Pattern Recognition 88 (2019) 383–396

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Incremental semi-supervised learning on streaming data

Yanchao Li[a], Yongli Wang[a,*], Qi Liu[c], Cheng Bi[b], Xiaohui Jiang[a], Shurong Sun[d]

[a] *School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China*
[b] *School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, UK*
[c] *School of Computing, Edinburgh Napier University, Edinburgh EH10 5DT, UK*
[d] *Zhenjiang Analysis InfoTech Ltd, Zhenjiang 212100, China*

ARTICLE INFO

Article history:
Received 19 February 2018
Revised 27 September 2018
Accepted 10 November 2018
Available online 16 November 2018

Keywords:
Semi-supervised learning
Dynamic feature learning
Streaming data
Classification

ABSTRACT

In streaming data classification, most of the existing methods assume that all arrived evolving data are completely labeled. One challenge is that some applications where only small amount of labeled examples are available for training. Incremental semi-supervised learning algorithms have been proposed for regularizing neural networks by incorporating various side information, such as pairwise constraints or user-provided labels. However, it is hard to put them into practice, especially for non-stationary environments due to the effectiveness and parameter sensitivity of such algorithms. In this paper, we propose a novel incremental semi-supervised learning framework on streaming data. Each layer of model is comprised of a generative network, a discriminant structure and the bridge. The generative network uses dynamic feature learning based on autoencoders to learn generative features from streaming data which has been demonstrated its potential in learning latent feature representations. In addition, the discriminant structure regularizes the network construction via building pairwise similarity and dissimilarity constraints. It is also used for facilitating the parameter learning of the generative network. The network and structure are integrated into a joint learning framework and bridged by enforcing the correlation of their parameters, which balances the flexible incorporation of supervision information and numerical tractability for non-stationary environments as well as explores the intrinsic data structure. Moreover, an efficient algorithm is designed to solve the proposed optimization problem and we also give an ensemble method. Particularly, when multiple layers of model are stacked, the performance is significantly boosted. Finally, to validate the effectiveness of the proposed method, extensive experiments are conducted on synthetic and real-life datasets. The experimental results demonstrate that the performance of the proposed algorithms is superior to some state-of-the-art approaches.

1. Introduction

Web-scale data has explosively created in the social media. Learning and analyzing from streaming data has attracted increasing interest in both academia and industry, such as analysis of long-term financial fraud detection, weather changes, or buyer habits. Practitioners have proposed several methods to deal with streaming data, for example, modifying model into a non-stationary environment [1–3], incremental learning [4–7] and online learning [5,7–9]. However, most of the existing classification algorithms on streaming data assume that all arrived evolving examples are completely labeled [1,7,10]. In some applications, where data appear at a high speed, it is not always possible to manually label all the examples as soon as they arrive. To be specific, the trained examples may available at time-stamp t, but labeling of examples need to be evaluated and may be available at time-stamp $t+1$. If we only wait for the future labels passively, it is likely that much potentially useful information is lost.

Streaming data classification is a challenging problem because of its incremental and concept drift nature. An important problem that is faced in data mining or machine learning process is continuously evolving new data. Traditional supervised or semi-supervised learning methods allow us to learn with the help of training data one time. Facing the evolving new data, they must retrain the models, which is big time overhead for large-scale datasets and cannot adapt into the non-stationary environments. Besides, changes in the data distribution over time are commonly referred to as concept drift, which is a challenging problem in a variety of streaming data. To cope with these, practitioners have been studying incremental learning [4–6,11]. The key idea of that is

* Corresponding author.
E-mail addresses: yongliwang@njust.edu.cn, yongliwang@mail.njust.edu.cn (Y. Wang).

https://doi.org/10.1016/j.patcog.2018.11.006
0031-3203/© 2018 Elsevier Ltd. All rights reserved.

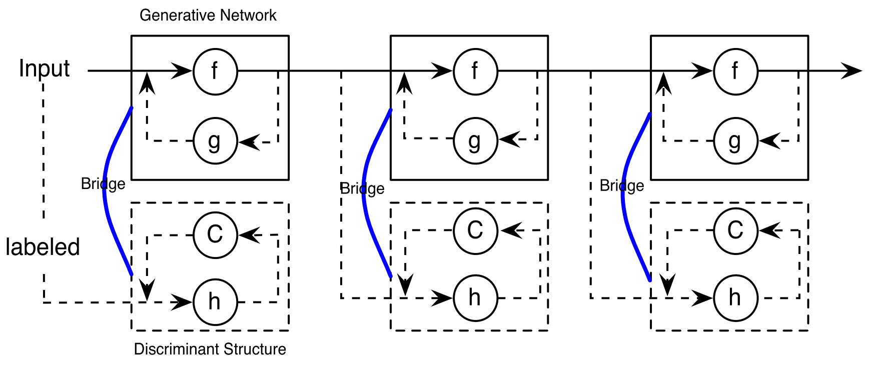


Fig. 1. Each layer of model consists of three components: generative network, discriminative structure and the bridge. After training, the model is used to produce very compact representations by a feed-forward pass through the chain of encoders *f* for deep architecture (*f*, *h* are two encoders, *g* is a decoder, *C* is a discernibility function and the feedback decoding modules *g* and the encoder modules *h* with the corresponding classifier modules are discarded).

to address the problem of learning the knowledge while maintaining the previous one. Traditional incremental supervised learning methods have been proposed, such as Learn++ [11] and incremental SVM [6]. In semi-supervised learning setting, subspace learning [12], bayesian learning [13], semi-supervised clustering and classification [14–17] are modified for incremental learning. However, existing semi-supervised learning (SSL) methods are mostly based on clustering or manifold assumptions [18,19]. According to [19], they found that the performance degeneration of SSL is caused by incorrect model assumptions, because fitting unlabeled data based on an incorrect model assumption will mislead the learning process. It is also very difficult to make a correct model assumption without sufficient domain knowledge. Moreover, other methods may generate pseudo-labels of unlabeled data by learners during the training process, but, incorrect pseudo-labels may disrupt the learning process. Besides, the training process may even cause deteriorated performance if there are some noise in the streaming data. More recently, many regularizing neural networks, such as autoencoders and restricted boltzmann machines (RBMs) have incorporated various side information, i.e., pairwise constraints or user-provided labels [20,21]. In addition, stacking multi-layers of neural networks e.g., convolutional neural networks (CNN) or denoising autoencoders that can be efficiently pre-trained in layer-wise manner, followed by supervised back-propagation in order to fine-tune the parameters [22–24]. However, it is hard to put them into practice, especially for non-stationary environments due to the effectiveness and parameter sensitivity of such algorithms.

To address these issues, we propose a model referred as “ISLSD” (Incremental Semi-supervised Learning on Streaming Data). Each layer of model is comprised of a generative network, a discriminant structure and the bridge that connects them. To be specific, the generative network uses dynamic feature learning based on autoencoders to learn generative features from streaming data, and the discriminant structure regularizes the network construction via building semantic similarity and dissimilarity constraints between data pairs. It is also used for facilitating the parameter learning of the generative network and will be removed when handling out-of-sample data. The network and structure are bridged by enforcing the correlation of their parameters, which balances the flexible incorporation of supervision information and numerical tractability for non-stationary environments as well as explores the intrinsic data structure. The architecture is shown in Fig. 1. The autoencoders [25] are neural network architectures, which have been demonstrated its potential in learning latent feature representations from training examples. They have gained numerous successes in the fields such as computer vision, speech recognition and natural language processing. However, most of these approaches are limited in that the feature learning has to be fixed during the training process. In order to adapt to the non-stationary environments, we use dynamic feature learning based on denoising autoencoder [5] to learn more robust representations from streaming data, which could avoid underfit (the lack of relevant features) and overfitting (too more features) of models. Meanwhile, it could save time overhead and computational resources of cross-validation process. The generative network and discriminant structure are complementary to each other. Compared with the standard autoencoders, stacking our networks could transmit both labeled and unlabeled data to the next layer. The benefits of dynamic feature learning based on autoencoder is robust to noise data that might lead to the error-prone classification.

The motivation of our method is that deep learning has been demonstrated its potential in learning latent feature representations. Recent years have witnessed an increasing interest for regularizing deep neural networks by incorporating various side information, such as pairwise constraints or user-provided labels. However, existing methods for incorporating side information into neural networks are not adapted into non-stationary environments. For example, the two-step procedure, that is unsupervised pre-training followed by supervised fine-tuning, cannot effectively handle sparse side information, such as similarity or dissimilarity constraints on limited labeled data pairs. Conversely, optimizing parameters of unsupervised or supervised separately tend to converge to non-optimal results. Moreover, incremental semi-supervised learning on streaming data motivates us to efficiently use small amount of labeled examples via adding label-specific output besides the data reconstruction and using recursive structure. Traditional SSL methods leverage unlabeled examples by making assumptions, using label propagation or generating pseudo-labels during learning process. In these methods, incorrect assumptions or pseudo-labels may disrupt the learning process. Besides, the training process may even cause deteriorated performance if there are some noise in the streaming data.

Our method uses dynamic feature learning based on autoencoders, which is robust to noise data. In addition, Our model utilizes both autoencoders and hashing learning models to support real-time computing and applications. In the stage of parameter learning, these two components are simultaneously optimized. Meanwhile, it could save time overhead, which is important for real-time computing. Moreover, we apply our methods into the Youtube-Objects dataset for evaluation. To the best of our knowledge, little work has done this on streaming data. The contribution of this paper is summarized as follows.

- We propose a novel incremental semi-supervised learning model that each layer consists of a generative network, a discriminant structure and the bridge. The generative network is regularized by discriminant structure that provides a sparsely-supervised guidance via pairwise similarity or dissimilarity constraints among data. Stacking networks balance numerical tractability and the flexible utilization of supervision information as well as explores the intrinsic data structure.

- The proposed method is a general one that can leverage several existing methods as special cases into model. We give the ensemble method of our proposed algorithm. Moreover, the optimization method is effective and converges quickly.
- Compared with standard autoencoders, it could transmit both labeled and unlabeled data to the next layer. The experimental study on synthetic and real-life datasets demonstrates that the proposed methods are effective and efficient, and achieve state-of-the-art results against current models.

The rest of this paper is organized as follows. Section 2 introduces some related works. In Section 3, we present model of our method and give its specified algorithm as well as ensemble and extension methods. Also, we analyze the advantages and disadvantages of our method. The extensive experimental results are provided and analyzed in Section 4. Finally, Section 5 concludes this work with future direction.

2. Related work

In this section, we briefly review the related researches on semi-supervised learning and incremental learning on streaming data.

2.1. Semi-supervised learning

Semi-supervised Learning (SSL) [18,19] aims to make use of unlabeled data for training—typically a small set of labeled data together with a large collection of unlabeled data. A large number of semi-supervised learning algorithms jointly optimize two training objective functions: the supervised loss over labeled data and the unsupervised loss over both labeled and unlabeled data such as graph-based SSL algorithms [26,27]. SSL propagates their limited label information to unlabeled examples and it mainly follows the clustering or manifold assumption. The clustering assumption methods [28,29] assume that the examples of different classes from several well-separated clusters, and the decision boundary falls into the low density area in the feature space. Most manifold-based methods [19,30] assume that there is a low-dimensional manifold structure embedded in the data space. Moreover, the other works have incorporated pairwise constraints into SSL, such as Gong and co-workers [31,32] incorporate both similarity and dissimilarity between examples via pairwise constraints. Semi-supervised Hashing (SSH) [33] builds semantic similarity and dissimilarity constraints between data pairs and formulates as minimizing empirical error on the labeled data while maximizing variance and independence of hash bits over the labeled and unlabeled data. Co-training and its variations with the main idea of they train the classifiers and let them label the unlabeled examples for each other during the learning process, such representative works like tri-training [34] and co-labeling [35]. Recently, a large number of deep leaning via semi-supervised embedding were proposed for representation learning, such as generative models [36], ladder networks [37], Π model [38] and *Mean Teacher* [39]. However, the effectiveness and parameter sensitivity of such neural network algorithms have been major obstacles for putting them into non-stationary environments. Meanwhile, the single-model methods incrementally update their model when new data arrive, which involve high complexity for large-scale datasets. Moreover, most of the traditional SSL methods leverage unlabeled examples by making assumptions, using label propagation or generating pseudo-labels during learning process. In these methods, incorrect assumptions or pseudo-labels may disrupt the learning process. Different from the above methods, the discriminant structure of our model provides a sparsely-supervised guidance for regularizing the generative network. Also, it is used for facilitating the parameter learning of the generative network. Compared with the standard autoencoders, stacking our networks could transmit both labeled and unlabeled data to the next layer, which is especially useful for tasks that have limited labeled data.

