2009 Ninth IEEE International Conference on Data Mining

Mining Data Streams with Labeled and Unlabeled Training Examples*

Peng Zhang⋆, Xingquan Zhu†‡, and Li Guo⋆
⋆Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China
†QCIS Center, Faculty of Eng. & IT, Univ. of Technology, Sydney, NSW 2007, Australia
‡FEDS Center, Graduate University, Chinese Academy of Sciences, Beijing, 100190, China
zhangpeng@ict.ac.cn; xqzhu@it.uts.edu.au; guoli@ict.ac.cn

Abstract

In this paper, we propose a framework to build prediction models from data streams which contain both labeled and unlabeled examples. We argue that due to the increasing data collection ability but limited resources for labeling, stream data collected at hand may only have a small number of labeled examples, whereas a large portion of data remain unlabeled but can be beneficial for learning. Unleashing the full potential of the unlabeled instances for stream data mining is, however, a significant challenge, consider that even fully labeled data streams may suffer from the concept drifting, and inappropriate uses of the unlabeled samples may only make the problem even worse. To build prediction models, we first categorize the stream data into four different categories, each of which corresponds to the situation where concept drifting may or may not exist in the labeled and unlabeled data. After that, we propose a relational k-means based transfer semi-supervised SVM learning framework (RK-TS^{3}VM), which intends to leverage labeled and unlabeled samples to build prediction models. Experimental results and comparisons on both synthetic and real-world data streams demonstrate that the proposed framework is able to help build prediction models more accurate than other simple approaches can offer.

1. Introduction

Due to the increasing availability of data recording and transmission techniques, applications such as data warehousing and sensor networking are facing dynamic data streams instead of static data sets. Such dynamic natures of data streams raise two fundamental challenges for data mining research: (1) large and continuous data volumes and (2) evolving or drifting of the concepts (or patterns) underneath the data. Motivated by the two challenges, a lot of work exists for association rule mining [8], clustering [1], and building prediction models [4, 7, 12, 16, 15].

From classification perspective, two sets of solutions exist for building prediction models from data streams: incremental learning [4] and ensemble learning [12, 9, 16, 15]. The former uses new data to update the models trained from historical stream data so the learning process is able to scale up to large data volumes as well as adapt to the changing concepts. 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. For either approach, one presumption is that the underlying stream data must be fully labeled. In practice, labeling training examples is a costly procedure which requires full attention and intensive investigation on the instances. For stream data with continuous data volumes, this becomes a huge burden or even infeasible to realize.

Consider a bank fraud detection center which audits daily credit card transactions to determine high risk or fraudulent transactions. In order to build an automatic fraud prediction model, it is essential to let banking experts provide (label) a number of fraud transactions as training samples. The labeling ability the experts can offer is rather limited and probably no bank is willing to pay experts to manually label every single transaction, assuming that 10,000 transactions may arrive on a daily basis. A more practical solution is to label a small portion, say 1% or 5%, of transactions, but keep the rest of the transactions unlabeled. Due to the missing of the labels, none of the existing stream mining algorithms is able to utilize such unlabeled data to help train fraud prediction models.

Indeed, just because stream data have large volumes it does not necessarily mean that mining algorithms are able to utilize them all. Hiding behind the large/continouse data volumes of the stream data is the reality that only a small portion of samples are actually labeled. Unfortunately, al-

*This research has been supported by the National Science Foundation of China (NSFC) under Grant No. 60674109, and the National Basic Research Program of China (973 Program) under Grant No. 2007CB311100

1550-4786/09 $26.00 © 2009 Crown Copyright
DOI 10.1109/ICDM.2009.76

though many methods exist for mining data streams, very few attempts have been made to address unlabeled sample issue. For static data sets, active learning and semi-supervised learning are two common approaches to solve the problem, where the former intends to reduce the labeling cost and the latter directly utilizes the unlabeled samples to boost the learning. In [16], Zhu et al. proposed an active learning framework to minimize the labeling cost for stream data, and the main goal is to select the most important sample for labeling such that the classifiers built from the stream data can gain maximum prediction accuracies. A recent work [10] also addresses the limited training example problem for stream data by using semi-supervised clustering techniques. But no work currently exists to help understanding the impact of the unlabeled samples on the concept drifting of the data streams.

