Neurocomputing 92 (2012) 170–182

Contents lists available at SciVerse ScienceDirect

Neurocomputing

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

A framework for application-driven classification of data streams

Peng Zhang [a,b,*], Byron J. Gao [b], Ping Liu [a], Yong Shi [c], Li Guo [a]

[a] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[b] Department of Computer Science, Texas State University, San Marcos, TX 78666, USA
[c] FEDS Center, Graduate University, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Available online 13 March 2012

Keywords:
Data stream classification
Transfer learning
Semi-supervised learning
Relational *k*-means
Concept drifting

ABSTRACT

Data stream classification has drawn increasing attention from the data mining community in recent years. Relevant applications include network traffic monitoring, sensor network data analysis, Web click stream mining, power consumption measurement, dynamic tracing of stock fluctuations, to name a few. Data stream classification in such real-world applications is typically subject to three major challenges: concept drifting, large volumes, and partial labeling. As a result, training examples in data streams can be very diverse and it is very hard to learn accurate models with efficiency. In this paper, we propose a novel framework that first categorizes diverse training examples into four types and assign learning priorities to them. Then, we derive four learning cases based on the proportion and priority of the different types of training examples. Finally, for each learning case, we employ one of the four SVM-based training models: classical SVM, semi-supervised SVM, transfer semi-supervised SVM, and relational *k*-means transfer semi-supervised SVM. We perform comprehensive experiments on real-world data streams that validate the utility of our approach.

1. Introduction

Recent advances in computing technology and networking architectures have enabled generation and collection of unprecedented amount of *data streams* of various kinds, such as network traffic data, wireless sensor readings, Web page visits, online financial transactions and phone call records [5]. Consequently, data stream mining has emerged to be one of the most important research frontiers in data mining. Common stream mining tasks include classification [34,30], clustering [3] and frequent pattern mining [15]. Among them, data stream classification has drawn particular attention due to its vast real-world applications.

Example. In wireless sensor networks, data stream classification has been used to monitor environment changes. For example, in the sensor data collected by the Intel Berkeley Research Lab [37], each sensor reading contains information (temperature, humidity, light and sensor voltage) collected from 54 sensors deployed in the lab. The whole stream contains consecutive information recorded over a 2-month period (one reading per 1–3 min). By using the sensor ID as class label, the learning task is to correctly identify the sensor ID (one out of 54 sensors) purely based on the sensor data and the corresponding recording time.

Example. In power consumption analysis, data stream classification has been used to measure power consumptions. For example, the power supply stream collected by an Italian electricity company [37] contains hourly power supply of the company recording the power from two sources: power supplied from main grid and power transformed from other grids. The stream contains 3-year power supply records from 1995 to 1998, and the learning task is to predict which hour (1 out of 24 h) the current power supply belongs to.

Example. In information security, data stream classification has been widely used to monitor Web traffic streams. For example, the KDDCUP'99 intrusion detection dataset [4] was provided by the MIT Lincoln Labs collecting 9 weeks of raw TCP dump data for a local area network. The learning task is to build a predictive model capable of distinguishing between normal connections and intrusive connections such as DOS (denial-of-service), R2L (unauthorized access from a remote machine), U2R (unauthorized access to local super user privileges), and Probing (surveillance and other probing) attacks.

In these applications, the essential goal is *to efficiently build classification models from data streams for accurate prediction*. Compared to traditional stationary data, building prediction models from stream data faces three additional challenges:

- *Concept drifting*: In data streams, hidden patterns continuously change with time [26]. For example, in the wireless sensor

* Corresponding author at: Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China.
E-mail addresses: zhangpeng@ict.ac.cn (P. Zhang), bgao@txstate.edu (B.J. Gao), liuping@ict.ac.cn (P. Liu), yshi@gucas.ac.cn (Y. Shi), guoli@ict.ac.cn (L. Guo).

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.11.026

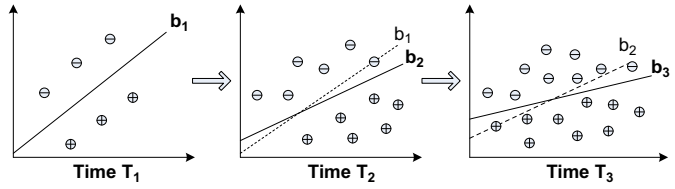


Fig. 1. An illustration of concept drifting in data streams. In the three consecutive time stamps T_1, T_2 and T_3, the classification boundary gradually drifts from b_1 to b_2 and finally to b_3.

stream, lighting during working hours is generally stronger than off-hours. Fig. 1 illustrates the concept drifting problem, where the classification boundary (concept) continuously drifts from b_1 to b_2, and finally to b_3 down the streams.

- *Large volumes*: Stream data come rapidly and continuously in large volumes. For example, the wireless sensor stream contains 2,219,803 examples recorded over a 2-month period (one reading per 1–3 min). It is impossible to maintain all historical stream records for in-depth analysis.
- *Partial labeling*: Due to large volumes of stream data, it is infeasible to label all stream examples for building classification models. Thus data streams are typically partially labeled and training data contain both labeled and unlabeled examples.

As a result, training examples in data streams are very diverse. To see why, let us assume data streams are buffered chunk by chunk. Examples in the most recent *up-to-date chunk* are training data, and examples in the *yet-to-come chunk* are testing data [31]. Due to the concept drifting, training examples in the up-to-date chunk often exhibit two distributions: target domain and similar domain, where the former represents the distribution of the testing data, and the latter represents a distribution similar to the target domain [9]. Then, training examples can be categorized into four types: labeled and from the target domain (Type I), labeled and from a similar domain (Type II), unlabeled and from the target domain (Type III) and unlabeled and from a similar domain (Type IV).

In order to build accurate prediction models from such diverse training examples with efficiency, it is necessary to closely examine the characteristics, in particular, proportion and learning priority, of the different types of examples in the training chunk.

- *Proportion*: The proportion of training examples from different types is determined by the concept drifting probability and labeling percentage (percentage of labeled examples). For example, when concept drifting is low and labeling percentage is high (low), the training chunk will have a large portion of Type I (III) examples. When concept drifting is high and labeling percentage is high (low), the training chunk will have a large portion of Type II (IV) examples.
- *Learning priority*: Generally, examples from the target domain (Types I and III) are capable of capturing the genuine concept of the testing data, and have a higher priority than examples from similar domains (Types II and IV). Besides, since Type I examples are labeled, they have a higher priority than Type III examples. Similarly, Type II examples have a higher priority than Type IV examples.

We take an example to explain how our framework can achieve accuracy with efficiency in building prediction models. If the Type I examples dominate the training chunk, according to the learning priority, we do not use the remaining three types of examples for training. By doing so, we gain in efficiency by building a simple model, comparing to a very complex model if we have to learn from all four types of training examples. On the other hand, the most informative examples, i.e., the ones in Type I, are utilized in model construction and the learning accuracy is not sacrificed comparing to some sophisticated model, e.g., TS^{3}VM, a very accurate yet complex learning model that we propose in this paper.

Based on the same observation and argument, we categorize learning from data streams into four cases: Type I dominates (Case 1), Type III dominates (Case 2), Type II dominates (Case 3) and Type IV dominates (Case 4). For Cases 1 and 2, we apply classical SVM and semi-supervised SVM, respectively, for training. For Cases 3 and 4, we propose two novel learning models, transfer semi-supervised SVM (TS^{3}VM) and relational k-means-based TS^{3}VM (RK-TS^{3}VM), for training.

The rest of the paper is organized as follows: Section 2 introduces categorization of training examples and learning cases. Sections 3 describes the corresponding learning models for the four learning cases. Section 4 reports experimental results. Section 5 surveys the related work. We conclude the paper in Section 6.

2. Categorization of training examples and learning cases

Consider a data stream S consisting of an infinite sequence of examples $\{x_i,y_i\}$, where $x_i \in \mathbb{R}^d$, d is the dimensionality and $y_i \in \{-1,+1\}$ indicates the class label of x_i. Note that y_i may not be always observed. Assume that the stream S arrives at a speed of n examples per second. The decision boundary (concept) underneath drifts with a probability of c, where $0 \le c \le 1$. Besides, assume that at each time stamp, a training chunk $D = \{x_1, \ldots, x_n\}$ is buffered and labeled by experts with a labeling rate of l per chunk where $0 < l < 1$.

Categorization of training examples: As discussed previously, due to the concept drifting, not all examples in the up-to-date chunk share the same distribution with the testing data in the yet-to-come chunk. In other words, examples in the up-to-date chunk could be generated from some similar domain instead of the target domain. Besides, since it is impractical to label all examples in the up-to-date training chunk, the training chunk will contain both labeled and unlabeled examples. By combining these two factors, we categorize training examples in data streams into four types.

Definition (*Four types of training examples*). In an up-to-date training chunk, there are four types of examples: labeled and from the target domain (Type I), labeled and from a similar domain (Type II), unlabeled and from the target domain (Type III) and unlabeled and from a similar domain (Type IV).

Fig. 2 illustrates the four types of training examples, where blue solid circles denote the Type I examples, red solid circles denote the Type II examples, blue hollow circles denote the Type III examples, and red hollow circles denote the Type IV examples. Due to the temporal correlation of concepts [18], Type I and III examples are usually located at the tail of a training chunk and close to the yet-to-come chunk. Type II and IV examples are usually located at the head of a training chunk and relatively far away from the yet-to-come chunk.

Estimation of number of examples: By estimating the number of examples of each type, we can gain insights into the training chunk and apply an appropriate learning model. Intuitively, the percentage of labeled examples depends on how fast labeling can be done by the experts, and the number of target domain examples depends on the concept drifting probability. By considering the two factors, the number of examples of each type can be estimated as follows.