2.2. Incremental learning on streaming data

In order to apply into complex real scenarios, incremental learning algorithms have been developed [40,41]. In particular, the influential algorithms have an incremental learning version [5,6,41]. Here, incremental learning refers to the situation of continuous model adaptation based on a constantly arriving data. It comes in various forms in the literatures, and the use of this term is not always consistent; for some settings, as an example, a memory limitation cannot be guaranteed, or models are designed for stationary distributions only. Incremental learning techniques have been developed for alternative tasks such as clustering [42], feature selection and data representation [43], reinforcement learning [44], and inference [45,46]. In addition, there are a lot of incremental models designed for dealing with concept drift of streaming data, such as clustering streaming data [10]; data stream classification for decision trees [45] and learning rule sets from streaming data [1]. Most of these algorithms on streaming data assume that all arrived evolving data are completely labeled. However, the labeled resources are limited in the non-stationary environments. Thus, the incremental semi-supervised learning methods have been proposed, for example, subspace learning [12], bayesian learning [13], discriminant analysis [47], semi-supervised clustering [14,15,48,49] and classification [8,16,17,27,50] are modified for incremental learning. In addition, deep learning for incremental semi-supervised learning has been proposed [51–53]. Moreover, there are some incremental SSL algorithms used in infectious disease prediction [54], object detection from videos [55] and weather prediction [56]. However, those methods incorporated traditional SSL algorithms may cause massive accuracy degeneration in the non-stationary environments because incorrect assumptions or generating pseudo-labels from SSL algorithms may disrupt the learning process. Besides, using classification or clustering techniques during the training process might lead to error-prone classification if there are some noise examples in the streaming data. Moreover, the two-step procedure of neural networks, that is unsupervised pre-training followed by supervised fine-tuning, cannot effectively handle sparse side information, such as similarity or dissimilarity constraints on limited labeled data pairs. In addition, the methods of separately performing unsupervised or supervised parameters optimization are easy to converge the non-optimal results. The generative network of our method is regularized by discriminant structure that provides a sparsely-supervised guidance and facilitates the parameter learning. Stacking our networks balance numerical tractability and the flexible utilization of supervision information as well as explores the intrinsic data structure. To the best of knowledge, little work has done this on streaming data.

3. Our approach

We now describe the details of our approach. First, we give the model of our approach. Then, we introduce the components of our model that are dynamic feature learning based on autoencoders, semi-supervised hashing and the bridge. Finally, we propose a specific algorithm to deal with limited labeled examples in the non-stationary environments and give its ensemble as well as extension methods. Also, we analyze the advantages and disadvantages of our approach.

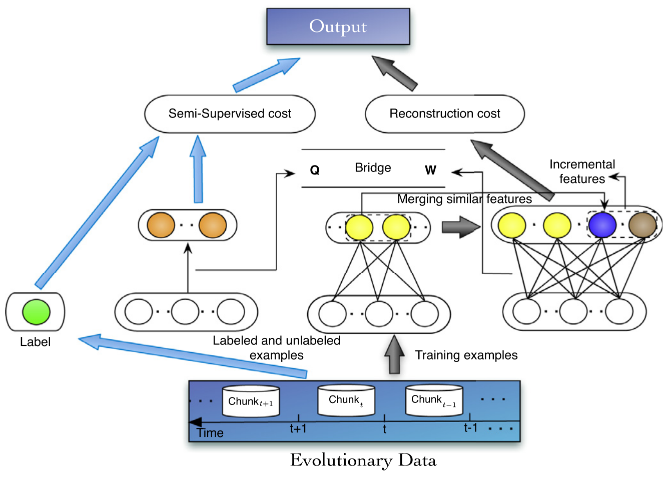


Fig. 2. The workflow of our approach (the right part uses dynamic features based on autoencoders to learn generative features; The brown units are incremental features, and the blue units are merged feature from the similar existing features; The left part uses SSH to learn discriminant features; The bridge builds a connection between the generative and discriminant features). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.1. Model

In this section, we briefly introduce three main components of our proposed model,

- **Generative network** is used to learn representations from the input, i.e., autoencoders;
- **Discriminant structure** is used to regularize the generative network by building the pairwise similarity or dissimilarity constraints, i.e., semi-supervised hashing;
- **The bridge** is used to connect generative network and discriminant structure by enforcing the correlation of their parameters.

For semi-supervised learning in the non-stationary environments, we address the labeled and unlabeled examples with streaming data. The generative network learns the feature representations from denoising autoencoder and is regularized by semi-supervised hashing. Thus, these two components are integrated and bridged by enforcing the correlation of their parameters. Those two components are complementary to each other. Finally, we output the classification result of the mixed cost that is the reconstruction and hashing cost. The workflow of our approach is illustrated in Fig. 2.

3.2. Dynamic feature learning based on autoencoder

In this section, we formally specify dynamic feature learning based on autoencoder. Given training examples $\{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_l \dots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in R^D$ and there are l labeled training examples. The classical autoencoder is an unsupervised learning algorithm that applies back propagation, setting the target values to be equal to the inputs. The key idea can be described as two phases: (1) an encoder phase: the input $\mathbf{x}$ is encoded into a lower dimensional representation $\mathbf{h} = f(\mathbf{x})$. (2) a decoder phase: reconstructing the input again as $\hat{\mathbf{x}} = g(\mathbf{h})$ from the representation form (1). The encoding, decoding and the cost function can be expressed as follows:

$$\begin{aligned}\mathbf{h} &= f(\mathbf{W}\mathbf{x} + \mathbf{b}),\\ \hat{\mathbf{x}} &= g(\mathbf{W}^T\mathbf{h} + \mathbf{c}),\end{aligned} \tag{1}$$

$$\begin{aligned}L(\mathbf{x}) &= \ell(\mathbf{x}, \hat{\mathbf{x}}) = -\left(\sum_{i=1}^{D} \mathbf{x}_i \log \hat{\mathbf{x}}_i + (1 - \mathbf{x}_i)\log(1 - \hat{\mathbf{x}}_i)\right),\\ L(\mathbf{x}) &= \ell(\mathbf{x}, \hat{\mathbf{x}}) = \| \mathbf{x} - \hat{\mathbf{x}} \|^2,\end{aligned} \tag{2}$$

where $\mathbf{W} \in R^{N \times D}$ is a weight matrix, $\mathbf{b} \in R^N$ is a hidden bias vector, $\mathbf{c} \in R^D$ is an input bias vector, and when active function of $g(\cdot)$ is sigmoid function: $sigmoid(z) = 1/\{1 + exp(-z)\}$, then the cross-entropy is as a cost function. Otherwise, the squared error is as the cost function when $g(\cdot)$ is the identity function. Since we consider the input and hidden variables are bounded between 0 and 1. If the input values are in range of 0 to 255, we normalize to the range 0 to 1 by dividing each value of the maximum observation. Thus, $\mathbf{x} \in [0, 1]^D$ and $\mathbf{h} \in [0, 1]^K$. For denoising autoencoders (DAE) [25], it takes a partially corrupted input whilst training to recover the original undistorted input. In our practice, we randomly set some of the coordinates to zeros. The conditional distribution $q(\tilde{\mathbf{x}}|\mathbf{x})$, that is over corrupted versions $\tilde{\mathbf{x}}$ of the input $\mathbf{x}$, which is $\ell_{DAE}(\cdot) = \sum_{\mathbf{x}\in\mathbf{X}} E_{\tilde{\mathbf{x}}\sim q(\tilde{\mathbf{x}}|\mathbf{x})[\ell(\mathbf{x},\hat{\mathbf{x}})]}$.

In order to adapt to the non-stationary environments, one uses dynamic feature learning method [5] to address online data. In practice, the dynamic feature learning methods include two phases: (1) adding new features into the existing feature set, this phase can reduce the bias of the model; (2) merging parts of feature sets to reduce variance of model. Moreover, in order to describe the process of adding features and merge similar existing features, one defines some other parameters: The weight matrix $\boldsymbol{\Theta} \in R^{S \times N}$ between hidden and output units, the output bias $\boldsymbol{\epsilon} \in R^S$, N for the number of (existing) features, and S for the number of class labels. One uses $\boldsymbol{\theta}$ to denote all parameters $\{\mathbf{W}, \mathbf{b}, \mathbf{c}, \boldsymbol{\Theta}, \boldsymbol{\epsilon}\}$. For incremental feature learning, one uses $\mathcal{O}$ and $\mathcal{N}$ to denote the old (existing) and new features respectively. Thus, the encoding function with old features and newly added features are represented as $f_{\mathcal{O}}(\tilde{\mathbf{x}}) = \mathbf{h}_{\mathcal{O}} \in [0, 1]^N$ and $f_{\mathcal{N}}(\tilde{\mathbf{x}}) = \mathbf{h}_{\mathcal{N}} \in [0, 1]^{\Delta N}$. A combined encoding function is represented as $f_{\mathcal{O}\cup\mathcal{N}}(\tilde{\mathbf{x}}) = [\mathbf{h}_{\mathcal{O}}; \mathbf{h}_{\mathcal{N}}] \in [0, 1]^{N+\Delta N}$. Similarly, the parameters of existing and new features are represented as $\{\mathbf{W}_{\mathcal{O}}, \mathbf{b}_{\mathcal{O}}, \mathbf{c}, \boldsymbol{\Theta}_{\mathcal{O}}, \boldsymbol{\epsilon}\}$ and $\{\mathbf{W}_{\mathcal{N}}, \mathbf{b}_{\mathcal{N}}, \mathbf{c}, \boldsymbol{\Theta}_{\mathcal{N}}, \boldsymbol{\epsilon}\}$ respectively. Therefore, the encoding, decoding and optimization functions for the new features can be written as the following:

$$\begin{aligned}\mathbf{h}_{\mathcal{N}} &= f(\mathbf{W}_{\mathcal{N}}\tilde{\mathbf{x}} + \mathbf{b}_{\mathcal{N}}),\\ \hat{\mathbf{x}} &= g(\mathbf{W}_{\mathcal{N}}^T\mathbf{h}_{\mathcal{N}} + \mathbf{W}_{\mathcal{O}}^T\mathbf{h}_{\mathcal{O}} + \mathbf{c})\\ &= g(\mathbf{W}_{\mathcal{N}}^T\mathbf{h}_{\mathcal{N}} + c_d(\mathbf{h}_{\mathcal{O}})),\end{aligned} \tag{3}$$

$$\hat{\mathbf{y}} = softmax(\boldsymbol{\epsilon} + \boldsymbol{\Theta}_{\mathcal{O}} f_{\mathcal{O}}(\tilde{\mathbf{x}}) + \boldsymbol{\Theta}_{\mathcal{N}} f_{\mathcal{N}}(\tilde{\mathbf{x}})), \tag{4}$$

$$L(\mathbf{x}, \mathbf{y}) = \ell(\mathbf{x}, \hat{\mathbf{y}}(\mathbf{x})) + \gamma\,\ell(\mathbf{x}), \tag{5}$$

where $\mathbf{y}$ is a binary vector with a softmax unit that allows K-way classification problem, $\hat{\mathbf{y}}$ denotes the posterior probability of the labels of softmax function, γ gives a trade-off between discriminative and generative training of autoencoder, $c_d(\mathbf{h}_{\mathcal{O}})$ is viewed as a *dynamic* decoding bias. Since $c_d(\mathbf{h}_{\mathcal{O}})$ is not changing during the training features, we can recall the $c_d(\mathbf{h}_{\mathcal{O}})$ and optimize the new parameters greedily via stochastic gradient descent. The basic idea for efficient training is that only the new features and corresponding parameters $\theta_{\mathcal{N}}$ are trained to minimize the objective function while keeping $\theta_{\mathcal{O}}$ fixed.