To build prediction models from stream data, one important task is to identify and emphasize on historical examples which share identical or similar distributions to the test data (usually the stream data at the next time point). This problem, in practice, is difficult to solve given that we may not have any prior knowledge on the test data and we also do not know when and where the concept drifting may occur [6]. Consequently, most stream mining methods take the assumption that data which are temporally close are also relevant to each other at the concept level. So if we can separate data streams into chunks with the assumption that instances within a chunk share identical distributions (e.g., no concept driting), the most recent data chunk (*up-to-date chunk*) can be used to train classification models to predict instances yet to arrive (i.e, *yet-to-come chunk*) [12, 9, 15]. Although this assumption is popularly accepted in data mining research, for data stream with both labeled and unlabeled samples (referred to as *mixed stream* in this paper), samples within one data chunk may not share an identical distribution. Consider that even fully labeled stream data are suffering from the concept drifting problem, the existence of unlabeled samples only makes the learning from data stream even more challenging. Answers to the following three fundamental questions are needed before we can devise effective algorithms for mixed data streams.

- What are the typical concept drifting scenarios for data streams with unlabeled samples?
- How to categorize the mixed stream data in order to reveal the genuine concept drifting underneath the data?
- How to devise effective solutions to handle different categories of stream data, so the whole mining framework can be effective for mixed data streams?

In this paper, we report our research efforts in resolving the above three concerns. In short, we revisit the concept drifting scenarios by categorizing instances in mixed data streams into four types: (1) labeled (Type I) and unlabeled (Type III) examples which have the ***same*** distributions as the yet-to-come data chunk; and (2) labeled (Type II) and unlabeled (Type IV) examples which have ***similar*** distributions to the yet-to-come data chunk. Based on the categorizations, we propose a new Transfer Semi-Supervised SVM (TS^3VM) model to learn from Types I, II, and III instances. A relational k-means (RK) based model is also proposed for learning from Type IV examples. By combining TS^3VM and RK together, we propose a RK-TS^3VM learning framework which is able to fulfill the learning from the mixed data streams.

The remainder of the paper is structured as follows. Section 2 discusses data categorization for mixed data streams. We derive a new Transfer Semi-Supervised SVM (TS^3VM) model and a relational k-means (RK) model in Section 3. Section 4 formulates the complete RK-TS^3VM learning framework. Experimental results are reported in Section 5, and we conclude this paper in Section 6.

2. Categorizing sample types for mixed data streams

Assume stream data arrive chunk by chunk with each chunk containing a number of labeled and unlabeled instances. At any specific time, the yet-to-come data chunk is dedicated as the *target domain* and the mining purpose is to build prediction model to accurately classify instances in the yet-to-come data chunk. For this purpose, the instances in the up-to-date chunk can be used to train prediction models, under the constraint that only a small portion of instances in the up-to-date chunk are labeled. To accurate describe the concept drifting scenarios for mixed data streams, we can category the examples in the up-to-date chunk into following four types based on the samples' distributions with respect to the distributions of the target domain (i.e, the yet-to-come chunk): (1) labeled and same distribution examples (*Type I Examples*), labeled and similar distribution examples (*Type II Examples*), unlabeled and same distribution examples (*Type III Examples*), and unlabeled and similar distribution examples (*Type IV Examples*).