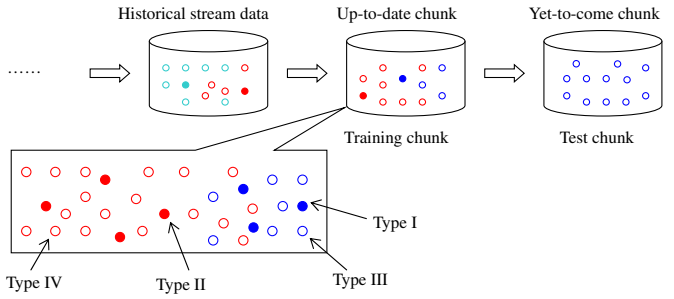


Fig. 2. An illustration of the four types of training examples in an up-to-date training chunk. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Theorem 1. *Let L_1, L_2, L_3 and L_4 be the number of examples of Type I, Type II, Type III and Type IV respectively in the up-to-date chunk. Then*

$$L_1 \propto \gamma \cdot c^{-1} \cdot l \cdot n$$

$$L_2 \propto (1-\gamma \cdot c^{-1}) \cdot l \cdot n$$

$$L_3 \propto \gamma \cdot c^{-1} \cdot (1-l) \cdot n$$

$$L_4 \propto (1-\gamma \cdot c^{-1}) \cdot (1-l) \cdot n \quad (1)$$

where $\gamma > 0$ is a constant coefficient.

Proof. Recall that stream S flows at a speed of n examples per second, the concept drifting probability is c, and the labeling rate is l. The number of target domain examples is inversely proportional to the concept drifting rate c with a coefficient of γ, so it can be easily estimated that $\gamma \cdot c^{-1} \cdot n$ examples in the up-to-date chunk have the same distribution as the testing data. The remaining $(1-\gamma \cdot c^{-1}) \cdot n$ examples have a similar distribution to the testing examples. From the estimates the theorem follows immediately. □

Learning priority: Not all the four types of training examples have to be used in model construction. For example, consider a data stream where the concept drifting is low and the labeling rate is high, the training chunk will have a large portion of Type I examples. In this case, we are able to build a satisfactory model by training only on the Type I examples. We observe that the four types of training examples have the following learning priorities.

Observation 1. *The learning priority of the four types of training examples is*

$$\textit{Type I} > \textit{Type III} > \textit{Type II} > \textit{Type IV} \quad (2)$$

What is the intuition behind Observation 1? Generally, examples from the target domain (Types I and III) are capable of capturing the genuine concept of the testing data, and thus have a high priority than examples from similar domains (Types II and IV). Besides, since Type I examples are labeled, they have a higher priority than Type III examples. Similarly, Type II examples have a higher priority than Type IV examples.

Based on Observation 1, when a particular type dominates the training examples, examples with lower priorities will not be used for training. For example, if Type III dominates the training examples, only Type I and Type III examples will be used for training. This is because Type I examples have a higher priority than Type III examples, and the remaining two types have lower priorities. By doing so, we gain in efficiency by building a simple model, comparing to a very complex model if we have to learn from all four types of training examples. On the other hand, the most informative examples are utilized in model construction and the learning accuracy is not sacrificed.

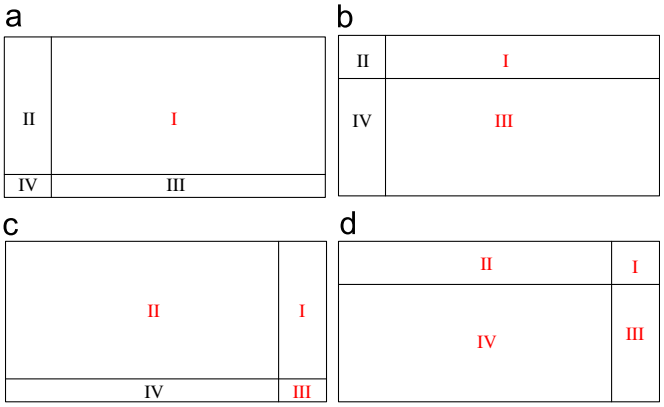


Fig. 3. The proportion of the four types of training examples with respect to different labeling rate l and concept drifting probability c. (a) l is high and c is low. Case 1, (b) both l and c are low. Case 2, (c) Both l and l are high. Case 3 and (d) l is low and c is high. Case 4.

Learning cases: Aiming at both accuracy and efficiency in learning prediction models, we categorize learning from data streams into the following four cases:

- *Case* 1: Type I dominates. When labeling rate is high and the concept drifting probability is low, Type I dominates the training examples. In this case, we can train a satisfactory model by using only Type I examples.
- *Case* 2: Type III dominates. When both labeling rate and concept drifting probability are low, Type III dominates the training examples. According to the learning priority, it is necessary to combine both Type I and Type III examples for training.
- *Case* 3: Type II dominates. When both labeling rate and concept drifting probability are high, Type II dominates the training examples, and we will use Type I, Type II and Type III examples for training.
- *Case* 4: Type IV dominates. When labeling rate is low and the concept drifting probability is high, Type IV dominates the training examples. This is the most difficult case because most examples are unlabeled and not from the target domain. According to the learning priority, we need to use all the four types of training examples for training.

These learning cases are further illustrated in Fig. 3.

3. Learning models

We have introduced the four learning cases. In this section, we present their corresponding learning models.

Throughout the section, $T_1 = (x_1, y_1), \ldots, (x_{L_1}, y_{L_1})$ denotes the set of Type I examples. $T_2 = \{(x_{L_1+1}, y_{L_1+1}), \ldots, (x_L, y_L)\}$ denotes the set of Type II examples, where $L = L_1 + L_2$. $T_3 = \{x_{L+1}, \ldots, x_{L+U}\}$ denotes the set of Type III examples, where U is the set of unlabeled examples. $T_4 = \{x_{L+U+1}, \ldots, x_{L+U+N}\}$ denotes the set of Type IV examples, where N is the set of unlabeled examples.

3.1. Case 1: Type I dominates

In this case, Type I examples T_1 dominate the training chunk and has the highest learning priority. Thus, only T_1 will be used for training. Formally, to learn from $T_1 = \{(x_1, y_1), \ldots, (x_{L_1}, y_{L_1})\}$, a generic SVM model can be trained by maximizing the margin

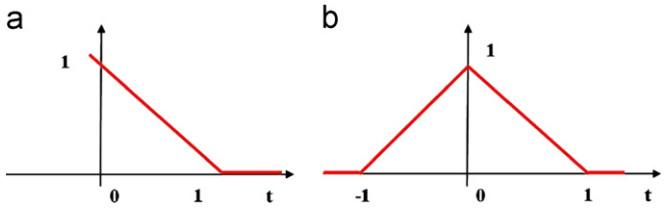


Fig. 4. An illustration of the Hinge loss function (a) $H(t)=\max(0,1-t)$, and the Symmetric Hinge loss function (b) $H(t)=\max(0,1-|t|)$. The Hinge loss function is equivalent to the following optimization problem: min ξ, s.t. : $\xi\geq 0$, $\xi\geq 1-t$.

distance between classes while minimizing the error rates as

$$\begin{aligned}\min\quad & \frac{1}{2}\|w\|^2+C\sum_{i=1}^{L_1}\xi_i\\ \text{s.t.}:\quad & y_i(wx_i+b)\geq 1-\xi_i\\ & \xi_i\geq 0,\quad 1\leq i\leq L_1\end{aligned}\tag{3}$$

where w is the projection direction, b is the classification boundary, ξ_i is the error distance from x_i to b, and parameter C is the penalty for the examples inside the margin.

The SVM model given in Eq. (3) is a constrained convex optimization problem. To simplify the expression, the Hinge loss function [8] in Fig. 4 can be used to transform Eq. (3) into an unconstrained convex optimization problem as

$$\min_{\theta}\quad \frac{1}{2}\|w\|^2+C\sum_{i=1}^{L_1}H(y_i f_\theta(x_i))\tag{4}$$

where $\theta=(w,b)$ and $f_\theta(x)=(wx+b)$.

3.2. Case 2: Type III dominates

In this case, Type III examples T_3 dominate the training chunk and Type I examples T_1 have a higher learning priority than Type III examples. Thus, both T_1 and T_3 will be used for training.

Learning from T_1 and T_3 is a semi-supervised learning problem [27]. Generally speaking, adding unlabeled T_3 examples into learning will further improve the performance for the following reasons: (1) labeled examples in T_1 are too few to build a satisfactory model. (2) T_3 contains a relatively large number of examples that come from the target domain, which can greatly help in differentiating the genuine classification boundaries.

Formally, in order to learn from both T_1 and T_3, semi-supervised SVM (S^3VM) [7] can be used as the learning model. The logic behind S^3VM is to find a classification boundary that achieves a maximum margin not only between labeled examples, but also unlabeled examples. That is, adding an extra term $C^*\sum_{i=L+1}^{L+U}H(|f_\theta(x_i)|)$ to penalize the misclassification of unlabeled examples located inside the margin as

$$\min_{\theta}\quad \frac{1}{2}\|w\|^2+C\sum_{i=1}^{L_1}H(y_i f_\theta(x_i))+C^*\sum_{i=L+1}^{L+U}H(|f_\theta(x_i)|)\tag{5}$$

Balance constraint: A possible limitation of the TS^3VM model is that all unlabeled examples in T_3 may be classified into one class with a very large margin, leading to deteriorated performance. To address this issue, an additional balance constraint should be added to ensure that unlabeled examples in T_3 be assigned into both classes. In the case that we do not have any prior knowledge about the class ratio in T_3, a reasonable approach [8] is to estimate its class ratio from T_1 and T_2 as

$$\frac{1}{U}\sum_{i=L+1}^{L+U}f_\theta(x_i)=\frac{1}{L_1}\sum_{i=1}^{L_1}y_i\tag{6}$$

where L denotes the number of labeled examples and U denotes the number of unlabeled examples.

By taking account of the balance constraint, we can derive a modified semi-supervised SVM model as

$$\begin{aligned}\min_{\theta}\quad & \frac{1}{2}\|w\|^2+C\sum_{i=1}^{L_1}H(y_i f_\theta(x_i))+C^*\sum_{i=L+1}^{L+U}H(|f_\theta(x_i)|)\\ \text{s.t.}:\quad & \frac{1}{U}\sum_{i=L+1}^{L+U}f_\theta(x_i)=\frac{1}{L_1}\sum_{i=1}^{L_1}y_i\end{aligned}\tag{7}$$

where $\theta=(w,b)$.

Obviously, Eq. (7) is a standard S^3VM model and can be easily solved by using off-the-shelf tools [19].

3.3. Case 3: Type II dominates

In this case, Type II examples T_2 dominate the training chunk, and Type I and Type III examples T_1 and T_3 have higher learning priorities than Type II examples. Thus, T_1, T_2 and T_3 will be used for training.