Finally, we introduce the **Adding Process**: Choosing a threshold μ to decide whether the training examples should be learned and added into feature set. For example, if objective function value of training examples is greater than μ, we greedily add new features into the feature set. Normally, we set μ as the average of objective

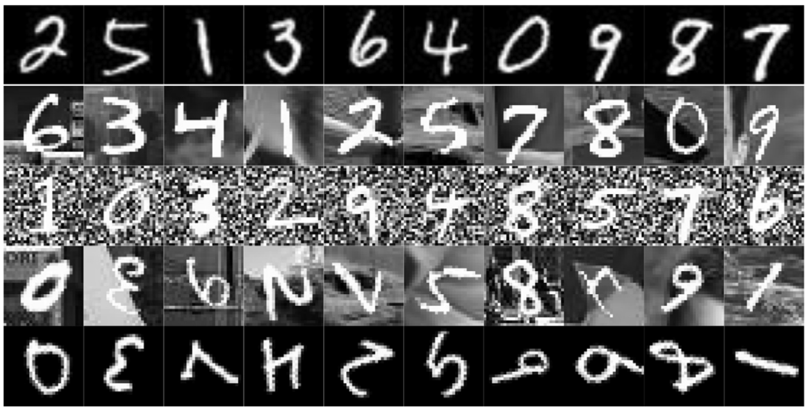


Fig. 3. Samples from MNIST dataset and its variations.

function values for the recent batch examples. The **Merging Process** consists two steps: choosing a pair of candidate features and merging them to a single feature. We use cosine distance to merge candidate features. There are other similarity functions such as KL divergence, and forward and backward search strategy can be utilized to reduce model complexity. The details can be described as the following:

- Choosing evolving data Z to select a pair of features to be merged whose distance is minimal: $\hat{\mathcal{M}} = \arg\min_{m_1,m_2} d(\mathbf{W}_{m_1}, \mathbf{W}_{m_2})$, where $\mathcal{M} = \{m_1, m_2\} \subset \mathcal{O}$, and replace $f_{\mathcal{O}}$ by $f_{\mathcal{O}\setminus\mathcal{M}}$ (remove m_1-th and m_2-th features from $\mathcal{O}$).
- Adding new features to $\theta_{\mathcal{O}\setminus\mathcal{M}}$ by solving $\hat{\theta}_{\mathcal{N}} = \arg\min_{\theta_{\mathcal{N}}} \frac{1}{|Z|}\sum_{i\in Z} L(\mathbf{x}_i, \mathbf{y}_i)$.

3.3. Semi-supervised hashing

Semi-supervised hashing [33] has been used in large scale image retrieval by leveraging similarity-preserving criterion. Hashing aims to map the input data $\mathbf{X} \in R^{D\times n}$ into a K-dimensional Hamming space to obtain its compact representation $\{\pm 1\}^{K\times n}$. The inner product of two hash codes approximately reflect the corresponding data similarity in the original feature space. One lets $\mathbf{X}$ be normalized to the mean of zero for centered data and use linear projection coupled with mean thresholding as a hash function. To be specific, given a vector $\mathbf{q}_k \in R^D$, the k^{th} hash function is defined as:

$$h_k(\mathbf{x}_i) = sgn(\mathbf{q}_k^T \mathbf{x}_i + t_k), \tag{6}$$

where $sgn(\cdot)$ is the signum function, $\mathbf{H} = [h_1, \ldots, h_K]$ is a sequence of K hash functions and the weight matrix $\mathbf{Q} = [\mathbf{q}_1^T, \ldots, \mathbf{q}_k^T] \in R^{K\times D}$ is a projection matrix. For each element $t_k = -\frac{1}{n}\sum_{i=1}^n \mathbf{q}_k^T\mathbf{x}_i$ is equal to zero since $\mathbf{X}$ is zero-mean. One can get the binary bit as $y_{ki} = \frac{1}{2}(1 + h_k(\mathbf{x}_i)) = \frac{1}{2}(1 + sgn(\mathbf{q}_k^T\mathbf{x}_i))$. One also denote a pair $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{B}$, which means $\mathbf{x}_i$ and $\mathbf{x}_j$ are either neighbors in a metric space or shared common class labels. Similarly, a pair $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$ reflects the fact that $\mathbf{x}_i$ and $\mathbf{x}_j$ are far away in a metric space or has different class labels [33]. One wants to learn a $\mathbf{Q}$ that gives the same bits for $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{B}$ and different bits for $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$. In order to measure each similarity or dissimilarity constraints between data pairs, one can define the following objective function:

$$\Psi(\mathbf{H}) = \sum_k \left\{ \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{B}} h_k(\mathbf{x}_i)h_k(\mathbf{x}_j) - \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} h_k(\mathbf{x}_i)h_k(\mathbf{x}_j) \right\}. \tag{7}$$

Such an idea has been well explored in the content of discriminative subspace learning. For simplicity, Eq. (7) is therefore rewritten as the following:

$$\Psi(\mathbf{H}) = \frac{1}{2}\mathbf{tr}\{\mathbf{H}(\mathbf{X}_l)\mathbf{\Phi}\mathbf{H}(\mathbf{X}_l)^T\},$$
$$\Rightarrow \Psi(\mathbf{Q}) = \frac{1}{2}\mathbf{tr}\{sgn(\mathbf{Q}\mathbf{X}_l)\mathbf{\Phi}\ sgn(\mathbf{Q}\mathbf{X}_l)^T\}, \tag{8}$$

where $\mathbf{H}(\mathbf{X}_l) \in R^{K\times L}$ maps the examples in $\mathbf{X}_l$ to their K-bit hash codes. The indicator matrix $\mathbf{\Phi}$ incorporates the pairwise labeled information from $\mathbf{X}_l$ as:

$$\Phi_{ij} = \begin{cases} 1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{B}, \\ -1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}, \\ 0 & otherwise, \end{cases} \tag{9}$$

where $sgn(\mathbf{Q}^T\mathbf{X}_l)$ is the matrix of signs of individual elements. For semi-supervised hashing (i.e. $L \ll n$), the goal of it is to learn hash functions that try to minimize the empirical error on the labeled data, while maximally satisfying the desirable properties like variance and independence of individual bits. In order to prevent overfitting, one uses all the data $\mathbf{X}$ including the unlabeled examples as additional regularization terms, which can get better generalization ability and lead to desirable properties of hash codes [33]. Inspired by spectral hashing, one can generate hash codes in which bits are independent and each bit maximizes the information by generating a balanced partition of the data [33]. Normally, we relax the stronger conditions of independent to pairwise decor-relation of bits and the hash function$\sum_{i=1}^n h_k(\mathbf{x}_i) = 0, k = 1, \ldots, K$ should partition the entire dataset $\mathbf{X}$ into two sets of equal size. Thus, the relaxed maximizing objective function with constraints is represented as the following:

$$\begin{aligned} &\arg\max_{\mathbf{H}} \Psi(\mathbf{H}) \\ &s.t. \qquad \sum_{i=1}^n h_k(\mathbf{x}_i) = 0, k = 1, \ldots, K \\ &\tfrac{1}{n}\mathbf{H}(\mathbf{X})\mathbf{H}(\mathbf{X})^T = \mathbf{I}. \end{aligned} \tag{10}$$

Since the above function $\Psi(\mathbf{H})$ is non-differentiable even without the constraints and the balancing constraint makes the problem NP hard, one proposes the strategies to obtain an approximate solution, such as relaxing objective function and imposing orthogonality constraints. The strategy of relaxing objective function is that one replaces the sign of projection with its *signed magnitude*. This relaxation not only desires similar examples to have the same sign but also has large projection magnitudes. The function $\Psi(\mathbf{H})$ can be replaced by $\Psi(\mathbf{Q})$

$$\Psi(\mathbf{Q}) = \tfrac{1}{2}\mathbf{tr}\{\mathbf{Q}\mathbf{X}_l\mathbf{\Phi}\mathbf{X}_l^T\mathbf{Q}^T\}. \tag{11}$$

Relaxing the balancing constraint $\sum_{i=1}^n h_k(\mathbf{x}_i) = 0$ is equivalent to maximizing the variance for the k^{th} bit (see the Appendix 6.1).

Table 1
Youtube-Objects dataset.

	Class name									
	Aero	Bird	Boat	Car	Cat	Cow	Dog	Horse	Mbike	Train
Videos	13	16	17	9	21	11	24	15	14	15
Shots	482	175	191	212	245	70	217	151	444	324
Frames	79,487	34,517	119,448	27,607	59,822	41,158	86,306	70,392	68,421	132,998

It is hard to ensure perfectly balanced partitions when it uses the sign of projections to generate bits. The objective function of maximizing the variance condition can be derived as

$$\Psi(\mathbf{Q}) = -\frac{1}{2}\mathbf{tr}\{\mathbf{Q}\mathbf{X}_l\mathbf{\Phi}\mathbf{X}_l^T\mathbf{Q}^T\} + \frac{\eta}{2}\mathbf{tr}\{\mathbf{Q}\mathbf{X}\mathbf{X}^T\mathbf{Q}^T\} \tag{12}$$

The derivation is shown in Appendix 6.2. Since imposing hard orthogonality constraints may reduce the quality of lower bits (most of the variance is contained in top few projections), one converts them into a penalty term added to the objective function. This allows the learning algorithm to pick suitable directions by balancing various terms. The final objective function can be expressed as:

$$\begin{aligned}\Psi(\mathbf{Q}) &= -\frac{1}{2}\mathbf{tr}\{\mathbf{Q}\mathbf{X}_l\mathbf{\Phi}\mathbf{X}_l^T\mathbf{Q}^T\} + \frac{\eta}{2}\mathbf{tr}\{\mathbf{Q}\mathbf{X}\mathbf{X}^T\mathbf{Q}^T\} - \frac{\rho}{2}\parallel \mathbf{Q}\mathbf{Q}^T - \mathbf{I}\parallel_{\mathcal{F}}^2 \\ &= \frac{1}{2}\mathbf{tr}\{\mathbf{Q}\mathbf{M}\mathbf{Q}^\mathbf{T}\} - \frac{\rho}{2}\mathbf{tr}\{(\mathbf{Q}\mathbf{Q}^T - \mathbf{I})^T(\mathbf{Q}\mathbf{Q}^T - \mathbf{I})\}.\end{aligned} \tag{13}$$

where $\mathbf{M} = \mathbf{X}_l\mathbf{\Phi}\mathbf{X}_l^T + \eta\mathbf{X}\mathbf{X}^T$.