A conceptual view of the above four categorizations is shown in Fig. 1. Assume the yet-to-come data chunk (test chunk) is dedicated as the target domain, blue solid circles denote the labeled and same distribution examples (Type I), red solid circles denote the labeled and similar distribution examples (Type II), blue hollow circles denote the unlabeled and same distribution examples (Type III), and red hollow circles denote the unlabeled and similar distribution examples (Type IV). Due to the temporal correlations [12] of the concepts, Types I and III data usually locate at the tail of the training chunk, which are close to the yet-to-come chunk. Types II and IV data usually locate at the head of

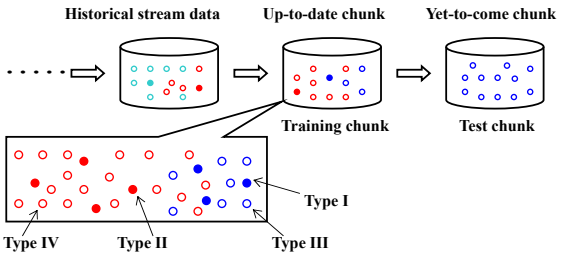


Figure 1. A conceptual view of the example type categorization for mixed data streams.

the training chunk, which are relatively far away from the yet-to-come chunk.

After defining the four types of examples for mixed data streams, a following question is how to identify their sizes and locations in the up-to-date chunk (training chunk). Intuitively, the percentage of labeled examples depends on the labeling speed the experts can offer and the number of the same distribution examples depends on the concept drifting rate. As shown in Fig. 1, Types I and III examples usually locate at the tail of the up-to-date chunk while Types II and IV examples locate at the head of the up-to-date chunk. Consider a data stream which flows at a speed of n examples per second, and its concept drifts with a possibility of c, where $0 \leq c \leq 1$. At each time stamp, a training chunk $D = \{x_1, \cdots, x_n\}$ is buffered and labeled by experts with a speed of l examples ($l \ll n$) per second. The number of the same distribution examples will have a reverse proportion to the concept drifting rate c with a constant coefficient γ, so we can estimate that about $\gamma \cdot c^{-1} \cdot n$ examples in chunk D have the same distributions as the test examples while the remaining $(1 - \gamma \cdot c^{-1}) \cdot n$ examples will have similar distributions as the test examples. Denote the size of the four types of examples as L_1, L_2, L_3, and L_4 respectively, we can estimate that their values are given in Eq. (1).

$$\begin{aligned} L_1 = \gamma \cdot c^{-1} \cdot l; \quad L_2 = (1 - \gamma \cdot c^{-1}) \cdot l \\ L_3 = \gamma \cdot c^{-1} \cdot (n - l); \quad L_4 = (1 - \gamma \cdot c^{-1}) \cdot (n - l) \end{aligned} \tag{1}$$

Ideally, we can exactly identify the sizes of the four types of examples by calculating Eq.(1). In practice, we may not have any prior knowledge on the concept drifting rate c, so the above two equations can't be used directly. In this case, an alternative solution is to assign an empirical small value to c. In our experiments in Section 5, the 10% examples at the tail of the training chunk are assigned as the same distribution examples. In following sections we will propose detailed solutions for learning from the four types of data.

3. TS^{3}VM and RK learning models

In this section, we introduce two learning models to handle the four types of examples in the mixed stream. More specifically, because Types I & III examples are assumed to have the same distributions as the target domain, generic semi-supervised learning model [14, 3, 2] (we use SVM in this paper) can be used to directly train classifiers from Types I & III data. For Type II samples, although they are assumed to have different distributions from the target domain, because they are labeled, we can employ transfer learning principle [5] to build models from Type II samples. Consequently, our first objective is to devise a transfer semi-supervised learning based model (TS^{3}VM) to train classifiers from Types I, II, & III examples. For type IV examples, because they are unlabeled and have different distributions from the target domain, we will propose a relational k-means based learning model (RK) to handle such data. The final learning framework (RK-TS^{3}VM) (discussed in Section 4) will then combine the strength of TS^{3}VM and RK to train prediction models from the mixed stream.

For ease of understanding, we will use bi-class classification as an examples to articulate our algorithm design. The four types of examples are simplified as follows: Type I data are denoted by $T_1 = (x_1, y_1), \ldots, (x_{L_1}, y_{L_1})$, where $x_i \in \mathbb{R}^d$, d is the dimension, $y_i \in \{-1, +1\}$; Type II data are denoted by $T_2 = \{(x_{L_1+1}, y_{L_1+1}), \ldots, (x_L, y_L)\}$, where $L = L_1 + L_2$; Type III data (U unlabeled examples) are denoted by $T_3 = \{x_{L+1}, \ldots, x_{L+U}\}$; and Type IV data ($N$ unlabeled examples) is denoted by $T_4 = \{x_{L+U+1}, \ldots, x_{L+U+N}\}$.