Accurately learning from these three types of examples is non-trivial. For this purpose, we design a novel transfer semi-supervised SVM model (TS^3VM for short). Intuitively, the TS^3VM model can be formulated by incorporating examples in T_1, T_2 and T_3 sequentially. Specifically, we can first formulate a generic SVM model by taking T_1 into consideration. Then, a transfer SVM model can be formulated by taking T_2 into consideration. Finally, we can include T_2 and formulate the TS^3VM model.

Learning from T_1 has been discussed in Eq. (4), based on which T_2 can be incorporated by applying the transfer learning strategy. Practically, transfer learning can use labeled examples in T_2 to refine the classification boundary by transferring the knowledge from T_2 to T_1. An effective way of doing so is to consider the problem as a multi-task learning procedure [13]. A common two-task learning SVM model on T_1 and T_2 can be formulated as

$$\begin{aligned}\min\quad & \frac{1}{2}\|w\|^2+C_1\|v_1\|^2+C_2\|v_2\|^2+C\sum_{i=1}^{L}\xi_i\\ \text{s.t.}:\quad & y_i((w+v_1)x_i+b)\geq 1-\xi_i,\quad 1\leq i\leq L_1\\ & y_i((w+v_2)x_i+b)\geq 1-\xi_i,\quad L_1+1\leq i\leq L\\ & \xi_i\geq 0,\quad 1\leq i\leq L\end{aligned}\tag{8}$$

where parameters C_1 and C_2 are the penalties on the two tasks, and v_1 and v_2 are the discrepancies between the global optimal decision boundary w and the local optimal decision boundary (i.e., $w+v_1$ for the task of learning from T_1 and $w+v_2$ for the task of learning from T_2).

In Eq. (8), parameters C_1 and C_2 control the preference between the two tasks. If $C_1>C_2$, task 1 is preferred over task 2; otherwise, task 2 is preferred over task 1. The relationship between C_1 and C_2 is furthered studied in the Experiments section. By using the Hinge loss function, Eq. (8) can be transformed into an unconstrained form

$$\min_{\theta}\quad \frac{1}{2}\|w\|^2+C_1\|V_1\|^2+C_2\|V_2\|^2+C\sum_{i=1}^{L}H(y_i f_\theta(x_i))\tag{9}$$

where $\theta=(w,v_1,v_2,b)$, $f_\theta(x)=(w+v_1)x+b$ for task 1 and $f_\theta(x)=(w+v_2)x+b$ for task 2.

In addition to T_1 and T_2, the additional semi-supervised learning method can be used to learn from the remaining T_3. As discussed in Eq. (5), by adding an extra term $C^*\sum_{i=L+1}^{L+U}H(|f_\theta(x_i)|)$ to penalize the misclassification of unlabeled examples in T_3 located inside the margin decided by Eq. (9), as well as the balance constraint in Eq. (6), we can finally get the TS^3VM

model as

$$\min_{\theta} \quad \frac{1}{2}\|w\|^2+C_1\|v_1\|^2+C_2\|v_2\|^2 +C\sum_{i=1}^{L}H(y_i f_\theta(x_i))+C^*\sum_{i=L+1}^{L+U}H(|f_\theta(x_i)|)$$

$$\text{s.t.}: \quad \frac{1}{U}\sum_{i=L+1}^{L+U}f_\theta(x_i)=\frac{1}{L}\sum_{i=1}^{L}y_i \tag{10}$$

where $\theta=(w,v_1,v_2,b)$, $f_\theta(x_i)=(w+v_1)x_i+b$ for $1\le i\le L_1$, $f_\theta(x_i)=(w+v_2)x_i+b$ for $L_1+1\le i\le L$, and $f_\theta(x_i)=wx_i+b$ for $L+1\le i\le L+U$.

Solution to the TS³VM objective function: As shown in Eq. (10), optimizing the objective function of TS³VM is a non-convex optimization problem, which is difficult to find global minima especially for large-scale problems. We propose to solve this non-convex problem by using Concave–Convex Procedure (CCCP), which has been developed by the optimization community [29,8,7]. CCCP decomposes a non-convex function into the sum of a convex function and a concave function, and then approximates the concave part by using a linear function (a tangential approximation). By doing so, the whole optimization procedure can be carried out iteratively by solving a sequence of convex problems. Algorithm 1 describes the CCCP algorithm in detail.

Algorithm 1. CCCP Algorithm.

Require: objective function $J(\theta)$
  generate an initial point θ_0 with a best guess
  $J(\theta)=J_{vex}(\theta)+J_{cav}(\theta)$
  repeat
    $\theta_{t+1}=argmin_\theta J_{vex}(\theta)+J'_{cav}(\theta_t)\cdot\theta$
  until convergence of θ
  return a local minima solution θ^*

From the CCCP perspective, we can observe that the first four terms of TS³VM are convex functions, whereas the last Symmetric Hinge loss part $C^*\sum_{i=L+1}^{L+U}H(|f_\theta(x_i)|)$ makes it a non-convex model. Thus, we will decompose and analyze the last part by using the CCCP method. To simplify the notation, we denote $z_i=f_\theta(x_i)$, so the last part can be rewritten as $C^*\sum_{i=L+1}^{L+U}H(|z_i|)$. Considering a specific z_i (without loss of generality, we denote it as z here), the Symmetric Hinge loss on z can be denoted by $J(z)$ as

$$J(z)=C^*H(|z|) \tag{11}$$

Eq. (11) is a non-convex function, which can be split into a convex part and a concave part as

$$J(z)=C^*H(|z|)=\underbrace{C^*\max(0,1-|z|)+C^*|z|}_{J_{vex}(t)}\underbrace{-C^*|z|}_{J_{cav}(t)} \tag{12}$$

According to Algorithm 1, the next iterative point can be calculated by the approximation of the concave part J_{cav} as

$$\frac{\partial J_{cav}(z)}{\partial z}\cdot z=\begin{cases}C^*z, & z<0\\ -C^*z, & z\ge 0\end{cases} \tag{13}$$

and then minimizing

$$J(z)=C^*\cdot\max(0,1-|z|)+C^*|z|+\frac{\partial J_{cav}(z)}{\partial z}z \tag{14}$$

If $z<0$ in the current iteration, then in the next iteration, the current effective loss can be denoted as

$$L(z,-1)=\begin{cases}2C^*z, & z\ge 1\\ C^*(1+z), & |z|<1\\ 0, & z\le -1\end{cases} \tag{15}$$

On the other hand, if $z\ge 0$, then in the next iteration, the current effective loss can be denoted as

$$L(z,+1)=\begin{cases}0, & z\ge 1\\ C^*(1-z), & |z|<1\\ -2C^*z, & z\le -1\end{cases} \tag{16}$$

By doing so, within each iteration, when taking all $z_i=f_\theta(x_i)$ into consideration, solving the TS³VM model is equivalent to solving Eq. (17) under the balance constraint equation (6)

$$\min_{\theta} \quad \frac{1}{2}\|w\|^2++C_1\|v_1\|^2+C_2\|v_2\|^2 +C\sum_{i=1}^{L}H(y_i f_\theta(x_i))+\sum_{i=L+1}^{L+U}L(f_\theta(x_i),y_i) \tag{17}$$

where y_i $(L+1\le i\le L+U)$ is the class label of x_i that has been assigned in the previous iteration. If $y_i<0$, Eq. (15) will be used to calculate the loss function; otherwise, Eq. (16) will be used to calculate the loss function. The detailed description of solving TS³VM is given in Algorithm 2.

Algorithm 2. TS³VM learning model.

Require: T_1, T_2 and T_3
  use T_1 and T_2 to build a transfer SVM model as shown in Eq. (8), and generate an initial point $\theta_0=(w_0,v_{10},v_{20},b_0)$
  repeat
    $y_i\leftarrow sgn(wx_i+b)$, $\forall L+1\le i\le L+U$
    $\theta\leftarrow$ calculate Eq. (17) under the balance constraint equation (6)
  until y_i remains unchanged, $\forall L+1\le i\le L+U$
  return $f(x)=sgn(wx+b)$

Theorem 2 (*Convergence of TS³VM*). *The TS³VM learning model in Algorithm 2 converges after a limited number of iterations.*

Proof. In Algorithm 2, in each iteration t, the objective function $J(\theta_t)$ is split into a convex part $J_{vex}(\theta_t)$ and a concave part $J_{cav}(\theta_t)$. Then, in the next iteration $t+1$, the point θ_{t+1} is the minimal solution of the current objective function, and we have

$$J_{vex}(\theta_{t+1})+J'_{cav}(\theta_t)\theta_{t+1}\le J_{vex}(\theta_t)+J'_{cav}(\theta_t)\theta_t \tag{18}$$

Meanwhile, because the concavity of $J_{cav}(\theta)$, we have

$$J_{cav}(\theta_{t+1})\le J_{cav}(\theta_t)+J'_{cav}(\theta_t)(\theta_{t+1}-\theta_t) \tag{19}$$

By adding both sides of Eqs. (18) and (19), we have

$$J_{vex}(\theta_{t+1})+J_{cav}(\theta_{t+1})+J'_{cav}(\theta_t)\theta_{t+1} \le J_{vex}(\theta_t)+J'_{cav}(\theta_t)\theta_t+J_{cav}(\theta_t)+J'_{cav}(\theta_t)(\theta_{t+1}-\theta_t) \tag{20}$$

Move the third item on the left-hand side of Eq. (20) to the right-hand side, we have

$$J_{vex}(\theta_{t+1})+J_{cav}(\theta_{t+1})\le J_{vex}(\theta_t)+J'_{cav}(\theta_t)\theta_t +J_{cav}(\theta_t)+J'_{cav}(\theta_t)(\theta_{t+1}-\theta_t)-J'_{cav}(\theta_{t+1})\theta_{t+1} \tag{21}$$

The right-hand side of the above inequation equals to $J_{vex}((\theta_t))+J_{cav}(\theta_t)$. Therefore, the objective function will decrease after each iteration $J_{vex}(\theta_{t+1})\le J(\theta_t)$. □

Consequently, Algorithm 2 will converge after a limited number of iterations. In fact, as long as the initial point is carefully selected (i.e., using a multi-task SVM model built on T_1 and T_2 as the initial point), Algorithm 2 will converge very fast.