3.4. The bridge and optimization algorithm

In this section, we first introduce the bridge, which is the most important component of our model, because it integrates these two components (generative network and discriminant structure) by enforcing the correlation of their parameters. As introduced before, the generative network is used to minimize the reconstruction error via learning generative features from evolving data, and the discriminant structure is used to regularize the generative network via building pairwise similarity and dissimilarity constraints. The bridge builds a connection between feature parameters $\mathbf{W}$ of $\boldsymbol{\theta}$ learned from autoencoders and feature parameters $\mathbf{Q}$ from SSH. The generative features $\mathbf{W}$ learned from the denoising autoencoder is significantly useful for data reconstruction rather than classifier. Therefore, a good connection can make the features $\mathbf{W}$ be as consistent to the features $\mathbf{Q}$ as possible where $\mathbf{Q}$ is discriminative. It yields the following mixed objective function. Then, we give an optimization method to solve the proposed problem. Finally, we give our ISLSD algorithm.

$$\arg\min_{\boldsymbol{\theta},\mathbf{Q}} \alpha f(\boldsymbol{\theta}) + (1-\alpha) f(\mathbf{Q}) + \tfrac{\sigma}{2}\parallel \mathbf{Q} - \mathbf{W}\parallel_{\mathcal{F}}^2 + \beta \parallel \mathbf{W}\parallel_{\ell 1}, \tag{14}$$

where $\alpha \in [0, 1]$ is a guiding coefficient between autoencoders and SSH, σ is a correlation coefficient between $\mathbf{Q}$ and $\mathbf{W}$, β is sparsity penalty ratio, and two objectives of autoencoders ($f(\boldsymbol{\theta})$) and SSH ($f(\mathbf{Q})$) using in below are

$$f(\boldsymbol{\theta}) = \sum_{\mathbf{x}\in\mathbf{X}} \ell(\mathbf{x}, \hat{\mathbf{x}}) = \tfrac{1}{2}\sum_{\mathbf{x}\in\mathbf{X}} \parallel \mathbf{x} - \hat{\mathbf{x}} \parallel^2, \tag{15}$$

$$f(\mathbf{Q}) = \tfrac{1}{2}\mathbf{tr}\{\mathbf{Q}\mathbf{M}\mathbf{Q}^\mathbf{T}\} - \tfrac{\rho}{2}\mathbf{tr}\{(\mathbf{Q}\mathbf{Q}^T - \mathbf{I})^T(\mathbf{Q}\mathbf{Q}^T - \mathbf{I})\}. \tag{16}$$

When we do $\frac{\partial(f(\mathbf{Q}) + \frac{\sigma}{2}\|\mathbf{Q}-\mathbf{W}\|_{\mathcal{F}}^2)}{\partial\mathbf{Q}} = 0$, we can get $\mathbf{Q}((\sigma+\mathbf{1})\mathbf{I} + \frac{1}{\rho}\mathbf{M} - \mathbf{Q}^\mathbf{T}\mathbf{Q}) = 0$, obtaining a solution $\mathbf{Q}\mathbf{Q}^\mathbf{T}\mathbf{Q} = \mathbf{Q}((\sigma+1)\mathbf{I} + \frac{1}{\rho}\mathbf{M})$. The proposition (see the proof in Appendix 6.3) shows that $\mathbf{V} = (\sigma+1)\mathbf{I} + \frac{1}{\rho}\mathbf{M}$ is positive definite if the coefficient ρ and σ are chosen appropriately. Thus, we can decompose the $\mathbf{V}$ as $\mathbf{V} = \mathbf{L}\mathbf{L}^T$ according to Cholesky decomposition. Moreover, we can verify that $\mathbf{Q} = \mathbf{L}\mathbf{U}$ satisfies the equation of $\mathbf{Q}\mathbf{Q}^\mathbf{T}\mathbf{Q} = \mathbf{Q}((\sigma+1)\mathbf{I} + \frac{1}{\rho}\mathbf{M})$. The optimal solution of our problem $f(\mathbf{Q}) + \frac{\sigma}{2}\parallel \mathbf{Q} - \mathbf{W}\parallel_{\mathcal{F}}^2$ can be calculated by selecting its first k columns of $\mathbf{Q}$:

$$\mathbf{Q}^* = \mathbf{L}\mathbf{U}_k, \tag{17}$$

where $\mathbf{U}_k$ are the top k eigenvectors of $\mathbf{M}$.

In order to deal with non-differentiable with $\|\mathbf{W}\|_{\ell 1}$, we convert it into an approximation form, which add a pre-specified positive scale ϖ (negligibly small) and $\parallel \mathbf{W}\parallel_{\ell 1} \approx \sum_{ij}\sqrt{W_{ij}^2 + \varpi}$. Here, we try to take mini-batches to update for each iteration, where the true gradient is approximated by a sum over a small number of randomly training examples. During each iteration, we alternate the optimization of $\mathbf{Q}$ and $\mathbf{W}$ while fixing the other one. To be specific, the optimization method consists two steps: (1) fix $\boldsymbol{\theta}$ and update $\mathbf{Q}$ which is $\mathbf{Q} \leftarrow \mathbf{L}\mathbf{U}_k$; (2) fix $\mathbf{Q}$ and update $\boldsymbol{\theta}$ which is $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\frac{\partial\ell_2}{\partial\boldsymbol{\theta}}$. The pseudo-code of ISLSD is summarized in Algorithm 1.

3.5. Ensemble, extensions, advantages and disadvantages

Our model is general one, we can leverage special cases into it. As an example, an extend method (ISLSD-E) which ensemble autoencoders is utilized to learn features. To be specific, T autoencoders modules $\mathbf{\Omega} = \{(\mathbf{E}_i, \mathbf{D}_i)\}$, where $\mathbf{E}_i$ represents encoder map, and $\mathbf{D}_i$ represents decode map. We could train ensemble autoencoders modules and easily incorporate into our model. The difference between our models ISLSD and ISLSD-E is that ISLSD-E use more autoencoders (e.g., different initial hidden units). We compare ISLSD and ISLSD-E with different initial features (e.g., 200, 400 and 600). Given below is the loss function:

$$\ell_\lambda(\mathbf{\Omega}, \mathbf{x}) = \underbrace{\frac{1}{T}\sum_{i=1}^{T} \parallel \mathbf{E}_i\mathbf{D}_i\mathbf{x} - \mathbf{x}\parallel^2}_{reconstruction\ error} - \lambda \underbrace{\frac{1}{T}\sum_{i=1}^{T} \parallel \mathbf{E}_i\mathbf{D}_i\mathbf{x} - \frac{1}{T}\sum_{j=1}^{T}\mathbf{E}_j\mathbf{D}_j\mathbf{x}\parallel^2}_{diversity}. \tag{18}$$

Moreover, there are various techniques of autoencoders to improve their ability to capture important information and learn richer representations, such as sparse autoencoders [57], variational autoencoders (VAE) [58] and contractive autoencoders (CAE) [59]. These extensions for our proposed approach are our future directions.

Now, we analyze the advantages and disadvantages of our approach. Our method is able to take advantage of limited labeled examples in non-stationary environments. The dynamic features and sparse regularization are employed, which make ISLSD produce good, sparse and robust representations. The experimental study on synthetic and real-life datasets demonstrate that our method can effectively improve the classification accuracy and outperforms the existing state-of-the-art models. Moreover, our method can be easily extended, such as replacing the denoising autoencoders by CAE or incorporating CNN in the supervised setting. The disadvantage is that our method is not good at dealing with concept drift on streaming data. Designing the concept drift adaptation methods for our proposed model is also our future works.

Table 2
Comparison of the proposed ISLSD and ISLSD-E methods with some state-of-art approaches.

Data sets	Methods	Classification error							
		Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Avg.
IMDB	Learn++	16.46	17.60	–	–	–	–	–	17.03
	ADAIN.MLP	14.12	14.64	–	–	–	–	–	14.38
	Incremental SVM	14.21	14.57	–	–	–	–	–	14.39
	WEA	11.88	12.56	–	–	–	–	–	12.22
	COMPOSE	11.66	12.20	–	–	–	–	–	11.93
	SAND	10.08	9.80	–	–	–	–	–	9.94
	SARL	9.92	9.64	–	–	–	–	–	9.78
	TLP	7.85	8.12	–	–	–	–	–	**7.99**
	ISLSD	7.98	8.26	–	–	–	–	–	8.12
	ISLSD-E	7.50	7.40	–	–	–	–	–	**7.45**
ForestCover	Learn++	40.22	41.51	37.42	40.54	40.00	38.37	39.25	39.62
	ADAIN.MLP	36.95	36.04	36.12	37.73	37.65	37.66	36.91	37.00
	Incremental SVM	36.66	37.95	33.89	37.02	36.48	34.83	35.72	36.08
	WEA	34.15	36.08	36.99	35.85	36.39	36.09	35.06	35.80
	COMPOSE	34.08	33.00	34.68	32.64	32.72	32.98	30.85	32.99
	SAND	34.36	32.28	32.02	33.89	30.68	32.16	31.98	32.48
	SARL	34.42	33.34	32.18	34.33	31.28	32.83	32.75	33.02
	TLP	31.59	30.28	30.22	31.64	29.88	31.62	32.45	**31.10**
	ISLSD	32.21	32.20	31.03	31.86	31.73	32.52	32.98	32.07
	ISLSD-E	31.26	29.29	30.23	28.94	29.04	30.14	31.68	**30.08**

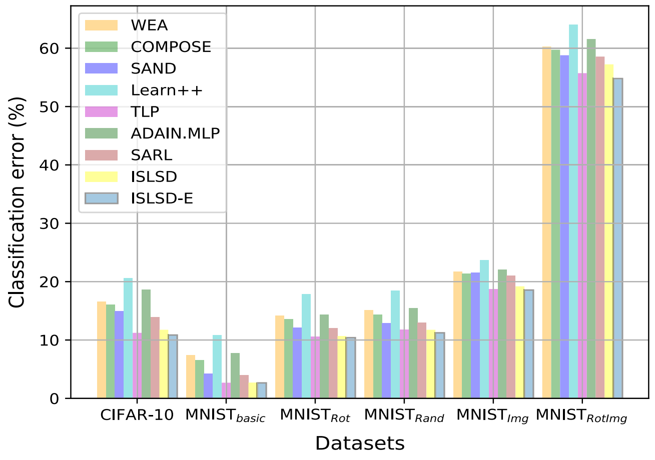


Fig. 4. Classification error rates on MNIST, its variations and CIFAR-10 datasets.

4. Experimental study

In this section, we first introduce datasets used in our experiments. Then, we compare the proposed methods with some baselines and study the effect on varying the number of labeled examples. We also show empirical results on some parameters, which affect our proposed methods, including analyzing on the effects of number of chunks, number of features, and hyper-parameters. Finally, we study the convergence of our algorithm and apply it into real-life dataset.

4.1. Datasets

The MNIST[1] dataset consists of a total of 70,000 hand-written digit examples, which each of size is 28×28 pixels and associated with a label from 0 to 9. Meanwhile, we also adopt MNIST variations[2] dataset, which consists of digits images with different variation types, such as rotation, random, or image background, and their combinations. We show the samples of the MNIST datasets and its variations in the Fig. 3. The CIFAR-10[3] dataset consists of a total of 60,000 32×32 color images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images in the official data.

The IMDB[4] dataset contains 25,000 highly polar moving reviews (good or bad) and 50,000 unlabeled examples for training and 25,000 examples for testing. The problem is to determine whether a given moving review has a positive or negative sentiment. The UCI Forest CoverType[5] dataset contains 581,012 examples with 54 attributes and 7 target classes. The actual forest cover type for a given observation (30×30 meter cell) was determined from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Independent variables were derived from data originally obtained from US Geological Survey (USGS) and USFS data where 40 attributes are binary columns representing soil type, 4 attributes are binary columns representing wilderness area, and the remaining 10 are quantitative attributes. The aim is to define to which type of cover this patch of forest belongs.