3.1 TS^{3}VM Learning Model

Intuitively, the TS^{3}VM model can be formulated by sequentially incorporating instances in T_1, T_2, and T_3 for learning. More specifically, we first formulate a generic SVM model by taking Type I examples into consideration. After that, we can formulate a transfer SVM model by taking Type II examples into consideration. Finally, we can include Type III examples and formulate the TS^{3}VM model.

3.1.1 Learning from Type I Examples

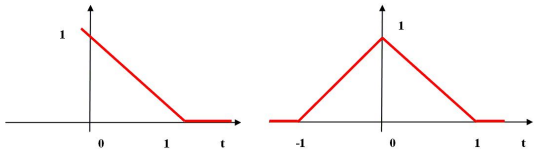
To learn from $T_1 = \{(x_1, y_1), \ldots, (x_{L_1}, y_{L_1})\}$, a generic SVM model can be trained by maximizing the margin distance between classes while minimizing the error rates as given in Eq. (2), where w is the projection direction, b is the classification boundary, ξ_i is x_i's error distance to b, and parameter C denotes the penalty of the examples inside the margin.

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

The SVM model given in Eq. (2) is a constrained convex optimization problem. To simplify the expression, the Hinge Loss function [3] in Fig. 2 can be used to transform

Eq. (2) into an unconstrained convex optimization problem as defined by Eq. (3), where $\theta = (w, b)$ and $f_\theta(x) = (wx+b)$. The similar approaches have been commonly used to formulate the semi-supervised SVM model [2].

$$min_\theta \qquad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L_1} H(y_i f_\theta(x_i)) \tag{3}$$



(a) Hinge Loss Function (b) Symmetric Hinge Loss Function

Figure 2. 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: $min\ \xi,\ s.t. : \xi \geq 0,\ \xi \geq 1-t$.

3.1.2 Learning from Types I & II Examples

Existing research has shown that SVM is capable of identifying optimal classification boundaries given sufficient number of training samples. In practice, the number of samples in T_1 may be very limited, which may leave classical SVM incapable of learning the optimal classification boundaries. To overcome such deficiency, transfer learning can use labeled samples in T_2 to refine the classification boundary by transferring the knowledge from T_2 to T_1. An effective way to transfer the knowledge between T_2 and T_1 is to take the problem as a multi-task learning procedure [5]. A common two-task learning SVM model on T_1 and T_2 can be formulated as in Eq. (4), where parameters C_1 and C_2 are the penalties on each task, 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 task 1 and $w + v_2$ for task 2).

$$\begin{aligned} min \quad & \frac{1}{2}\|w\|^2 + C_1\|v_1\|^2 + C_2\|v_2\|^2 + C\sum_{i=1}^{L}\xi_i \\ s.t.: \quad & y_i((w+v_1)x_i + b) \geq 1-\xi_i, \quad 1 \leq i \leq L_1 \\ & y_i((w+v_2)x_i + b) \geq 1-\xi_i, \quad L_1+1 \leq i \leq L \\ & \xi_i \geq 0, \quad 1 \leq i \leq L \end{aligned} \tag{4}$$

In Eq. (4), parameters C_1 and C_2 controls the preference of the two tasks. If $C_1 > C_2$, then it prefers task 1 to task 2; otherwise, if $C_1 < C_2$, it prefers task 2 to task 1 (Our experimental results in Section 5 will further study the relationship between C_1 and C_2). By using the Hinge loss function, Eq. (4) can be transformed into an unconstrained form in Eq.(5) where $\theta = (w, v_1, v_2, b)$, $f_\theta(x) = (w+v_1)x+b$ for task 1 while $f_\theta(x) = (w+v_2)x+b$ for task 2.

