifiers," in *Proc. 24th IEEE Conf. Comput. Vision Pattern Recog.*, 2011, pp. 577–584.

[16] C. H. Lampert, "Predicting the future behavior of a time-varying probability distribution," in *Proc. 28th IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 942–950.

[17] A. Levey and M. Lindenbaum, "Sequential Karhunen-Loeve basis extraction and its application to images," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1371–1374, Aug. 2002.

[18] W. Fan, "Systematic data selection to mine concept-drifting data streams," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 128-137.

[19] R. Klinkenberg, "Learning drifting concepts: Example selection vs. example weighting," *Intell. Data Anal.*, vol. 8, no. 3, pp. 281–300, Aug. 2004.

[20] S. A. Hosseini, H. R. Rabiee, H. Hafez, and A. Soltani-Farani, "Classifying a stream of infinite concepts: A bayesian nonparametric approach," in *Machine Learning and Knowledge Discovery in Databases*, Ed. Berlin, Germany: Springer, 2014, pp. 1–16.

[21] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: An application to email filtering," *Knowl. Inform. Syst.*, vol. 22, no. 3, pp. 371–391, Mar. 2010.

[22] M. J. Hosseini, Z. Ahmadi, and H. Beigy, "Using a classifier pool in accuracy based tracking of recurring concepts in data stream classification," *Evolving Syst.*, vol. 4, no. 1, pp. 43–60, Mar. 2013.

[23] M. J. Hosseini, Z. Ahmadi, and H. Beigy, "Pool and accuracy based stream classification: A new ensemble algorithm on data stream classification using recurring concepts detection," in *Proc. 11th IEEE Int. Conf. Data Mining Workshops*, 2011, pp. 588–595.

[24] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2283–2301, Oct. 2013.

[25] I. Zliobaite and B. Gabrys, "Adaptive preprocessing for streaming data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 309–321, Feb. 2014.

[26] M. J. Hosseini, A. Gholipour, and H. Beigy, "An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams," *Knowl. Inform. Syst.*, vol. 26, no. 3, pp. 1–31, Apr. 2015.

[27] M. M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, K. W. Hamlen, et al., "Facing the reality of data stream classification: Coping with scarcity of labeled data," *Knowl. Inform. Syst.*, vol. 33, no. 1, pp. 213–244, Oct. 2012.

[28] M. Dredze and K. Crammer, "Online methods for multi-domain learning and adaptation," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2008, pp. 689–697.

[29] I. S. Helland, "On the structure of partial least squares regression," *Commun. Statistics-Simul. Comput.*, vol. 17, no. 22, pp. 581–607, 1988.

[30] S. De Jong, "SIMPLS: An alternative approach to partial least squares regression," *Chemometrics Intell. Laboratory Syst.*, vol. 18, no. 3, pp. 251–263, Mar. 1993.

[31] X.-Q. Zeng and G.-Z. Li, "Incremental partial least squares analysis of big streaming data," *Pattern Recog.*, vol. 47, no. 11, pp. 3726–3735, Nov. 2014.

[32] R. Fergus, Y. Weiss, and A. Torralba, "Semi-supervised learning in gigantic image collections," in *Proc. 12th Neural Inf. Process. Syst.*, 2009, pp. 522–530.

[33] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learning Res.*, vol. 7, pp. 2399–2434, Nov. 2006.

[34] O. Chapelle, B. Schlkopf, and A. Zien, *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.

[35] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT press, 2002.

[36] M. Belkin and P. Niyogi, "Towards a theoretical foundation for Laplacian-based manifold methods," *J. Comput. Syst. Sci.*, vol. 74, no. 8, pp. 1289–1308, Dec. 2008.

[37] E. Mantziou, S. Papadopoulos, and Y. Kompatsiaris, "Scalable training with approximate incremental laplacian eigenmaps and PCA," in *Proc. 21st ACM Int. Conf. Multimedia*, 2013, pp. 381–384.

[38] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, Oct. 2011.

[39] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, et al., "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. 29th Int. Conf. Mach. Learn.*, 2014, pp. 647–655.

[40] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learning*, vol. 90, no. 3, pp. 317–346, Mar. 2013.

Adeleh Bitarafan received the BSc degree in computer engineering from Qom University, Iran, in 2013 and the MSc degree from the Computer Engineering Department, Sharif University of Technology, Tehran, Iran, in 2015. Her research focuses on applications of machine learning algorithms to computer vision problems. She also interests to work on online learning, domain adaptation, and streaming data.

Mahdieh Soleymani Baghshah received the BS, MSc, and PhD degrees from the Department of Computer Engineering, Sharif University of Technology, Iran, in 2003, 2005, and 2010. She is an assistant professor in the Computer Engineering Department, Sharif University of Technology, Tehran, Iran. Her research interests include machine learning and pattern recognition.

Marzieh Gheisari received the BS degree from the Department of Electrical and Computer Engineering, University of Kashan, Iran, in 2012, and the MSc degree from the Computer Engineering Department, Sharif University of Technology, Iran, in 2014. Her research interests are in machine learning, statistics, and optimization.