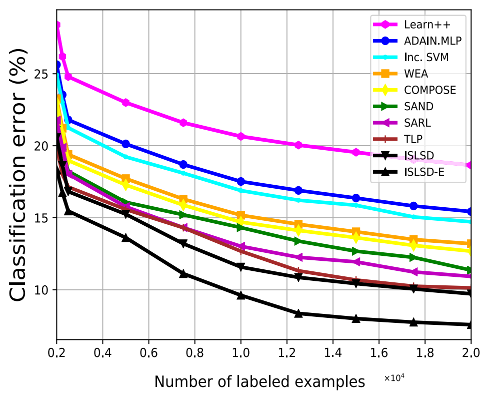
The Youtube-Objects[6] dataset is composed of videos collected from Youtube by querying for the names of 10 object classes of the PASCAL VOC Challenge. It contains between 9 and 24 videos per class; each video is 30–180 seconds. The videos are weakly annotated, i.e. each video is only ensured to contain one object of the corresponding class. The summarization of Youtube-Objects dataset is shown in Table 1.

4.2. Baselines and evaluation setup

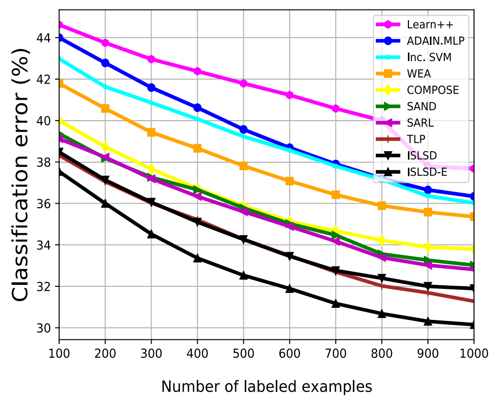
To validate the effectiveness of our method, we compare them with baselines and a number of related state-of-the-art approaches, which are enumerated as follows:

- **Learn++:** An algorithm for incremental training of neural networks, which utilizes ensemble of classifiers trained from sampled examples [11].
- **COMPOSE:** It introduces a computational geometry based framework to learn from non-stationary streaming data, where

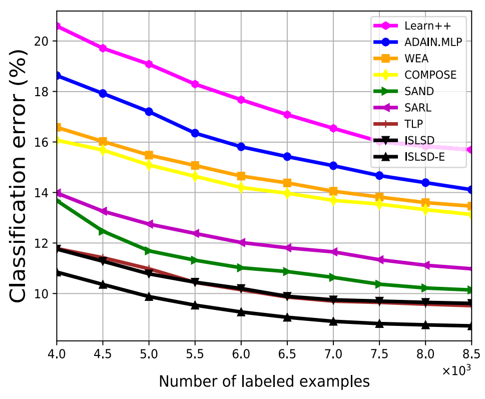
[1] http://yann.lecun.com/exdb/mnist/.
[2] http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations.
[3] http://www.cs.toronto.edu/~kriz/cifar.html.
[4] http://ai.stanford.edu/~amaas/data/sentiment/.
[5] https://archive.ics.uci.edu/ml/datasets/Covertype.
[6] http://calvin.inf.ed.ac.uk/datasets/youtube-objects-dataset/.



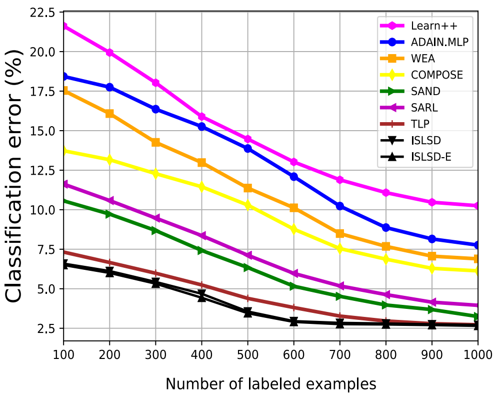
(a) IMDB



(b) ForstCover

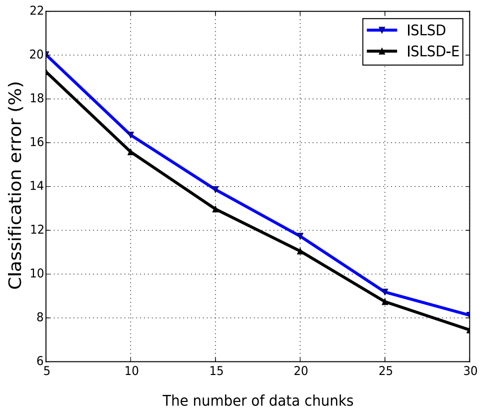


(c) CIFAR10

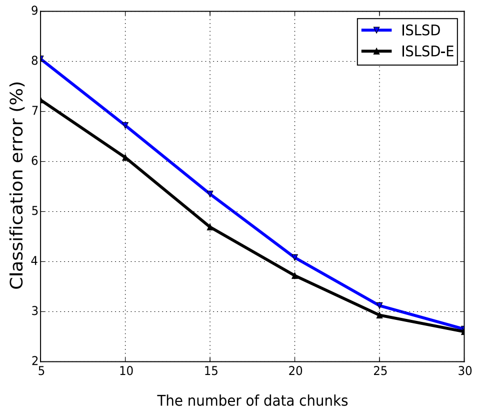


(d) MNIST

Fig. 5. Effect of number of labeled examples.



(a) IMDB



(b) MNIST

Fig. 6. Effect of number of data chunks.

labels are unavailable (or presented very sporadically) after initialization [56].

- **ADAIN.MLP:** A framework that could automatically learn from streaming data, which integrates the multilayer perceptron (MLP) into training process [60].
- **Incremental SVM:** It proposes a new design of storage and numerical operations, which speed up the training of incremental SVM for learning with limited resources [4].
- **WEA:** Weight Estimation Algorithm is an incremental ensemble based algorithm that uses labeled examples to build classifiers and unlabeled examples to aid in the calculation of the classifier voting weights [61].
- **SARL:** It exploits a sequence autoencoder to improve semi-supervised sequence learning with neural networks [62].
- **SAND** It is a semi-supervised framework which estimates classifier confidence in predicting instances from evolving data stream [8].
- **TLP** A principled approach for adapting the label propagation algorithm of [63] to streaming data [27].

Here, we introduce some package tools, which used in algorithms. We use TSVM[7] in COMPOSE and SAND framework, LIB-

[7] http://svmlight.joachims.org/;http://mloss.org/software/view/19/.

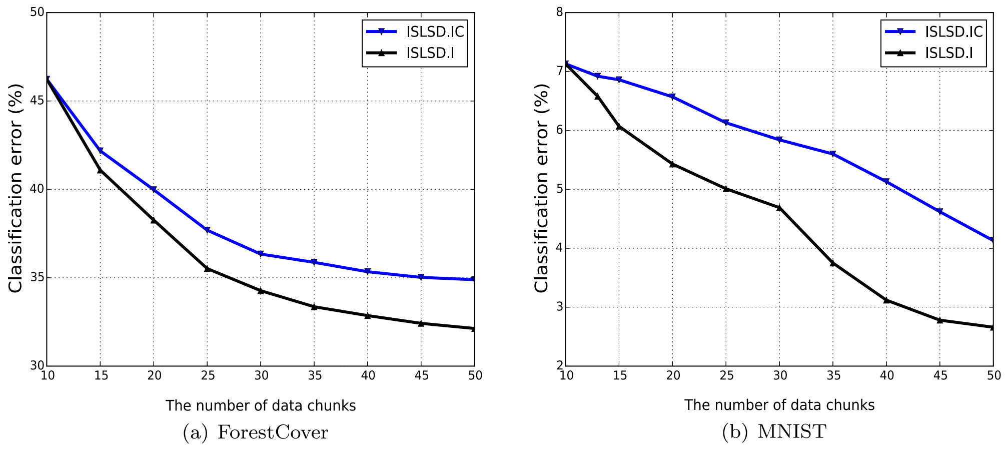


Fig. 7. New concept learning for the proposed method.

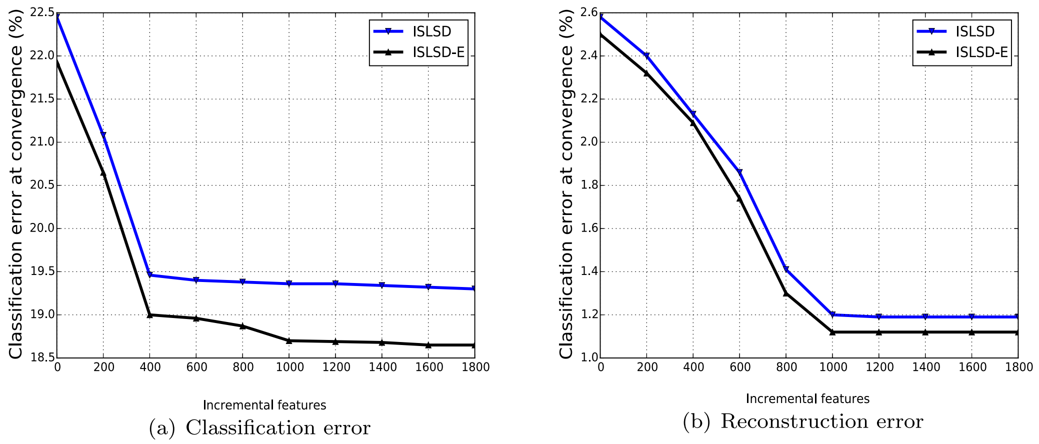


Fig. 8. Effect of dynamic features.

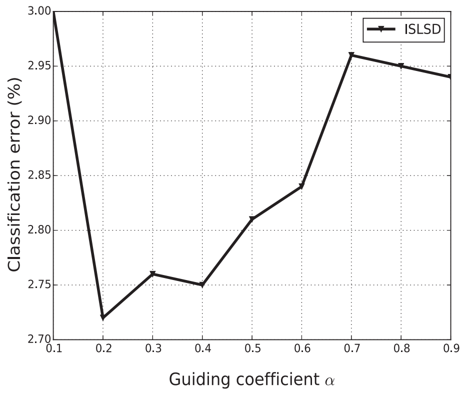
SVM[8] and efficient LIBLINEAR[9] are utilized in WEA algorithm. Stacking our networks is similar with stacking RBMs in deep belief networks (DBN) [64] or stacked denoising autoencoders [57,65]. It consists two stages: pre-training and fine-tuning. The greedy layer-wise strategy adopts this work [66]. Then, all parameters of the whole system can be fine-tuned using a back propagation technique (fine-tuning). As an empirical study of previous researchers, the hyper-parameters of autoencoders we referred in this work [67]. Such as, the learning rate of our model is set 0.01 or 0.001 when we train autoencoders. The guiding coefficient α, which trade-off the importance between denoising autoencoder and SSH, setting α at intervals of 0.1. The correlation coefficient σ between $\mathbf{W}$ and $\mathbf{Q}$ we set as 0.000001 or 0.0000001. The penalty ratio β, which controls the sparseness of $\mathbf{W}$, setting at intervals of 10%. The parameters η and ρ of SSH we choose using cross validation set. We also set $T = 3$ of ISLSD-E with different initial features (hidden units: e.g., 200, 400 and 600). Moreover, increasing λ of ISLSD-E yields an increasing level of emphasis on a diversity of reconstructions, we set $\lambda = 0.5$. The parameter γ of the mixed loss function with incremental autoencoder is set as between 0.3 and 0.6.