$$min_\theta \qquad \frac{1}{2}\|w\|^2 + C_1\|V_1\|^2 + C_2\|V_2\|^2 + C\sum_{i=1}^{L} H(y_i f_\theta(x_i)) \tag{5}$$

3.1.3 Learning from Types I, II, & III Examples

In addition to T_1 and T_2, learning on T_3 may further improve the performance in the reason that (i) labeled samples in T_1 and T_2 are only a small percentage of the whole training examples; and (ii) T_3 contains a relatively large number of examples that come from the same distribution as the test examples, which can greatly help in differentiating the genuine classification boundaries. In past several years, learning from a large number of unlabeled examples has been extensively studied from semi-supervised learning perspective. An effective way to train a semi-supervised learning model is to find a classification boundary that achieves a maximum margin not only between labeled examples, but also unlabeled examples, i.e., semi-supervised SVM [2] adds an extra term $C^* \sum_{i=L+1}^{L+U} H(|f_\theta(x_i)|)$ to penalty the misclassification of unlabeled examples which locate inside the margin as shown in Eq. (6),

$$min_\theta \qquad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L} H(y_i f_\theta(x_i)) + C^*\sum_{i=L+1}^{L+U} H(|f_\theta(x_i)|) \tag{6}$$

Thus, by adding the last term of Eq. (6) (which is used to learn from T_3) onto Eq. (5) (which is used to learn from T_1 and T_2), We finally derive the objective function of TS^{3}VM in Eq. (7), where $\theta = (w, v_1, v_2, b)$, and $f_\theta(x_i) = (w+v_1)x_i+b$ for $1 \leq i \leq L_1$, $f_\theta(x_i) = (w+v_2)x_i + b$ for $L_1+1 \leq i \leq L$, $f_\theta(x_i) = wx_i + b$ for $L+1 \leq i \leq L+U$.

$$\begin{aligned} min_\theta \quad & \frac{1}{2}\|w\|^2 + C_1\|v_1\|^2 + C_2\|v_2\|^2 \\ & + C\sum_{i=1}^{L} H(y_i f_\theta(x_i)) + C^*\sum_{i=L+1}^{L+U} H(|f_\theta(x_i)|) \end{aligned} \tag{7}$$

Balance Constrain A possible difficulty of the TS^{3}VM model is that all unlabeled examples in T_3 may be classified into one class with a very large margin, which may lead to poor performance. To solve the problem, an additional balance constraint should be added to ensure that unlabeled examples in T_3 should be assigned into both classes. In the case that we don't have any prior knowledge about the class ratio in T_3, a reasonable way [3] is to estimate its class ratio from T_1 and T_2 as in Eq.(8),

$$\frac{1}{U}\sum_{i=L+1}^{L+U} f_\theta(x_i) = \frac{1}{L}\sum_{i=1}^{L} y_i \tag{8}$$

Thus by taking account of the balance constrain, we derive the TS^{3}VM model in Eq.(9),

$$\begin{aligned} min_\theta \quad & \frac{1}{2}\|w\|^2 + C_1\|v_1\|^2 + C_2\|v_2\|^2 \\ & + C\sum_{i=1}^{L} H(y_i f_\theta(x_i)) + C^*\sum_{i=L+1}^{L+U} H(|f_\theta(x_i)|) \\ s.t.: \quad & \frac{1}{U}\sum_{i=L+1}^{L+U} f_\theta(x_i) = \frac{1}{L}\sum_{i=1}^{L} y_i \end{aligned} \tag{9}$$

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

3.1.4 Solution to the TS^{3}VM Objective Function

As shown in Eq. (9), the objective function of TS^{3}VM is a non-convex optimization problem, which is difficult to find global minima especially for large scale problems. To overcome this difficulty, we propose to solve this non-convex problem by using Concave-Convex Procedure (CCCP) which was developed by the optimization community in the last few decades [13, 3, 2]. CCCP method 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 problem. Algorithm 1 describes the algorithm in detail.