3.4. Case 4: Type IV dominates

This is the most complex learning case. In this case, Type IV examples T_4 dominate the training chunk and has the lowest learning priority. Thus, it is necessary to use all T_1, T_2, T_3 and T_4 for training.

To solve this learning problem, we design a novel Relational K-means-based Transfer Semi-Supervised learning model (RK-TS^{3}VM for short). The TS^{3}VM model, as discussed previously, is used to learn from T_1, T_2 and T_3. Now we discuss how to learn from T_4 using a Relational K-means model [38] (RK for short).

Learning from T_4 is more challenging than from other three types of training examples, mainly because examples in T_4 are unlabeled and have different distributions from the target domain. The aim of the RK model is to *transfer knowledge from T_4 to T_1, T_2 and T_3 by constructing some new features for the three types of examples using the relational information between T_1, T_2, T_3 and T_4*.

An example of RK learning is shown in Fig. 5, where T_4 examples are first clustered into k clusters, $G_1,\ldots,G_k$ based on a relational matrix built between T_1 and T_4. After that, k new features $f(x_i,G_\tau)$ $(\tau=1,\ldots,k)$ are added to each example x_i in T_1 to construct a new data set T_1' by calculating the relationship between x_i and each cluster center. By doing so, the new data set T_1' will contain information transferred from T_4, which can help to build a more accurate prediction model.

Given L_1 examples in T_1 and N examples in T_4, the purpose of the relational k-means clustering is to cluster instances in T_4 into k groups, by taking the relationships between instances in T_1 and T_4 into consideration. Let $W\in\mathbb{R}^{L_1\times N}$ denote the similarity matrix between T_1 and T_4 with each $w_{i,j}$ indicating the similarity (which can be calculated according to the Euclidian distance) between instance x_i in T_1 and instance x_j in T_4. For each cluster G_τ on W the average pairwise similarity for all examples in G_τ can be defined as

$$S_{G_\tau}=\frac{1}{|G_\tau|^2}\sum_{x\in G_\tau}\sum_{x'\in G_\tau}S(x,x') \tag{22}$$

where $S(x,x')$ denotes the similarity between two examples of x and x'. On the other hand, the variance of the relationship values of all examples in G_τ can be calculated as

$$\delta_{G_\tau}=\frac{1}{|G_\tau|}\sum_{y_i\in G_\tau}(\beta_j-\beta_{G_\tau})^T(\beta_j-\beta_{G_\tau}) \tag{23}$$

where β_{G_τ} denotes the average relationship vector of all instances in G_τ, and $\beta_i\in\mathbb{R}^{1\times L_1}$ denotes the relationships of instance x_j with respect to all examples in T_1. The objective of the relational k-means is to find k groups, G_τ, $\tau=1,\ldots,k$, such that the sum of the similarities is maximized while the sum of variances is minimized as

$$J_e'=\max\sum_{\tau=1}^{k}J_{G_\tau}=\max\sum_{\tau=1}^{k}\frac{S_{G_\tau}}{\delta_{G_\tau}} \tag{24}$$

Information from T_1 | Information from T_4 | Class label for T_1

A_1	A_2	..	A_d	G_1	..	G_k	Y
1	2	..	5	$f(x_1,G_1)$	..	$f(x_1,G_k)$	1
..	..	..	..	..	..	..	..
3	7	..	1	$f(x_{L_1},G_1)$	..	$f(x_{L_1},G_k)$	2

Fig. 5. An illustration of the RK learning model.

Explicitly solving Eq. (24) is very difficult. Alternatively, we can use a recursive hill-climbing search process as an approximation solution. Assume that examples in T_4 are clustered into k clusters, $G_1,\ldots,G_k$. Moving an instance x from cluster G_i to cluster G_j changes only the cluster objective values J_{G_i} and J_{G_j}. Therefore, in order to maximize Eq. (24), at each step t, we randomly select an example x from a cluster G_i, and move it to cluster G_j. Such a move is accepted only if the inequity (25) achieves a higher value at step $t+1$

$$J_{G_i}(t)+J_{G_j}(t)<J_{G_i}(t+1)+J_{G_j}(t+1) \tag{25}$$

Based on the search process in inequity (25), major steps of the relational k-means are listed in Algorithm 3.

Algorithm 3. Relational k-means clustering.

Require: T_1, T_4, number of clusters k, and number of iterations T
  $W\leftarrow$ calculate similarity matrix between T_1 and T_4
  $G_1,\ldots,G_k\leftarrow$ apply k-means to W
  for $t\leftarrow 1$ to T **do**
    $x\leftarrow$ randomly select an example from T_4
    $G_i\leftarrow$ current cluster of example x
    $J_{G_i}(t)\leftarrow$ calculate G_i's objective value in Eq. (24)
    $J_{G_i}(t+1)\leftarrow$ G_i's new value after excluding x
    for $j\leftarrow 1$ to k, $j\neq i$ **do**
      $J_{G_j}(t)\leftarrow$ calculate G_j's objective value
      $J_{G_j}(t+1)\leftarrow$ G_j's new value after including x
      if inequity (25) is *true* **then**
        $G_j\leftarrow G_j\cup x$; $G_i\leftarrow G_i\setminus x$
        break
      end if
    end for
  end for
  $\mu_1,\ldots,\mu_k\leftarrow$ calculate cluster centers for $G_1,\ldots,G_k$
  return $\mu_1,\ldots,\mu_k$

Algorithm 3 has three tiers of loops. Within each tier, it needs to frequently recalculate $J_{G_c}(t)$ when the current examples are removed from its current group to another. Nevertheless, because $J_{G_c}(t)$, as shown in Eq. (24), contains information from both the similarity S_{G_i} and variance δ_{G_i} in the relationship matrix, frequently recalculating $J_{G_c}(t)$ will be time-consuming. To alleviate this problem, we introduce an addictive update method and a subtractive update method to recalculate $J_{G_c}(t)$.

Consider an example x in T_4 that moves from group G_i to G_j. Before the move, β_{G_i} and β_{G_j} are the mean vectors, δ_{G_i} and δ_{G_j} are the variance vectors. After the move, the new groups are G_i' and G_j'. Then the addictive update is given in the following theorem:

Theorem 3 (*Additive update*). *When adding an example x into G_j, the mean vector of G_j, β_{G_i}, can be updated to β'_{G_i} as follows*:

$$\beta'_{G_j}=\frac{1}{|G_j'|}\sum_{x_i\in}\beta_l=\beta_{G_i}+\frac{\beta_k-\beta_{G_j}}{n_j+1} \tag{26}$$

Meanwhile, the variance δ_{G_j} can be updated to δ'_{G_j} as follows:

$$\begin{aligned}\delta'_{G_j}&=\frac{1}{|G_j'|}\sum_{y_l\in G_j'}(\beta_l-\beta'_{G_j})^T(\beta_l-\beta'_{G_j})\\&=\frac{n_j}{n_j+1}\delta_{G_j}+\frac{n_j}{(n_j+1)^2}(\beta_k-\beta_{G_j})^T(\beta_k-\beta_{G_j})\end{aligned} \tag{27}$$

where n_j is the number of examples in G_j.

Therefore, the updated mean and variance vectors of group G_j can be incrementally calculated, without recalculating Eq. (24).

Similarly, for a group G_i, where an example x is removed, its mean and variance vectors can be updated using the following theorem.

Theorem 4 (*Subtractive update*). *When an example x is removed from group G_i, the mean vector β_{G_i} can be updated to β'_{G_i} as follows*:

$$\beta'_{G_i} = \frac{1}{|G'_i|}\sum_{y_l \in G'_i}\beta_l = \frac{1}{n_i-1}(n_i\beta_{G_i}-\beta_k) = \beta_{G_i}-\frac{\beta_k-\beta_{G_i}}{n_i-1} \tag{28}$$

Meanwhile, the variance δ_{G_i} can be updated to δ'_{G_i} as follows:

$$\begin{aligned}\delta'_{G_i} &= \frac{1}{|G'_i|}\sum_{y_l \in G'_i}(\beta_l-\beta'_{G_i})^T(\beta_l-\beta'_{G_i}) \\ &= \frac{n_i}{n_i-1}\delta_{G_i}-\frac{n_i}{(n_i-1)^2}(\beta_k-\beta_{G_i})^T(\beta_k-\beta_{G_i})\end{aligned} \tag{29}$$

where n_i is the number of examples in G_i.

Time complexity: Now we analyze the time complexity of Algorithm 3. In Algorithm 3, when searching for a new group for each example in the relationship matrix, the updating operation, by using Theorems 3 and 4, can be executed within constant time $O(1)$. Besides, Algorithm 3 is a greedy algorithm. In each iteration, it uses a local optimization technique to cluster examples into groups that maximizes Eq. (24). There are three tiers of loops in the algorithm. The first tier aims to find the best group for each example x with the worst-case complexity of $O(k)$ (i.e., traversing all the k groups). The second tier aims to find the best groups for all examples in T_4, which has the worst-case complexity of $O(N)$ (i.e., searching over all the N examples). The last tier aims to make the algorithm converge to a stable solution. Obviously, the first two tiers dominate the time consumption of the whole algorithm, and thus the time complexity of Algorithm 3 is $O(k)\times O(N) = O(kN)$.

RK-TS^{3}VM learning model: Algorithm 4 lists the detailed procedures of the RK-TS^{3}VM learning model, which is the combination of the TS^{3}VM and RK learning models. Given a training chunk D, Step 1 identifies the four types of examples T_1, T_2, T_3 and T_4. Step 2 constructs a group of k feature vectors, denoted by $\mu = \{\mu_1, \ldots, \mu_k\}$, by applying RK to T_1 and T_4. In Steps 3 and 4, the k new features are appended to each example in T_1, T_2 and T_3 to form three new sets denoted by T'_1, T'_2 and T'_3, respectively. Step 5 builds a TS^{3}VM model F from T'_1, T'_2 and T'_3. In Step 6, the feature vectors μ and F are combined to form the final prediction model. For any example x in the testing chunk, RK-TS^{3}VM first calculates k new features for x, then uses the TS^{3}VM model to predict a label for x.