We evaluate each of the above algorithms in a non-stationary environment by training and then testing with chunks of data in sequence. In this simulation, each data is randomly picked into 30 chunks with identical size. 29 of 30 chunks are picked as training data, which is used for model estimation while one chunk as the test data is utilized to test the performance of our methods. The experiments repeat for 20 times and report the accuracy performance. At each time-stamp t, each chunk has a small set of labeled examples together with a large collection of unlabeled examples. First, we compute the objective ℓ for collected examples Z of chunk$_{t-1}$. Then, we greedily optimize the objective function by adding and merging features from examples of chunk$_t$. Finally, we choose the next chunk$_{t+1}$ when stopping criteria is met of each chunk$_t$. It will be demonstrated in below about the classification performance for running various baselines and related incremental SSL methods.

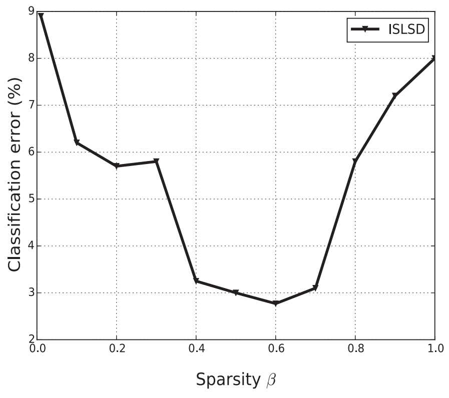
4.3. Comparing with various baselines

We systematically investigate the performance of the presented algorithms on five different datasets as summarized in Table 2 and Fig. 4. We report the classification error of Learn++, ADAIN.MLP, Incremental SVM, WEA, COMPOSE, SAND, SARL, TLP and our algorithms (ISLSD and ISLSD-E) in Table 2. We conduct our experiments on IMDB dataset with unlabeled examples and see that our methods obtain a strong performance than other methods. Meanwhile, ISLSD-E produces a better performance than ISLSD, showing ISLSD-E advantage on ensemble learning. 7.45% and 8.12% classification error can be achieved on the IMDB by ISLSD-E and ISLSD respectively. For ForestCover dataset, we sample 1000 labeled examples and the improved performance of our method is not obvious, but still better than other methods. In Fig. 4, we conduct our algorithms on MNIST, its variations and CIFAR-10 datasets, the

[8] http://www.csie.ntu.edu.tw/~cjlin/libsvm/.
[9] http://www.csie.ntu.edu.tw/~cjlin/linlinear/.



(a) Effect of α



(b) Effect of β

Fig. 9. Effect of hyper-parameters.

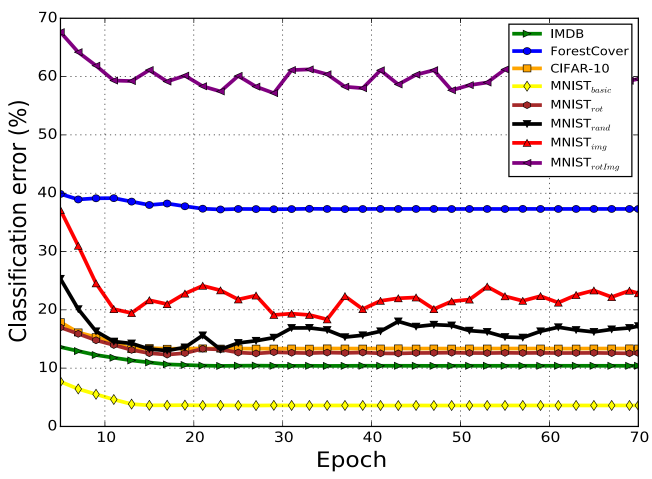


Fig. 10. Validation error rates of ISLSD on eight benchmark datasets.

Algorithm 1 The proposed ISLSD algorithm.

```
1: Input: Learning rate $\eta$, mini-batch size $m$, guiding coefficient $\alpha$, penalty coefficient $\sigma$, threshold $\mu$, sparse coefficient $\beta$ and positive coefficient $\rho$ and $\gamma$. Chunk data $X$ where labeled data is $X_l$ and time-stamp $t$.
2: Initialize: $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}, \mathbf{Q}$
3: repeat
4:     // Fix $\boldsymbol{\theta}$, Update $\mathbf{Q}$
5:     Pick $m$ samples from chunk$_{t-1}$
6:     Let $\ell_1 = (1-\alpha) f(\mathbf{Q}) + \frac{\sigma}{2} \parallel \mathbf{Q} - \mathbf{W} \parallel_{\mathcal{F}}^2 + \beta \parallel \mathbf{W} \parallel_{\ell 1}$
7:     Update parameters $\mathbf{Q}$ by
8:     $\mathbf{Q} \leftarrow \mathbf{Q}^* = \mathbf{LU}_k$
9:     // Fix $\mathbf{Q}$, Update $\boldsymbol{\theta}$
10:    Pick $m$ samples from chuck$_{t-1}$
11:    Let $\ell_2 = \alpha f(\boldsymbol{\theta}) + \frac{\sigma}{2} \parallel \mathbf{Q} - \mathbf{W} \parallel_{\mathcal{F}}^2 + \beta \parallel \mathbf{W} \parallel_{\ell 1}$
12:    Update parameters $\boldsymbol{\theta}$ by
13:    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial \ell_2}{\partial \boldsymbol{\theta}}$
14:    Compute the objective $\ell$ for test data of chunk$_{t-1}$
15:    Choosing a batch examples from chunk$_t$ into $Z$ ($Z \leftarrow \{\mathbf{x}\}$ if $\ell(\mathbf{x}) \geq \mu$)
16:    Select $2\Delta M$ candidates features and merge them into $\Delta M$ features (see the Merging Process: $\hat{\boldsymbol{\theta}}_{\mathcal{N}} \leftarrow \boldsymbol{\theta}_{\mathcal{N}}$)
17:    Add $\Delta N$ new features by greedily optimizing the objective function (Adding Process; Eq. (5): $\boldsymbol{\theta}_{\mathcal{N}} \leftarrow \boldsymbol{\theta}_{\mathcal{O}}$)
18:    Set $Z = \emptyset$ and update $\Delta N$ and $\Delta M$
19: until stopping criteria is met and choosing the next chunk$_{t+1}$;
```

same trend on these datasets is that our methods still achieve best performance. For variations of MNIST dataset, we sample 1000 labeled and 9000 unlabeled training examples. Since it exists some noise in these datasets, we can note that our methods still perform better than other methods. It is not very surprising to see that dynamic feature learning based on denoising autoencoder provides good performance because of the robust representations for learning latent features. And we also achieve the best result on MNIST dataset with 1000 labeled examples sampled, which the classification error is only 2.71%. Moreover, we experiment and sample 4000 labeled examples from CIFAR-10 dataset, thus our methods achieve good performance than other methods. Therefore, the result of experiments demonstrates that integrating dynamic feature learning and SSH into a joint learning framework can help to improve classification accuracy on streaming data.

4.4. Varying the number of labeled examples

In this section, we study the effect of our methods on different number of labeled examples. First, we set $\alpha = 0.3$ and $\sigma = 0.6$ with our methods, which validate the best performance in the below experiments. For IMDB dataset, we randomly sample 2k to 20k labeled samples incrementally; at the same time, use all the remaining data as unlabeled data. The classification error of all methods are shown in Fig. 5(a). Moreover we also randomly sample 100 to 1000 labeled examples from ForestCover and MNIST dataset, as well as 4k to 8.5k labeled examples from CIFAR-10 dataset, and the classification error of all methods are shown in Fig. 5(b–d). We also can find that our algorithms (ISLSD and ISLSD-E) are always performing better than other methods. In addition, the general trend is that all methods are performing better with increasing the number of labeled examples. Compared with ADAIN.MLP mehod, the overall prediction of Incremental SVM is limited. The performance of COMPOSE, SAND and TLP are more effective, but the improvements are less than ISLSD. Especially, in comparison with Learn++, our method has obviously improved. For instance, the difference of classification error is around 10% on CIFAR and MNIST datasets with 8.5k and 1k labeled examples respectively. As the same trend in the below experiments, ISLSD-E also perform better than ISLSD because it takes advantage of ensemble learning. All of the differences are repeated for 20 times and reported the classification error. Our methods produce a strong result on variety of datasets, showing ISLSD advantage on bridging the generative and discriminant features from denoising autoencoder and SSH into a joint framework.

Table 3
Experimental results on the Youtube-Objects dataset.

Methods	Prediction accuracy										
	aero	bird	boat	car	cat	cow	dog	horse	mbike	train	Avg.
Learn++	74.56	55.30	69.51	63.01	60.78	76.32	71.18	70.74	69.38	59.88	67.07
ADAIN.MLP	76.56	56.69	71.61	64.28	62.19	76.62	72.43	72.08	70.89	61.34	68.47
WEA	78.47	57.73	73.39	65.87	63.64	79.12	74.62	73.87	72.69	62.65	70.20
COMPOSE	78.97	58.34	73.49	64.18	64.16	78.92	74.80	74.47	72.96	63.00	70.33
SAND	79.56	58.82	75.02	67.12	64.39	80.91	75.12	75.69	74.39	64.78	71.58
SARL	80.14	59.19	75.07	67.62	65.13	80.82	76.12	75.58	74.21	64.15	71.80
TLP	81.92	**61.28**	**77.83**	**68.92**	**67.83**	**83.24**	**79.27**	**79.02**	**77.28**	**66.92**	**74.35**
ISLSD	**83.03**	61.15	77.61	67.45	67.32	81.04	78.93	78.25	76.78	66.34	73.79
ISLSD-E	**83.89**	**61.95**	**78.43**	**69.97**	**67.92**	**84.97**	**79.68**	**79.14**	**77.63**	**67.18**	**75.08**

4.5. Varying the number of data chunks

In this section, we evaluate our methods on varying the number of data chunks. As mentioned before, we divide the dataset into 30 chunks. We report the results on IMDB and MNIST datasets. The reason is that with increasing labeled and unlabeled examples from evolving data, SSH leverages similarity-preserving criterion and learns hash functions that try to minimize the empirical error on the labeled data. In addition, dynamic feature learning based on denoising autoencoders still can learn informative features from labeled and unlabeled examples. Meanwhile, we update the correlation of their parameters via the bridge. The corresponding classification error on IMDB and MNIST datasets are plotted in Fig. 6(a) and (b), respectively. This experimental results demonstrate that our algorithms are adapt for non-stationary environments.

4.6. New concept learning for the proposed method

We have studied the incremental nature of our method in the below, in this section, we would like to focus on the concept drift nature of our method. To do this, we also divide the ForestCover and MNIST datasets into chunks by sequence. However, different from the below setting, the data distribution is changed over time. In our simulation, all the examples in ForestCover are randomly and evenly divided into 50 data chunks. The data of class 1, class 2 and class 3 are spread into 50 data chunks. Data belonging to class 4 will be appeared in the 11th data chunk, data from class 5, class 6 and class 7 will be appeared in the 21th, 31th, 41th data chunk respectively. Similarly, all the examples of number 0 and number 5 in MNIST are randomly and evenly divided into 50 data chunks. The data from number 1 and number 9 will be introduced since the 11th data chunk; The data from number 2 and number 8 will be introduced since the 21th data chunk; The data from number 3 and number 7 will be introduced since the 31th data chunk; The data from number 4 and number 6 will be introduced since the 41th data chunk. We compare our method with incremental nature, referred as ISLSD.I. The new concept learning of our method is referred as ISLSD.IC. The performance of each class on ForestCover and MNIST datasets are shown in Fig. 7. Clearly, the performance of ISLSD.I is better than ISLSD.IC, but the performance of ISLSD.IC is not worse, especially for new concept learning. This experimental result demonstrate that our algorithms are adapt for non-stationary environments.