Algorithm 1 CCCP Algorithm

Require: the objective function $J(\theta)$
Get the 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 TS^{3}VM 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 analysis 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 in Eq.(10),

$$J(z) = C^* H(|z|) \tag{10}$$

Eq.(10) is a non-convex function, which can be spitted into a convex part and a concave part as in Eq.(11),

$$J(z) = C^* H(|z|) = \underbrace{C^* max(0, 1 - |z|) + C^*|z|}_{J_{vex}(t)} \underbrace{-C^*|z|}_{J_{cav}(t)} \tag{11}$$

According to Algorithm 1, the next iterative point can be calculated by the approximation of the concave part J_{cav} as shown in Eq.(12),

$$\frac{\partial J_{cav}(z)}{\partial z} \cdot z = \begin{cases} C^* z, & z < 0 \\ -C^* z, & z \geq 0 \end{cases} \tag{12}$$

and then minimizing Eq.(13),

$$J(z) = C^* \cdot max(0, 1 - |z|) + C^*|z| + \frac{\partial J_{cav}(z)}{\partial z} \cdot z \tag{13}$$

If at the current iteration $z < 0$, then to the next iteration, the effective loss on this point can be denoted as $L(z, -1)$ in Eq.(14):

$$L(z, -1) = C^* max(0, 1 - |z|) + C^*|z| + C^* z = \begin{cases} 2C\ z,^* & z \geq 1 \\ C^*(1 + z), & |z| < 1 \\ 0, & z \leq -1 \end{cases} \tag{14}$$

On the contrast, if $z \geq 0$, then to the next iteration, the effective loss on this point can be denoted as $L(z, +1)$ in Eq.(15),

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

By doing so, at each iteration, when taking all the $z_i = f_\theta(x_i)$ into considering, solving TS^{3}VM model is equivalent to solving Eq.(16) under the balance constrain Eq.(8),

$$min_\theta \qquad \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{L} H(y_i f_\theta(x_i)) + \sum_{i=L+1}^{L+U} L(f_\theta(x_i), y_i) \tag{16}$$

where y_i $(L + 1 \leq i \leq L + U)$ is the class label of the corresponding x_i which has been assigned at the previous iteration. If $y_i < 0$, then Eq.(14), will be used to calculate the loss function, else Eq.(15), will be used to calculate the loss function. The detailed description of solving TS^{3}VM is given in Algorithm 2.

Algorithm 2 TS^{3}VM 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. (4), and get the initial point $\theta_0 = (w_0, v_{10}, v_{20}, b_0)$
repeat
$y_i \leftarrow sgn(wx_i + b)$, $\forall L + 1 \leq i \leq L + U$
$\theta \leftarrow$ Calculate Eq.(16) under the balance constraint Eq.(8)
until y_i remains unchanged, $\forall L + 1 \leq i \leq L + U$
return $f(x) = sgn(wx + b)$

Kernel Trick The TS^{3}VM model in Eq.(9) can be only used for linear classification. To identify non-linear classification boundaries, we can incorporate the kernel function into TS^{3}VM by simply replacing the quadratic terms with a form of the sum of the mapped examples $\sum_{i=1}^{L+N} \beta_i \phi(x_i)$, where $\phi(\cdot)$ denotes a high dimensional feature mapping function [11]. Thus the optimization problem Eq.(9)) shifts to a new optimization problem with β_i as the variables.

3.2 RK Learning Model

Learning from T_4 is more challenging than learning from T_1, T_2, and T_3 because examples in T_4 are unlabeled and have different distributions from the target domain. In this section, we introduce a relational k-means (RK) [17] based learning model which aims to construct some new features to the labeled examples by using information extracted from unlabeled instances in T_4. An example of the RK learning is shown in Fig. 3, where instances in T_4 are fist clustered into a number of k clusters, $G_1, \cdots, G_k$ based on a relational matrix built between T_1 and T_4. After that, k new features $f(x_i, G_\tau)$ $(\tau = 1 \cdots, k)$ are added to each instance 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 build a more accurate prediction model.