Algorithm 4. RK-TS^{3}VM learning model.

Require: training chunk D, chunk size n, labeling rate l, concept drifting probability c, number of clusters k

Step 1: identify T_1, T_2, T_3 and T_4 in D according to the labeling rate l and concept drifting probability c using Eq. (1)

Step 2: use RK model on T_1 and T_4 to get k cluster centers denoted by $\mu = \{\mu_1, \ldots, \mu_k\}$

Step 3: for each example x in T_1, T_2 and T_3, add k attributes using the inner product between x and μ

Step 4: get the new examples T'_1, T'_2 and T'_3 from **Step 3**

Step 5: construct TS^{3}VM from T'_1, T'_2 and T'_3, and use it to train a model F

return μ and F together as the prediction model

4. Experiments

In this section, we report experimental results. The purpose of the experiments is to validate the following arguments: (1) when T_1 dominates the training examples, using SVM to train on T_1 examples is a desirable trade-off solution. (2) When T_3 dominates the training examples, using S^{3}VM to train on both T_1 and T_3 examples is a desirable trade-off solution. (3) When T_2 dominates the training examples, using TS^{3}VM to train on T_1, T_2 and T_3 examples is a desirable trade-off solution. (4) When T_4 dominates the training examples, using RK-TS^{3}VM is a desirable trade-off solution.

4.1. Experimental settings

Comparison partners: We implemented all the learning models SVM, S^{3}VM, TS^{3}VM and RK-TS^{3}VM in C++. To solve the quadratic programming problem for the RK-TS^{3}VM model as in Eq. (17), we used the optimization package in the IMSL Fortran Library.[1] While our proposed RK-TS^{3}VM model is the most accurate, it is also the most sophisticated and time-consuming. Other simpler models can perform sufficiently accurate with less training time in certain learning cases and thus can be desirable trade-off solutions.

Data streams: One synthetic data stream and three real-world data streams were used in our experiments. The synthetic data stream was used to assess the performance of our framework under different concept drifting scenarios, which are hard to capture from real-world streams. It was generated as an infinite sequence of $\{(x_i,y_i)_{i=1}^{+\infty}\}$, where $x_i \in \mathbb{R}^d$ is the feature vector and $y_i \in \{-1,+1\}$ is the class label. The feature values of x_i were generated by a uniform distribution between 0 and 1, and the classification boundary was controlled

$$\sum_{i=1}^{d} a_i x_i = a_0 \tag{30}$$

where a_i controls the decision boundaries.

For each example x_i, if $\sum_{i=1}^{d} a_i x_i \geq a_0$, it was labeled as $y_i = +1$; otherwise, $y_i = -1$. To simulate the labeling process, we randomly chose p percentage of examples and labeled them using Eq. (30). To simulate concept drifting, a_i was given a probability c, $0 \leq c \leq 1$, to evolve between a_i and $a_i+0.1$ with 10% probability to reverse the direction. Besides, we set a margin between the two classes. For the "$+1$" class, the boundary was set to $\sum_{i=0}^{d} a_i x_i + 0.05$, and for the "$-1$" class, the boundary was set to $\sum_{i=0}^{d} a_i x_i - 0.05$. To keep class distributions relatively balanced, we enforced the classification boundary around the central point of the feature space by setting $a_0 = \frac{1}{2}\sum_{i=1}^{d} a_i$. In order to make the decision surface nonlinearly separable, 3% noise was introduced to the stream by randomly flopping the class labels of the selected instances.

Three real-world streams were used in our experiments: wireless sensor stream, power supply stream, and intrusion detection stream. These data streams can be downloaded at www.cse.fau.edu/xqzhu/stream.html. The corresponding learning tasks have been discussed in our Introduction section.

4.2. Parameter study

Among the four learning models, SVM and S^{3}VM are not new. Thus, we performed parameter study on synthetic data for the TS^{3}VM and *RK* models under different experimental settings.

The four parameters of the TS^{3}VM model in Eq. (10) are C_1, C_2, C and C^*. From the multi-task learning perspective, C_1 and C_2 control the discrepancies between the global optimal boundary w and the local optimal boundary $w+v_1$ and $w+v_2$. If we assign a relative large value to C_1 (compared to C_2), the global optimal solution w will bias towards task 1, and vice versa. If we have $C_1 \gg C_2$, i.e., $C_1/C_2 > 10,000$, v_1 will approach to 0 and the classification boundary of task 1 becomes the global optimal boundary. The parameter C controls the penalty of the

[1] http://www.vni.com/products/imsl/

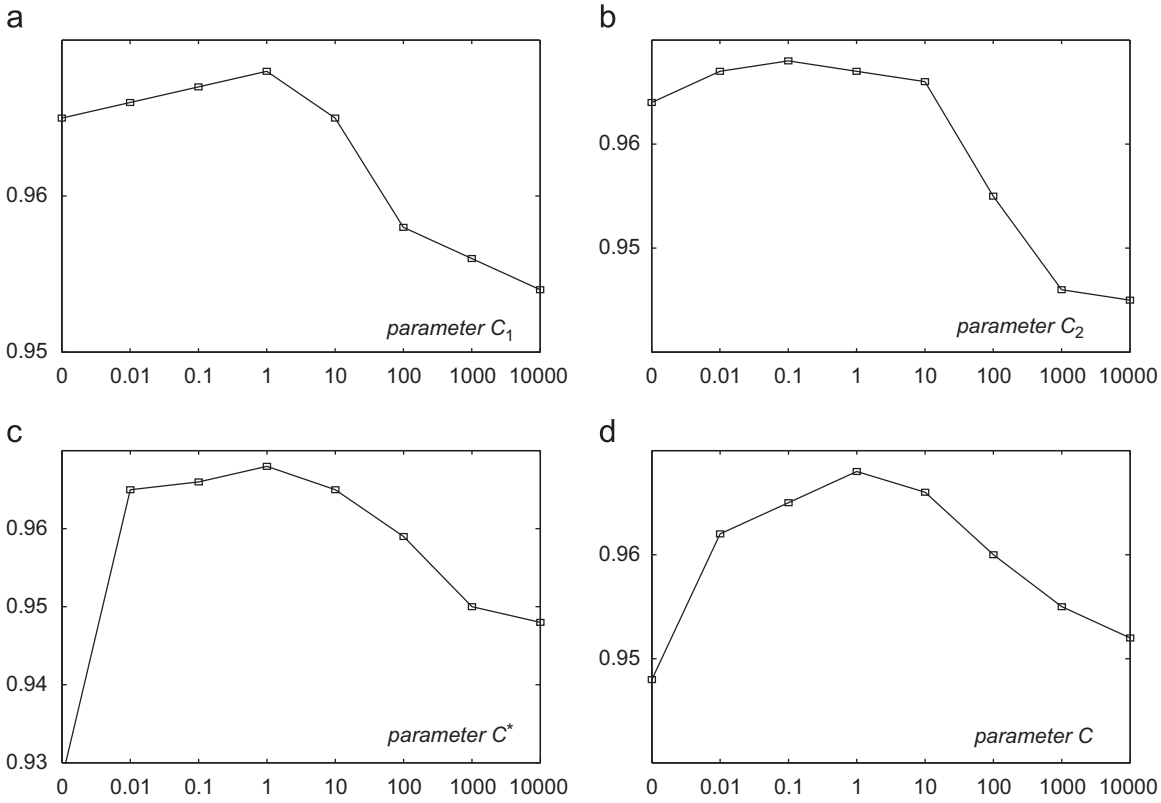


Fig. 6. Parameter study for TS^{3}VM. The y-axis denotes accuracy and the x-axis denotes parameter values for (a) C_1, (b) C_2, (c) C^* and (d) C, each varying from 0 to 10,000.

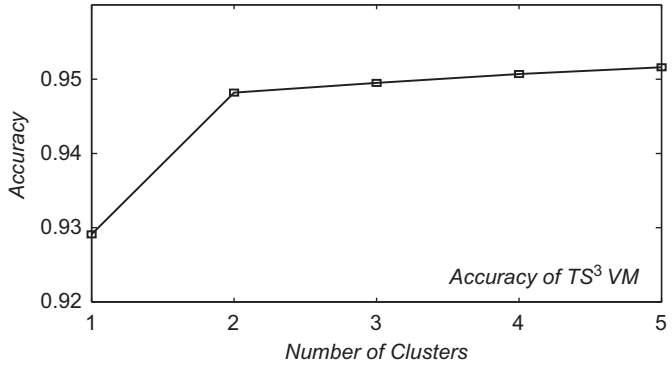


Fig. 7. Accuracy of TS^{3}VM with respect to different k values for the RK model. We can observe that when k increases from 1 to 2, the accuracy increases significantly. After that, the improvement becomes marginal.

misclassified examples in T_1 and T_2, and the last parameter C^* controls the penalty of the misclassified examples in T_3.

Fig. 6 reports the accuracy values (the y-axis) of TS^{3}VM with respect to different parameter values (the x-axis). All the results were averaged over 100 data chunks each containing 500 examples. The concept drifting probability c was set to 50% and the labeling percentage was set to $p=10\%$. From Fig. 6(a) and (b), we observe that increasing C_2 will result in a more noticeable performance deterioration than increasing C_1. This is consistent with our assumption that task 1 is supposed to comply with the same distribution as the target domain. The results in Fig. 6(a) and (b) also suggest that a reasonable setting for C_1 and C_2 is to ensure that $C_1/C_2=10$. From Fig. 6(c) we can observe that the performance of the TS^{3}VM model has a significant improvement when C^* increases from 0 (where the accuracy is about 0.93) to 1 (where the accuracy is above 0.96). That is to say, adding T_3 for training will significantly improve the accuracy. Based on these observations, in the following experiments, we set C_1 to 10 and the remaining parameters to 1.

For the RK model, the key parameter is k, i.e., the number of clusters in Algorithm 3. Intuitively, increasing k is equivalent to increasing the number of feature vectors, which is supposed to enhance the performance. Our experimental settings were as follows: the TS^{3}VM model was tested on 100 data chunks each containing 500 examples. The probability of concept evolution/change was 90% and only 5% examples were labeled. From Fig. 7 we can observe that when setting k to 2, there is a significant accuracy improvement (from 0.9291 to 0.9482). After that, the improvement becomes marginal (from 0.9482 to 0.9516). Therefore, in our following experiments, we set k to 2.