4.7. Effect of dynamic features

To evaluate the performance of dynamic features in our algorithms, we test on the mnist-background-image dataset and show the online classification and reconstruction error over the different number of features. For both ISLSD and ISLSD-E, the number of feature incremental ΔN_t and number of merged features ΔM_t at time-stamp t are adaptively determined by monitoring performance. We report the results with 200 initial features of ISLSD and 200, 400, 600 initial features of ISLSD-E for incremental learning. The classification and reconstruction error are shown in Fig. 8. The experimental results show that our algorithms have good convergence with dynamic features. This results also demonstrate that the dynamic feature learning could be an effective training method on streaming data.

4.8. Effect of hyper-parameters

In this section, we study the effect of guiding coefficient α and sparsity penalty β of our methods. α trade-off the importance between denoising autoencoder and SSH, and β which controls the sparseness of $\mathbf{W}$. We conduct the experiment on effect of varying the α while other parameters fixed. In practice, we set α=0.2 and fix other parameters on the set of β (1%, 10%, 20%,…,100%). Both experiments are sampled 1000 labeled examples from MNIST dataset. The result of classification error is showed in Fig. 9. Basically they first decrease and then increase as the guiding coefficient and sparsity increasing. The experimental results of these two parameters give us some guideline on the future work based on ISLSD.

4.9. Efficiency of the proposed algorithm

In this section, we experimentally validate the efficiency of our proposed method. Since the ISLSD-E is the special case for our proposed formulation, we adopt ISLSD algorithm in our efficiency experiments. In previous section, we have studied the effect of guiding coefficient α through the grid search on [0, 1]. For sparsity penalty β of our methods, we start with a smaller sparsity penalty ratio, such as $\beta = 0.0001$. Then we observe the sparsity of $\mathbf{W}$ and adjust β to the proper value. Moreover, we add the sparsity requirement to "stopping criteria" (the sparsity$\geq$50%). The validation error rates with classification error on all datasets are shown in Fig. 10. It needs only about 15 epochs to obtain almost the best validation error rate in most cases. It is not surprising that the variation of MNIST datasets seems to have oscillations during the validation process because random noise are added into these datasets. Following the experiments, which demonstrate that the proposed optimization algorithm is efficient.

4.10. Results on real-life dataset (Youtube-Object)

In this section, we apply our method into real-life dataset. The reason we adopt the Youtube-Object dataset is that learning from video data attract growing attention from both academia and industry. Besides, it fits the non-stationary setting by converting the video into sequence frames. We divide the annotate frames into training and test chunks. In the training chunks, we randomly sample 5% labeled examples of frames from each chunks, while in the test chunks we annotated all examples of the desired object class. In our experiments, we first use Object detection algorithm [68] to detect the objects for each frames, then we use our algorithms to classify the objects. For compared methods, we also choose the related methods as baselines. The same trend with baselines experimented in synthetic MNIST and CIFAR-10 datasets, our methods outperform other comparable methods. Specially, 73.79 and 75.08 average accuracy can be achieved by ISLSD and ISLSD-E respectively. From Table 3, we also achieve the best performance on every class of objects, which show the advantage of integrating autoencoder and SSH into a joint learning framework.

5. Conclusion

In this paper, we propose a novel incremental semi-supervised learning model on web-scale data. Each layer of model consists three components that is a generative network, a discriminant structure and the bridge. The generative network is regularized by discriminant structure that provides a sparsely-supervised guidance via building the pairwise similarity or dissimilarity constraints among data. Stacking our networks balance numerical tractability and the flexible utilization of supervision information as well as explores the intrinsic data structure. Moreover, our model can be extended and leverage several existing methods as special cases and their intrinsic relationships are elucidated. For evaluation, we conduct extensive experiments to compare the proposed algorithms with related methods on variety datasets and show that the proposed algorithms are quite effective.

In future work, we would like to design more efficient adaptation mechanism for our model since our approach is not very good at dealing with concept drift on streaming data. Moreover, we also consider three directions. First, we will consider to develop the end-to-end system for incremental semi-supervised learning. Second, we plan to integrate our method with homogeneous and heterogeneous transfer learning. Third, we would like to apply our model into more real-life datasets, such as sensor and traffic flow data.

Acknowledgments

We thank Kai Kang for generously sharing his object detection in videos source code. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported in part by the National Natural Science Foundation of China under Grant 61170035, 61272420 and 81674099, Six talent peaks project in Jiangsu Province (Grant No. 2014 WLW-004), the Fundamental Research Funds for the Central Universities (Grant No. 30920130112006), Jiangsu Province special funds for transformation of science and technology achievement (Grant No. BA2013047), the Research Innovation Program for College Graduates of Jiangsu Province (Grant No. KYLX15_0376), China Scholarship Council (Grant No. 201706840105), and the European Unions Horizon 2020 Research and Innovation Programme under the Marie Sklodowska-Curie (Grant No. 701697).

Appendix

A.1. Maximum variance condition

Proposition 1. *A hash function with maximum variance on data* $\mathbf{X}$ *must satisfy the balancing constraint, and vice-versa.*

$$\sum_i h(\mathbf{x}_i) = 0 \Longleftrightarrow \max var[h(\mathbf{x})] \tag{19}$$

Proof. Suppose one has m points with hash value -1, and $n-m$ points with hash value 1 in the set $\mathbf{X}$. Then, the expected hash value is

$$\mu = E[h(\mathbf{x})] = \tfrac{1}{n}\sum_i h(\mathbf{x}_i) = \tfrac{n-2m}{n}, \tag{20}$$

and the variance is

$$\begin{aligned} var[h(\mathbf{x})] &= E[(h(\mathbf{x})-\mu)^2] \\ &= \left(1-\frac{n-2m}{n}\right)^2\frac{(n-m)}{n} + \left(-1-\frac{n-2m}{n}\right)^2\frac{m}{n} \\ &= \frac{4}{n^2}(mn-m^2). \end{aligned} \tag{21}$$

Therefore, $var[h(\mathbf{x})]$ is concave with respect to m. Besides, since $var[h(\mathbf{x})]$ has a unique maximum $m = n/2$ $(\mu = 0)$, we can note that the balanced partitioning also maximizes the variance of hashing function. □

A.2. Derivation of Eq. (12)

One replaces the hard balancing constraint by a "soft" constraint, which maximizes the variance of the bits as

$$\Psi(\mathbf{Q}) = \frac{1}{2}\mathbf{tr}\{\mathbf{Q}\mathbf{X}_l\mathbf{\Phi}\mathbf{X}_l^T\mathbf{Q}^T\} + \frac{\eta}{2}\sum_k E[\| h_k(\mathbf{x}) - \mu_k \|^2]. \tag{22}$$

Here η is a positive scalar, which relatively weights the variance based regularization term. Moreover, to avoid addressing the non-differentiable $sgn(\cdot)$ function, one maximizes directly the variance of the projected data (If data is unit-norm, we can show that the variance of the projected data provides a lower bound on the maximum variance of bits). With this assumption, one can rewrite the Eq. (22) as

$$\Psi(\mathbf{Q}) = \frac{1}{2}\mathbf{tr}\{\mathbf{Q}\mathbf{X}_l\mathbf{\Phi}\mathbf{X}_l^T\mathbf{Q}^T\} + \frac{\eta}{2}\sum_k E[\| \mathbf{q}_k^T\mathbf{x} \|^2]. \tag{23}$$

Since $E[\mathbf{q}_k^T\mathbf{x}] = 0$. One can rewrite the variance term in the following:

$$\sum_k E[\| \mathbf{q}_k^T\mathbf{x} \|^2] = E[\mathbf{q}_k^T\mathbf{x}\mathbf{x}^T\mathbf{q}_k] = \frac{1}{n}\mathbf{tr}[\mathbf{Q}\mathbf{X}\mathbf{X}^T\mathbf{Q}^T]. \tag{24}$$

Replacing Eq. (24) in Eq. (23), and absorbing n in η, one can rewrite the overall objective function as

$$\Psi(\mathbf{Q}) = -\frac{1}{2}\mathbf{tr}\{\mathbf{Q}\mathbf{X}_l\mathbf{\Phi}\mathbf{X}_l^T\mathbf{Q}^T\} + \frac{\eta}{2}\mathbf{tr}\{\mathbf{Q}\mathbf{X}\mathbf{X}^T\mathbf{Q}^T\}. \tag{25}$$

A.3. Positive definite

Proposition 2. *The matrix* $\mathbf{V}$ *is positive definite if* $\rho \geq \max(0, -(1+\sigma)\bar{\lambda}_{\min})$*, where* $\bar{\lambda}_{\min}$ *is the smallest eigenvalue of* $\mathbf{M}$*.*

Proof. From definition of (16), $\rho \geq 0$. We can represent $\mathbf{M}$ as $\mathbf{U}diag(\lambda_1, \ldots, \lambda_D)\mathbf{U}^T$ where all $\lambda_i's$ are real, because $\mathbf{M}$ is symmetric from Eq. (13). Let $\bar{\lambda}_{\min} = \min(\lambda_1, \ldots, \lambda_D)$. Then $\mathbf{V}$ can be reformulated as

$$\begin{aligned} \mathbf{V} &= (\sigma+1)\mathbf{I} + \mathbf{U}diag(\frac{\lambda_1}{\rho}, \ldots, \frac{\lambda_D}{\rho})\mathbf{U}^T \\ &= \mathbf{U}diag(\frac{\lambda_1}{\rho}+1+\sigma, \ldots, \frac{\lambda_D}{\rho}+1+\sigma)\mathbf{U}^T. \end{aligned} \tag{26}$$

Therefore, $\mathbf{V}$ will have all eigenvalues positive if $\frac{\lambda_{\min}}{\rho} + 1 + \sigma \geq 0 \Rightarrow \rho \geq -(1+\sigma)\lambda_{\min}$.
□

References

[1] J. Gama, P. Kosina, et al., Learning decision rules from data streams, in: IJCAI Proceedings-International Joint Conference on Artificial Intelligence, 22, 2011, p. 1255.

[2] C.C. Loy, T.M. Hospedales, T. Xiang, S. Gong, Stream-based joint exploration-exploitation active learning, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 1560–1567.

[3] K. Fujii, H. Kashima, Budgeted stream-based active learning via adaptive submodular maximization, in: Advances in Neural Information Processing Systems, 2016, pp. 514–522.

[4] P. Laskov, C. Gehl, S. Krüger, K.-R. Müller, Incremental support vector learning: analysis, implementation and applications, J. Mach. Learn. Res. 7 (Sep) (2006) 1909–1936.

[5] G. Zhou, K. Sohn, H. Lee, Online incremental feature learning with denoising autoencoders, in: Artificial Intelligence and Statistics, 2012, pp. 1453–1461.

[6] G. Cauwenberghs, T. Poggio, Incremental and decremental support vector machine learning, in: Advances in Neural Information Processing Systems, 2001, pp. 409–415.

[7] Y. Wang, X. Fan, Z. Luo, T. Wang, M. Min, J. Luo, Fast online incremental learning on mixture streaming data, in: AAAI, 2017, pp. 2739–2745.

[8] A. Haque, L. Khan, M. Baron, Sand: semi-supervised adaptive novel class detection and classification over data stream, in: AAAI, 2016, pp. 1652–1658.

[9] L. Zhu, S. Pang, A. Sarrafzadeh, T. Ban, D. Inoue, Incremental and decremental max-flow for online semi-supervised learning, IEEE Trans. Knowl. Data Eng. 28 (8) (2016) 2115–2127.