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

Figure 3. An illustration of the RK learning 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 a number of groups, by taking the relationships between instances in T_1 and T_4 into consideration. Assume $W \in \mathbb{R}^{L_1 \times N}$ denotes the similarity matrix between T_1 and T_4 with each $w_{i,j}$ describing 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 pair-wise similarities of all examples in G_τ can be defined as in Eq. (17),

$$S_{G_\tau} = \frac{1}{|G_\tau|^2} \sum_{x \in G_\tau} \sum_{x' \in G_\tau} S(x, x') \tag{17}$$

where $S(x, x')$ denotes the similarity of two examples x and x'. On the other hand, the variance of the relationship values of all examples in G_τ can be calculated by Eq. (18).

$$\delta_{G_\tau} = \frac{1}{|G_\tau|} \sum_{y_i \in G_\tau} (\beta_j - \beta_{G_\tau})^T (\beta_j - \beta_{G_\tau}) \tag{18}$$

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, $\tau = 1, \cdots, k$, such that the sum of the similarities is maximized while the sum of variances is minimized as defined by Eq. (19),

$$J_e' = max \sum_{\tau=1}^{k} J_{G_\tau} = max \sum_{\tau=1}^{k} \frac{S_{G_\tau}}{\delta_{G_\tau}} \tag{19}$$

Explicitly solving Eq. (19) is difficult, alternatively, we can employ a recursive hill-climbing search process to find solutions. Assume instances in T_4 are clustered into k clusters, $G_1, \cdots, G_k$, moving an instance x from cluster G_i to cluster G_j will only change the cluster objective values J_{G_i} and J_{G_j}. Therefore, in order to maximize Eq. (19), at each step t, we can randomly select an instance x from a cluster G_i, and move x to cluster G_j. We accept the movement only if the Inequity (20) reaches a larger value at step $t + 1$.

$$J_{G_i}(t) + J_{G_j}(t) < J_{G_i}(t+1) + J_{G_j}(t+1) \tag{20}$$

Based on the search process in Inequity (20), major steps of the relational k-means are listed in the 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
  $G_1, \cdots, G_k \leftarrow$ Apply k-means to W
  for $t \leftarrow 1$ to T **do**
    $x \leftarrow$ Randomly select an instance from T_4
    $G_i \leftarrow$ current cluster of instance x
    $J_{G_i}(t) \leftarrow$ Calculate G_i's objective value in Eq. (19)
    $J_{G_i}(t+1) \leftarrow G_i$'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 (20) is *true* **then**
        $G_j \leftarrow G_j \cup x$; $G_i \leftarrow G_i \setminus x$
        Break
      end if
    end for
  end for
  $\mu_1, \cdots, \mu_k \leftarrow$ Calculate cluster centers for $G_1, \cdots, G_k$
  return $\mu_1, \cdots, \mu_k$

4 RK-TS^{3}VM Learning Framework

Algorithm 4 lists the detailed procedures of the RK-TS^{3}VM learning framework which is the combination of the TS^{3}VM and RK learning models. Given a training chunk D, the first step is to identify the four types of examples T_1, T_2, T_3, T_4 based on the procedures discussed in Section 2. The second step is to construct a group of k feature vectors, denoted by $\mu = \{\mu_1, \cdots, \mu_k\}$, by applying RK to T_1 and T_4. In the third and fourth steps, the k new features will be appended to each instances in T_1, T_2, T_3 to form three new sets denoted by T_1', T_2', and T_3' respectively (Section 3.2). The fifth step is to build a TS^{3}VM model F from T_1', T_2', and T_3' (Section 3.1). At the last step, the feature vectors μ and the TS^{3}VM model are combined as the final prediction model. For any instance x in the test chunk, RK-TS^{3}VM first calculate k new features for x, then uses TS^{3}VM model to predict a label for x.

5. Experiments

In this section, we report experimental results and comparisons of the proposed RK-TS^{3}VM framework from the following three aspects: (1)the parameter setting for TS^{3}VM; the algorithm performance with respect to the (2) concept drifting rate; and (3) the percentages of labeled examples.