4.3. Model study

In this series of experiments, we studied the performances of the four learning models under different concept drifting probability c and labeling percentage l. We designed four experiments, each corresponding to a different combination of parameters c and l representing one of the four learning cases. All results were averaged over 100 continuous data chunks.

Case 1: In the first experiment, the concept drifting probability was set to a very low value of 10%, and the labeling percentage was set to a high value of 90%. The proportion of the four types of training examples can be calculated as in Fig. 8. Obviously, most training examples in this case are of Type I. The experiments with respect to different chunk sizes D are shown in Table 1. From the results we can observe that the SVM model performs almost as well as other three more sophisticated models in terms of accuracy. This is because T_1 examples dominate the training chunk, and we can achieve satisfactory results by using only T_1 for training. Furthermore, from Fig. 9, we can see that SVM is the

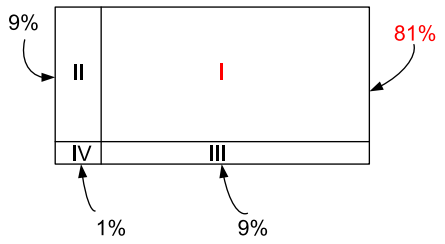


Fig. 8. Proportion of the four types of training examples in Case 1.

Table 1
Case 1 accuracy study.

Learning models	Chunk size		
	$n=100$	$n=500$	$n=1000$
SVM	$0.9877_{\pm 0.02}$	$0.9902_{\pm 0.02}$	$0.9945_{\pm 0.01}$
S^3VM	$0.9893_{\pm 0.02}$	$0.9902_{\pm 0.02}$	$0.9955_{\pm 0.01}$
TS^3VM	$0.9910_{\pm 0.02}$	$0.9922_{\pm 0.01}$	$0.9967_{\pm 0.01}$
RK-TS^3VM	$0.9922_{\pm 0.01}$	$0.9962_{\pm 0.01}$	$0.9973_{\pm 0.01}$

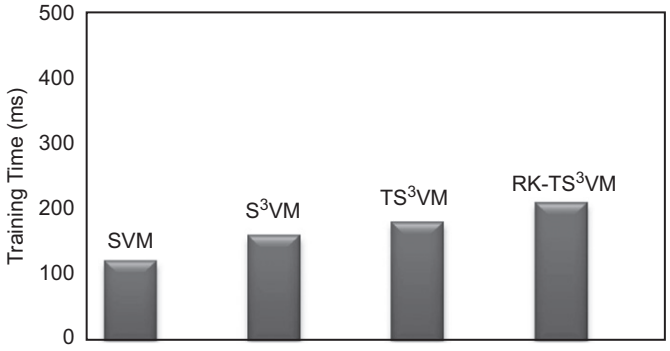


Fig. 9. Training time of the four models in Case 1.

Fig. 10. Proportion of the four types of training examples in Case 2.

Table 2
Case 2 accuracy study.

Learning models	Chunk size		
	$n=100$	$n=500$	$n=1000$
SVM	$0.8560_{\pm 0.07}$	$0.8792_{\pm 0.07}$	$0.8946_{\pm 0.04}$
S^3VM	$0.9293_{\pm 0.04}$	$0.9575_{\pm 0.04}$	$0.9621_{\pm 0.01}$
TS^3VM	$0.9297_{\pm 0.04}$	$0.9575_{\pm 0.04}$	$0.9622_{\pm 0.01}$
RK-TS^3VM	$0.9330_{\pm 0.01}$	$0.9610_{\pm 0.01}$	$0.9660_{\pm 0.01}$

most efficient model compared to its peers. Therefore, we can conclude that SVM is a better trade-off option of models when Type I examples dominate the training chunk (i.e., when concept drifting probability is low and labeling percentage is high).

Case 2: In the second experiment, both concept drifting probability and labeling percentage were set to a very low value of 10%. In this situation, Type III examples, as shown in Fig. 10, dominate the training chunk. The results with respect to different chunk sizes are presented in Table 2. We can observe that S^3VM outperforms SVM, and is almost as good as other two models in terms of accuracy. From Fig. 11 we can observe that S^3VM is more efficient than both TS^3VM and RK-TS^3VM. Therefore, we can conclude that S^3VM is a better trade-off option of models when Type III examples dominates the training chunk (i.e., both concept drifting probability and labeling percentage are low).

Case 3: In the third experiment, drifting probability was set to a high value of 70% and labeling percentage to a high value of 90%. The proportion of the four types training examples can be calculated as in Fig. 12. Obviously, most training examples in this case are of Type II. The experiments with respect to different chunk sizes are shown in Table 3. From the results we can observe that TS^3VM significantly outperforms other two models SVM and S^3VM, and is almost as good as RK-TS^3VM in terms of accuracy. This is because T_3 examples dominate the training examples, and TS^3VM learns from T_3, T_2 and T_1 simultaneously. Furthermore, from Fig. 13, we can see that TS^3VM is much more efficient than RK-TS^3VM. Therefore, we can conclude that TS^3VM is a better trade-off option of models when Type II examples dominate the training chunk (i.e., both concept drifting probability and labeling percentage are high).

Case 4: In the last experiment, concept drifting probability was set to a high value of 70% and labeling percentage was set to a low value of 10%. The proportion of the four types training examples is shown in Fig. 14. In this case, Type IV examples dominate the training chunk. The results with respect to different chunk sizes are shown in Table 4. We can observe that RK-TS^3VM significantly

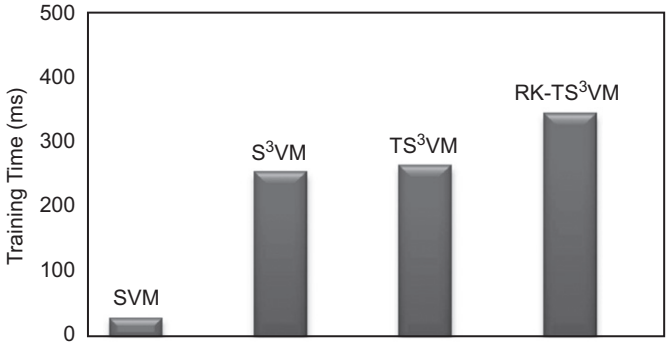


Fig. 11. Training time of the four models in Case 2.

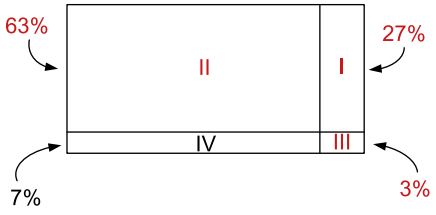


Fig. 12. Proportion of the four types of training examples in Case 3.

Table 3
Case 3 accuracy study.

Learning models	Chunk size		
	$n=100$	$n=500$	$n=1000$
SVM	$0.8311_{\pm 0.06}$	$0.8580_{\pm 0.04}$	$0.8602_{\pm 0.02}$
S^3VM	$0.8669_{\pm 0.04}$	$0.9072_{\pm 0.03}$	$0.9101_{\pm 0.02}$
TS^3VM	$0.9321_{\pm 0.04}$	$0.9503_{\pm 0.02}$	$0.9543_{\pm 0.02}$
RK-TS^3VM	$0.9330_{\pm 0.02}$	$0.9522_{\pm 0.02}$	$0.9545_{\pm 0.01}$

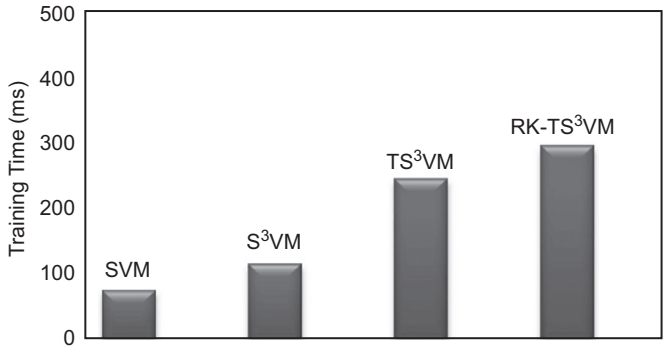


Fig. 13. Training time of the four models in Case 3.

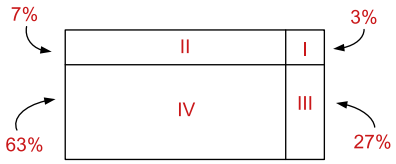


Fig. 14. Proportion of the four types of training examples in Case 4.

Table 4
Case 4 accuracy study.

Learning models	Chunk size		
	$n=100$	$n=500$	$n=1000$
SVM	$0.7950_{\pm 0.07}$	$0.8011_{\pm 0.05}$	$0.8129_{\pm 0.03}$
S^3VM	$0.8212_{\pm 0.05}$	$0.8310_{\pm 0.04}$	$0.8332_{\pm 0.03}$
TS^3VM	$0.8231_{\pm 0.05}$	$0.8242_{\pm 0.04}$	$0.8266_{\pm 0.02}$
RK-TS^3VM	$0.8843_{\pm 0.04}$	$0.8910_{\pm 0.04}$	$0.9018_{\pm 0.02}$

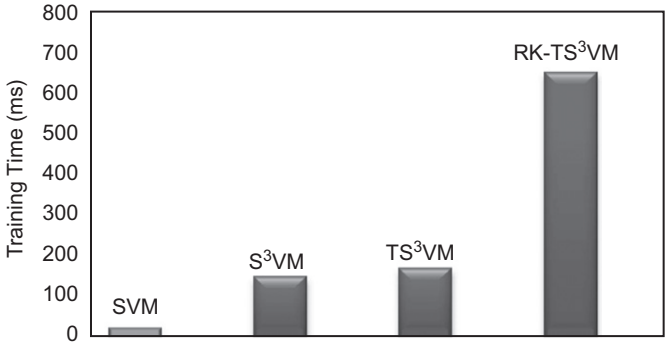


Fig. 15. Training time of the four models in Case 4.

outperforms all other models in terms of accuracy. On the other hand, we were unable to achieve accurate results by training only on T_1, T_2 and T_3. That is to say, in this case, we have to incorporate T_4 into training even if it is time-wise costly as shown in Fig. 15. The results validate our claim that learning under Case 4 is the most difficult, and we have to use the RK-TS^3VM model to gain satisfactory accuracy.