[10] P.P. Rodrigues, J. Gama, J. Pedroso, Hierarchical clustering of time-series data streams, IEEE Trans. Knowl. Data Eng. 20 (5) (2008) 615–627.

[11] R. Polikar, L. Upda, S.S. Upda, V. Honavar, Learn++: an incremental learning algorithm for supervised neural networks, IEEE Trans. Syst. ManCybern. Part C (Appl.Rev.) 31 (4) (2001) 497–508.

[12] X. He, Incremental semi-supervised subspace learning for image retrieval, in: Proceedings of the 12th Annual ACM International Conference on Multimedia, ACM, 2004, pp. 2–8.

[13] A.B. Goldberg, X. Zhu, A. Furger, J.-M. Xu, Oasis: online active semi-supervised learning, in: AAAI, 2011, pp. 362–367.

[14] M.M. Masud, J. Gao, L. Khan, J. Han, B. Thuraisingham, A practical approach to classify evolving data streams: training with limited amount of labeled data, in: Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on, IEEE, 2008, pp. 929–934.

[15] P. Zhang, X. Zhu, L. Guo, Mining data streams with labeled and unlabeled training examples, in: Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, IEEE, 2009, pp. 627–636.

[16] L. Wei, E. Keogh, Semi-supervised time series classification, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 748–753.

[17] Y. Jia, S. Yan, C. Zhang, et al., Semi-supervised classification on evolutionary data., in: IJCAI, 2009, pp. 1083–1088.

[18] X. Zhu, A.B. Goldberg, Introduction to semi-supervised learning, Synth. Lect. Artif. Intell. Mach. Learn. 3 (1) (2009) 1–130.

[19] O. Chapelle, B. Scholkopf, A. Zien, Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews], IEEE Trans. Neural Netw. 20 (3) (2009) 542.

[20] J. Zhang, G. Tian, Y. Mu, W. Fan, Supervised deep learning with auxiliary networks, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 353–361.

[21] H. Wu, S. Prasad, Semi-supervised deep learning using pseudo labels for hyperspectral image classification, IEEE Trans. Image Process. 27 (3) (2018) 1259–1270.

[22] J. Snoek, R.P. Adams, H. Larochelle, Nonparametric guidance of autoencoder representations using label information, J. Mach. Learn. Res. 13 (Sep) (2012) 2567–2588.

[23] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, Z. Tu, Deeply-supervised nets, in: Artificial Intelligence and Statistics, 2015, pp. 562–570.

[24] C. Silberer, V. Ferrari, M. Lapata, Visually grounded meaning representations, IEEE Trans. Pattern Anal. Mach. Intell. 39 (11) (2017) 2284–2297.

[25] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, ACM, 2008, pp. 1096–1103.

[26] Z. Yang, W.W. Cohen, R. Salakhutdinov, Revisiting semi-supervised learning with graph embeddings, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning, in: ICML'16, 2016, pp. 40–48.

[27] T. Wagner, S. Guha, S.P. Kasiviswanathan, N. Mishra, Semi-supervised learning on data streams via temporal label propagation, in: International Conference on Machine Learning, 2018, pp. 5082–5091.

[28] Y. Li, Y. Wang, X. Jiang, Z. Dong, Teaching-to-learn and learning-to-teach for few labeled classification, in: Advanced Cloud and Big Data (CBD), 2016 International Conference on, IEEE, 2016, pp. 271–276.

[29] O. Chapelle, A. Zien, Semi-supervised classification by low density separation, in: AISTATS, 2005, pp. 57–64.

[30] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (1) (2006) 2399–2434.

[31] C. Gong, K. Fu, Q. Wu, E. Tu, J. Yang, Semi-supervised classification with pairwise constraints, Neurocomputing 139 (2014) 130–137.

[32] M. Bilenko, S. Basu, R.J. Mooney, Integrating constraints and metric learning in semi-supervised clustering, in: Proceedings of the Twenty-First International Conference on Machine Learning, ACM, 2004, p. 11.

[33] J. Wang, S. Kumar, S.-F. Chang, Semi-supervised hashing for scalable image retrieval, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 3424–3431.

[34] Z.-H. Zhou, M. Li, Tri-training: exploiting unlabeled data using three classifiers, IEEE Trans. Knowl. Data Eng. 17 (11) (2005) 1529–1541.

[35] X. Xu, W. Li, D. Xu, I.W. Tsang, Co-labeling for multi-view weakly labeled learning, IEEE Trans. Pattern Anal. Mach. Intell. 38 (6) (2016) 1113–1125.

[36] D.P. Kingma, S. Mohamed, D.J. Rezende, M. Welling, Semi-supervised learning with deep generative models, in: Advances in Neural Information Processing Systems, 2014, pp. 3581–3589.

[37] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, T. Raiko, Semi-supervised learning with ladder networks, in: Advances in Neural Information Processing Systems, 2015, pp. 3546–3554.

[38] S. Laine, T. Aila, Temporal ensembling for semi-supervised learning, in: Proceedings of the 5th International Conference on Learning Representations, 2017, pp. 108–120.

[39] A. Tarvainen, H. Valpola, Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results, in: Advances in Neural Information Processing Systems, 2017, pp. 1195–1204.

[40] N. Cesa-Bianchi, A. Conconi, C. Gentile, On the generalization ability of on-line learning algorithms, IEEE Trans. Inf. Theory 50 (9) (2004) 2050–2057.

[41] G. Chechik, U. Shalit, V. Sharma, S. Bengio, An online algorithm for large scale image similarity learning, in: Advances in Neural Information Processing Systems, 2009, pp. 306–314.

[42] S. Mehrkanoon, O.M. Agudelo, J.A. Suykens, Incremental multi-class semi-supervised clustering regularized by kalman filtering, Neural Netw. 71 (2015) 88–104.

[43] X.-Q. Zeng, G.-Z. Li, Incremental partial least squares analysis of big streaming data, Pattern Recognit. 47 (11) (2014) 3726–3735.

[44] D. Precup, J. Pineau, A.S. Barreto, On-line reinforcement learning using incremental kernel-based stochastic factorization, in: Advances in Neural Information Processing Systems, 2012, pp. 1484–1492.

[45] J.B. Gomes, M.M. Gaber, P.A. Sousa, E. Menasalvas, Mining recurring concepts in a dynamic feature space, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 95–110.

[46] M. Pratama, S.G. Anavatti, P.P. Angelov, E. Lughofer, Panfis: a novel incremental learning machine, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 55–68.

[47] T.I. Dhamecha, R. Singh, M. Vatsa, On incremental semi-supervised discriminant analysis, Pattern Recognit. 52 (2016) 135–147.

[48] M.J. Hosseini, A. Gholipour, H. Beigy, An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams, Knowl. Inf. Syst. 46 (3) (2016) 567–597.

[49] Z. Yu, P. Luo, J. You, H.-S. Wong, H. Leung, S. Wu, J. Zhang, G. Han, Incremental semi-supervised clustering ensemble for high dimensional data clustering, IEEE Trans. Knowl. Data Eng. 28 (3) (2016) 701–714.

[50] A. Haque, L. Khan, M. Baron, Semi supervised adaptive framework for classifying evolving data stream, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2015, pp. 383–394.

[51] J. Read, F. Perez-Cruz, A. Bifet, Deep learning in partially-labeled data streams, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, ACM, 2015, pp. 954–959.

[52] G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Learning in nonstationary environments: a survey, IEEE Comput. Intell. Mag. 10 (4) (2015) 12–25.

[53] A.G. Ororbia, D. Reitter, J. Wu, C.L. Giles, Online learning of deep hybrid architectures for semi-supervised categorization, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2015, pp. 516–532.

[54] L. Zhao, J. Chen, F. Chen, W. Wang, C.-T. Lu, N. Ramakrishnan, Simnest: Social media nested epidemic simulation via online semi-supervised deep learning, in: Data Mining (ICDM), 2015 IEEE International Conference on, IEEE, 2015, pp. 639–648.

[55] I. Misra, A. Shrivastava, M. Hebert, Watch and learn: Semi-supervised learning for object detectors from video, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3593–3602.

[56] K.B. Dyer, R. Capo, R. Polikar, Compose: a semisupervised learning framework for initially labeled nonstationary streaming data, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 12–26.

[57] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in Neural Information Processing Systems, 2007, pp. 153–160.

[58] C.K. Sønderby, T. Raiko, L. Maaløe, S.K. Sønderby, O. Winther, Ladder variational autoencoders, in: Advances in Neural Information Processing Systems, 2016, pp. 3738–3746.

[59] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contractive auto-encoders: explicit invariance during feature extraction, in: Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 833–840.

[60] H. He, S. Chen, K. Li, X. Xu, Incremental learning from stream data, IEEE Trans. Neural Netw. 22 (12) (2011) 1901–1914.

[61] G. Ditzler, R. Polikar, Semi-supervised learning in nonstationary environments, in: Neural Networks (IJCNN), The 2011 International Joint Conference on, IEEE, 2011, pp. 2741–2748.

[62] A.M. Dai, Q.V. Le, Semi-supervised sequence learning, in: Advances in Neural Information Processing Systems, 2015, pp. 3079–3087.

[63] X. Zhu, Z. Ghahramani, J.D. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 912–919.
[64] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[65] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (December) (2010) 3371–3408.
[66] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, S. Bengio, Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. 11 (Febuary) (2010) 625–660.
[67] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, Y. Bengio, An empirical evaluation of deep architectures on problems with many factors of variation, in: Proceedings of the 24th International Conference on Machine Learning, ACM, 2007, pp. 473–480.
[68] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, X. Wang, Object detection in videos with tubelet proposal networks, in: Proc. CVPR, 2, 2017, p. 7.

Yanchao Li received the B.S. degree in department of computer science from Nanjing Xiaozhuang University, China, in 2012, and the MSc degree in computer science from Nanjing University of Science and Technology, china, in 2015. Now, he is currently pursuing the Ph.D. degree with the department of computer science and engineering of Nanjing University of science and technology. Also, he is now as a visiting scholar in the School of Informatics, University of Edinburgh, U.K. His research interests are mainly in machine learning and data management.

Yongli Wang received the PhD degree from the Southeast University in 2006. He is currently a professor in the Department of Computer Science and Engineering at Nanjing University of Science and Technology, China. He is the author or coauthor of more than 60 papers in database and leading wireless network journals and conferences, such as various IEEE Transactions (IEEE TDKE, IEEE TPDS, IEEE/ACM Wireless Network, and IEEE TITB). His research includes intelligent data analysis, data streams management, mobile computing, data mining, cyber-physical system, health care monitoring etc., and is supported by the NCSF and other agencies. For his research activities, he also spent extended periods of time at Drexel University, USA. He is a member of the ACM and the IEEE.

Qi Liu received his BSc degree in Computer Science and Technology from Zhuzhou Institute of Technology, China in 2003, and his M.Sc. and Ph.D. in Data Telecommunications and Networks from the University of Salford, UK in 2006 and 2010. His research interests include context awareness, data communication in MANET and WSN, and data analysis on smart grid.

Cheng Bi received the B.S. degree from School of Computer Science and Engineering, Nanjing University of Science and Technology, China, in 2016. He is currently pursuing Ph.D. degree at the school of informatics of University of Edinburgh, UK. His research interests are Machine Learning and Artificial Intelligence.

Xiaohui Jiang received the B.S. degree in department of mathematics from Northeast University, China, in 2005, and the M.S. degree in the college of information science and engineering from Northeast University, China, in 2013. He is currently working towards the Ph.D. degree at the Nanjing University of science and technology. His research interests are machine learning and data mining.

Shurong Sun (1974-), Senior Engineer, Research interest include machine learning, smart grid, security of IoT.