4.4. Experiments on real-world streams

In this series of experiments, we compared the four learning models on three real-world data streams: wireless sensor stream, power supply stream, and intrusion detection stream.

In order to decide the learning model, we need to identify the main type of examples in the training chunk. For this purpose, we need to detect the concept drifting probability c and decide the labeling rate l. While several concept change detection methods are available [16,20], for simplicity (we do not need very accurate estimates), we used prediction accuracy to approximate concept drifting probability. Specifically, we first assume that all the stream examples are labeled. If there is no concept drifting, the prediction accuracy will be close to 100%. Otherwise, the prediction error rate can be used to approximate the concept drifting probability. Fig. 16 shows the prediction accuracy over 50 continuous data chunks of the three data streams. The average accuracy values for the three streams are 84.32%, 91.07% and 99.13%, respectively. Thus, the concept drifting probability in the three streams are approximately 15%, 10% and 1%, respectively.

The labeling rate depends on the labeling experts. They were set to 10%, 10% and 90%, respectively, for the three streams. By doing so, in the wireless sensor and power supply streams, Type II examples dominate the training trunk. In the intrusion detection stream, Type I examples dominate the training chunk.

Fig. 17(a) shows chunk-by-chunk comparisons for the four learning models on the sensor stream. We can observe that SVM always has the worst prediction accuracy. The remaining three models performed similarly and better than SVM. Fig. 18(a) shows comparisons on time cost for the four models. We can observe that S^3VM is not as efficient as SVM, but better than the remaining two models. Considering both prediction accuracy and efficiency, we can conclude that S^3VM is the best trade-off option of models for classifying the sensor stream data. Similar results, as shown in Figs. 17(b) and 18(b), can also be observed for the power supply stream.

Figs. 18(c) and 17(c) show comparisons on the KDDCUP'99 intrusion detection stream. We can observe that SVM has the lowest time cost. On the other hand, SVM can achieve similar prediction accuracies to other three models over the continuous 50 data chunks. Therefore, we can conclude that SVM is the best trade-off option of models in this case.

Another important observation is that in all of the above experiments, the proposed RK-TS3VM model often had the highest prediction accuracy. However, due to the complexity of the model, it always has the highest time cost. Therefore, this model is preferred in accuracy-critical applications where the concept drifting probability is high and the labeling rate is low.

5. Related work

The work reported in this paper applies sophisticated machine learning models, such as transfer learning [9] and semi-supervised learning [27], to real-world data stream classification [40,42,33,2].

Two types of classification models have been proposed for data stream classification: incremental learning [10,11] and ensemble learning [35,32,39,41,36]. The former uses new data to update models trained from historical stream data. By doing so, the learning process scales to large data volumes as well as adapts to changing concepts. For example, Domingos [11] proposed a fast decision tree learner VFDT that incrementally builds Hoeffding trees from overwhelming volume of fast flowing data streams. An extended approach CVFDT [17] handles time changing and concept evolution streams. Other methods [25] also attempted to employ incremental learning to tackle the challenge of concept evolution.

Ensemble learning, on the other hand, trains a number of base models from a small portion of stream data (i.e., a data chunk), and combines all base models to form an ensemble classifier for prediction. Existing ensemble frameworks can be further

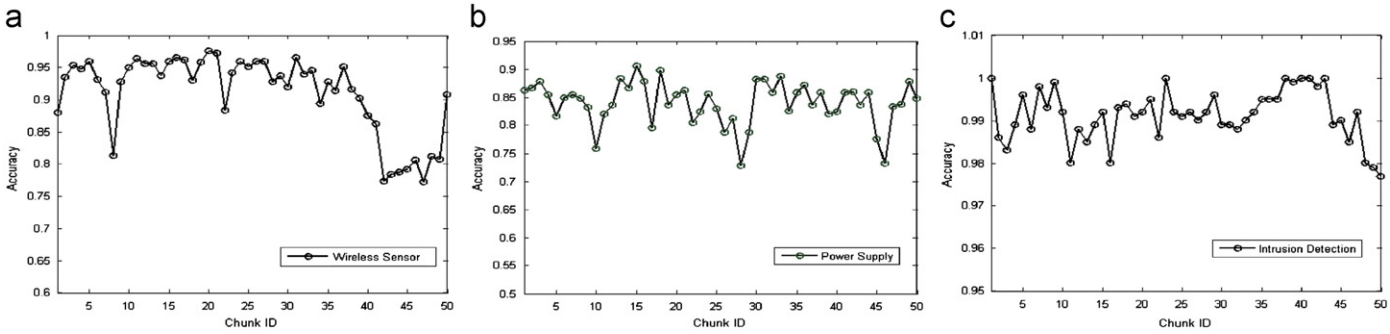


Fig. 16. Average accuracy over 50 data chunks on the three data streams. (a) Wireless sensor, (b) power supply and (c) intrusion detection.

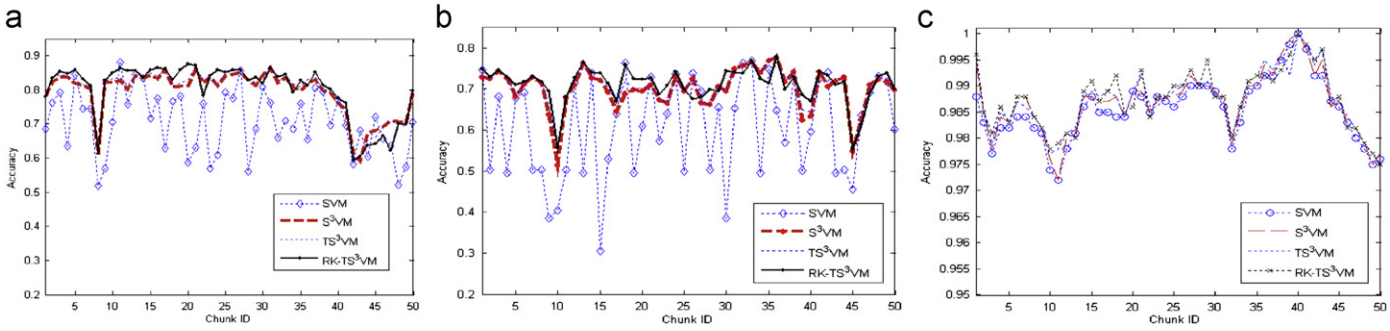


Fig. 17. Chunk by chunk comparisons of the first 50 data chunks on the three data streams. (a) Wireless sensor, (b) power supply and (c) intrusion detection.

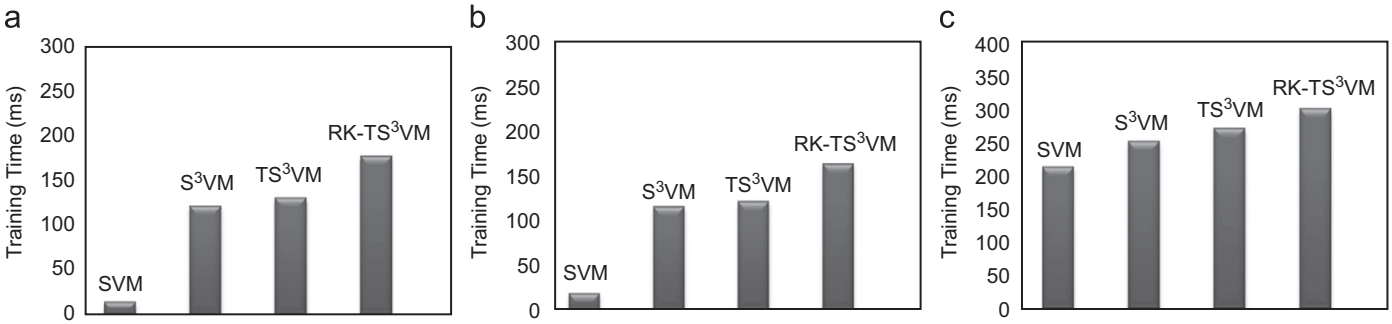


Fig. 18. Training time comparisons on the three data streams. (a) Wireless sensor, (b) power supply and (c) intrusion detection.

categorized into two paradigms: *horizontal ensemble* that builds base classifiers on different buffered chunks by using one single learning algorithm, and *vertical ensemble* that builds base classifiers on the most-recent buffered chunk by using various learning algorithms.

A large portion of ensemble methods belong to the horizontal ensemble paradigm. For example, Street [24] proposed a SEA algorithm that combines decision tree models using majority voting. Kolter [21] proposed an ensemble method by using weighted online learners to handle drifting concepts. Wang et al. [18] proposed an accuracy-weighted ensemble method that assigns each classifier a weight reversely proportional to its accuracy on the most recent data chunk. Yang et al. [28] proposed proactive learning where concepts (models) learnt from previous data chunks are used to foresee the best model to predict data in the current chunk. Zhu et al. [41] proposed an active learning framework that selectively labels examples for concept evolution data streams.

Some most recent ensemble methods, however, belong to the vertical ensemble paradigm. For example, Gao et al. [14] and Zhang et al. [34]. Without the prior knowledge that when and where concept evolution may occur, buffered data chunks may contain obsolete patterns that deteriorate the ensemble's performance. Based on this observation, they proposed a new approach that builds the ensemble only from the most recent data chunk by using different learning algorithms such as decision tree, SVMs, and logistic regression models.

Existing data stream classification models assume that *training examples are fully labeled*. However, labeling by experts is very expensive. For stream data with large, continuous volumes, this assumption would not hold. In other words, in practice, training data are often partially labeled.

Learning from unlabeled examples has been widely studied in the machine learning and data mining communities. For this purpose many semi-supervised learning models [19,6] have been proposed. Meanwhile, algorithms such as EM with generative mixture models [23], self-training [12], co-training [1], transductive Support Vector Machines [19,8], and graph-based [27] methods can be categorized as the semi-supervised learning category.

However, these methods only can be used in stationary databases. Very few studies consider data stream classification with unlabeled training examples. Previously, Masud et al. [22] proposed a micro-cluster-based method that clusters unlabeled examples into micro-clusters, which are combined with labeled examples to generate decision boundaries. Zhang et al. [31] proposed an ensemble-based

model that combines base classifiers built from labeled examples and clusters built from unlabeled examples. Zhu et al. [41] proposed an active learning-based model that minimizes labeling cost by selecting the most important examples for labeling. Different from these methods, we perform a systematic study and provide a comprehensive solution for classification of real-world data streams featuring different degrees of concept drifting and labeling percentage, targeting not only high predicting accuracy, but also improved learning efficiency.

6. Conclusions

Data stream classification for real-world applications has three major challenges: concept drifting, large volumes, and partial labeling. We have addressed these challenges and proposed a comprehensive learning framework that can efficiently learn accurate models from diverse training examples. In our framework, training examples are categorized into four types: labeled and from the target domain (Type I), labeled and from a similar domain (Type II), unlabeled and from the target domain (Type III) and unlabeled and from a similar domain (Type IV). Each type has its own learning priority. Then, based on the proportion and learning priority of the different types of training examples, four learning cases are considered and for each case, a different SVM-based learning model is applied. The learning models go from relatively simple (Cases 1 and 2) to fairly sophisticated (Cases 3 and 4). The framework gains in efficiency by learning simpler models whenever possible depending on the learning cases. Yet for each case, our framework makes sure the most informative examples are utilized and the learning model is sufficiently accurate. Thus, our framework also achieves high accuracy. Extensive experiments on both synthetic and real data streams have demonstrated the effectiveness and efficiency of our framework.

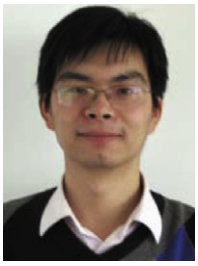
There are several interesting directions for future work. To improve efficiency of the TS^{3}VM model, we will study how to train linear TS^{3}VM model in linear time for high dimensional sparse data. We will also attempt to improve accuracy of the TS^{3}VM model by incorporating kernel functions.

Acknowledgment

This research was supported by the National Science Foundation of China (NSFC) Grants (61003167, 90718042, 71110107026), National High Technology Research and Development Program 863 Grant (2011AA010705), Texas Norman Hackerman Advanced Research Program Grant (003656-0035-2009), and CAS/SAFEA International Partnership Program for Creative Research Teams (70921061).

References

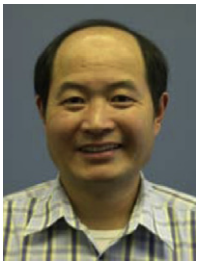
[1] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: The Workshop on Computational Learning Theory, 1998, pp. 92–100.
[2] C. Aggarwal, Data Streams: Models and Algorithms, Springer, 2006.
[3] C. Aggarwal, J. Han, J. Wang, Y. Philip, A framework for clustering evolving data streams, in: Proceedings of the VLDB, 2003, pp. 81–92.
[4] A. Asuncion, D. Newman, UCI Machine Learning Repository, Irvine, CA. University of California, School of Information and Computer Sciences, Irvine, 〈http://archive.ics.uci.edu/ml〉, 2007.
[5] B. Brian, B. Shivnath, D. Mayur, M. Rajeev, W. Jennifer, Models and issues in data stream systems, in: Proceedings of the PODS, 2002, pp. 1–16.
[6] O. Chapelle, B. Scholkopf, A. Zien, Semi-Supervised Learning, MIT Press, Cambridge, MA, 2006.
[7] O. Chapelle, V. Sindhwani, S. Keerthi, Optimization techniques for semi-supervised support vector machines, J. Mach. Learn. Res. 9 (2008) 203–233.
[8] R. Collobert, F. Sinz, J. Weston, Large scale transductive SVMS, J. Mach. Learn. Res. 7 (2006) 1687–1712.
[9] W. Dai, Q. Yang, G. Xue, Y. Yu, Boosting for transfer learning, in: Proceedings of the ICML, 2007, pp. 193–200.
[10] C. Domeniconi, D. Gunopulos, Incremental support vector machine construction, in: Proceedings of the ICDM, 2001, pp. 589–592.
[11] P. Domingos, G. Hulten, Mining high-speed data streams, in: Proceedings of the KDD, 2000, pp. 71–80.
[12] D. Yarowsky, Unsupervised word sense disambiguation rivaling supervised methods, in: Proceedings of the ACL, 1995, pp. 189–196.
[13] T. Evgeniou, M. Pontil, Regularized multi-task learning, in: Proceedings of the KDD, 2004, pp. 109–117.
[14] J. Gao, W. Fan, J. Han, On appropriate assumptions to mine data streams: analysis and practice, in: Proceedings of the ICDM, 2007, pp. 143–152.
[15] J. Han, H. Cheng, D. Xin, X. Yan, Frequent pattern mining: current status and future directions, Data Mining and Knowledge Discovery (DMKD) 15 (13–15) (2007) 55–86.
[16] S. Ho, A martingale framework for concept change detection in time-varying data streams, in: Proceedings of the ICML, 2005, pp. 321–328.
[17] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of the KDD, 2001, pp 97–106.
[18] H. Wang, W. Fan, P. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the KDD, 2003, pp. 226–235.
[19] T. Joachims, Transductive inference for text classification using support vector machines, in: Proceedings of the ICML, 1999, pp. 200–209.
[20] D. Kifer, J.G.S. Ben-David, Detecting change in data streams, in: Proceedings of the VLDB, 2004, pp. 180-191.
[21] J. Kolter, M. Maloof, Using additive expert ensembles to cope with concept drift, in: Proceedings of the ICML, 2005, pp. 449–456.
[22] 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: Proceedings of the ICDM, 2008, pp. 339–348.
[23] K. Nigam, R. Ghani, Analyzing the effectiveness and applicability of co-training, in: Proceedings of the CIKM, 2000, pp. 86–93.
[24] W.N. Street, Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in: Proceedings of the KDD, 2001, pp. 377–382.
[25] N. Syed, H. Liu, K. Sung, Handling concept drifts in incremental learning with support vector machines, in: Proceedings of the KDD, 1999, pp. 317–321.
[26] A. Tsymbal, The problem of concept drift: definitions and related work. Available online: 〈http://www.scss.tcd.ie/publications/tech-reports/reports.04/TCD-CS-2004-15.pdf〉.
[27] X. Zhu, Semi-supervised learning literature survey, Technical Report 1530, University of Wisconsin-Madison, 2005.
[28] Y. Yang, X. Wu, X. Zhu, Combining proactive and reactive predictions of data streams, in: Proceedings of the KDD, 2005, pp. 710–715.
[29] A. Yuille, A. Rangarajan, The concave-convex procedure, in: Proceedings of the NIPS, 2001, pp. 1033–1040.
[30] P. Zhang, J. Li, P. Wang, B. Gao, X. Zhu, L. Guo, Enabling fast prediction for ensemble models on data streams, in: Proceedings of the KDD, 2011, pp. 177–185.
[31] P. Zhang, X. Zhu, L. Guo, Mining data streams with labeled and unlabeled training examples, in: Proceedings of the ICDM, 2009, pp. 627–636.
[32] P. Zhang, X. Zhu, Y. Shi, L. Guo, X. Wu, Robust ensemble learning for mining noisy data streams, Dec. Support Syst. 50 (2) (2011) 469–479.
[33] P. Zhang, X. Zhu, J. Tan, L. Guo, Skif: a data imputation framework for concept drifting data streams, in: Proceedings of the CIKM, 2010, pp. 1869–1872.
[34] P. Zhang, X. Zhu, Y. Shi, Categorizing and mining concept drifting data streams, in: Proceedings of the KDD, 2008, pp. 812–820.
[35] P. Zhang, X. Zhu, J. Tan, L. Guo, Classifier and cluster ensembles for mining concept drifting data streams, in: Proceedings of the ICDM, 2010, pp. 1175–1180.
[36] P. Zhang, X. Zhu, X. Wu, Y. Shi, An aggregate ensemble for mining concept drifting data streams with noise, in: Proceedings of the PAKDD, 2009, pp. 1021–1029.
[37] X. Zhu, Stream data mining repository 〈www.cse.fau.edu/xqzhu/〉, 2009.
[38] X. Zhu, R. Jin, Multiple information sources cooperative learning, in: Proceedings of the IJCAI, 2007, pp. 1369–1375.
[39] X. Zhu, X. Wu, C. Zhang, Vague one-class learning for data streams, in: Proceedings of the ICDM, 2009, pp. 657–666.
[40] X. Zhu, P. Zhang, X. Lin, Y. Shi, Active learning from data streams, in: Proceedings of the ICDM, 2007, pp. 757–762.
[41] X. Zhu, P. Zhang, X. Lin, Y. Shi, Active learning from stream data using optimal weight classifier ensemble, IEEE Trans. Syst. Man Cybernet. Part B 40 (4) (2010) 1–15.
[42] X. Zhu, P. Zhang, X. Wu, D. He, C. Zhang, Y. Shi, Cleansing noisy data streams, in: Proceedings of the ICDM, 2008, pp. 1139–1144.



Peng Zhang is an Assistant Professor with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing (China). He received his PhD (2009) in Computer Science from the Graduate University of the Chinese Academy of Sciences, Beijing, China. His research interests include data stream mining and information security.

Byron J. Gao received PhD and BSc in Computer Science from Simon Fraser University, Canada, in 2007 and 2003, respectively. He was a postdoctoral fellow at the University of Wisconsin-Madison before joining Texas State University-San Marcos in 2008. His research spans several related fields including data mining, databases, information retrieval, and bioinformatics.

Ping Liu is an Assistant Professor with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing (China). Her research interests include string matching and information security.



Yong Shi is the Distinguished Professor of Information Technology, College of Information Science and Technology, Peter Kiewit Institute, University of Nebraska, USA. He is also the Executive Deputy Director of the Fictitious Economy and Data Science Research Center, Chinese Academy of Sciences, Beijing, China. He is the Editor-in-Chief of International Journal of Information Technology and Decision Making (SCI), an Area Editor of International Journal of Operations and Quantitative Management, a member of Editorial Board for a number of academic journals, including International Journal of Data Mining and Business Intelligence.

Li Guo is the Director of the Information Security Research Center, Institute of Computing Technology, Chinese Academy of Sciences. Her research interests include data stream management and information security.