5.1 Experimental Settings

Benchmark Methods: We implement RK-TS^{3}VM in C++ by using the VC++ 6.0 developing environment. Because RK-TS^{3}VM is a SVM based learning framework, we use

Algorithm 4 RK-TS^{3}VM Learning Framework

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

Step 1: Identify the four types of data T_1, T_2, T_3, T_4 in D according to the labeled rate l and concept drifting probability c using Eq. (1)

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

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

Step 4: Get the new samples T_1' , T_2', and T_3' from **Step 3**

Step 5: Construct a TS^{3}VM model using T_1', T_2', and T_3', and get the model F

return μ and F together as the prediction model

the optimization package in the IMSL Fortran Library [1] to solve the TS^{3}VM model in Eq.(9). To assess the algorithm performance, we use the conventional SVM model, the Transfer SVM (TrSVM) model (which is formulated following the mutli-task SVM model [5]), and the semi-supervsied SVM (S^{3}VM) model on both labeled and unlabeled examples, as the benchmark models for comparisons. We build the SVM and TrSVM models on labeled samples and the S^{3}VM model is trained on both labeled and unlabeled samples.

Data Streams: A synthetic data stream and two real-world data streams are used in our experiments. The employment of the synthetic data stream is to assess the algorithm performance under different concept drifting scenarios (which we usually don't have control on the real-world streams). The real-world data streams provide the genuine algorithm performance in real-world environments.

In our experiment, the ***synthetic data stream*** is 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 are generated by a uniform distribution between 0 and 1, and the classification boundary is controlled by Eq.(21),

$$\sum_{i=1}^{d} a_i x_i = a_0, \tag{21}$$

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 is labeled as $y_i = +1$; otherwise, $y_i = -1$. To simulate the labeling process, we randomly choose p percentage of examples and label them by using Eq.(21). To simulate the concept drifting, a_i is given a probability c ($0 \leq c \leq 1$) to evolve between a_i and $a_i + 0.1$ with 10% chance to reverse the direction. Besides that, we set a margin between the two classes. For the "+1" class, the boundary is set to be $\sum_{i=0}^{d} a_i x_i + 0.05$; while for the "-1" class, the boundary is set to be $\sum_{i=0}^{d} a_i x_i - 0.05$. To keep class distributions relatively balanced, we enforce 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 is introduced to the stream by randomly flopping the class labels of the selected instances.

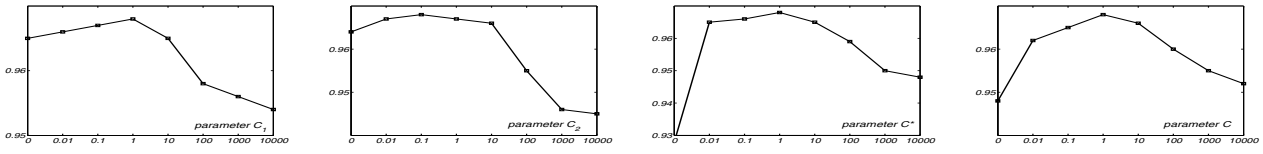
Real-world data streams Two real-world data streams, Sensor and Power Supply, downloaded from the Stream Data Mining Repository [18], are used in our experiments. Sensor stream contains information (temperature, humidity, light, and sensor voltage) collected from 56 sensors deployed in a lab environment. The data are read every 1-3 minutes from all sensors, and the whole stream contains information recorded over a 2 months period. The learning task is to correctly identify the sensor ID (1 out of 56 sensors) purely based on the sensor data and the recording time. While the data flow over time, so do the concepts underlying the stream. For example, the lighting during the working hours is generally stronger than the night, and the temperature of specific sensors (conference room) may suddenly rise during the meetings. Power Supply stream contains three year hourly power supply of an electricity company from two sources: power supply from main grid and power transformed from other grids. The learning task of this stream is to predict which hour (1 out of 24 hours) the current power supply belongs to. The concept drifting in this stream is mainly driven by the issues such as the season, weather, or hour of a day. In our experiments, both streams are transferred into binary-class classification problems.

5.2 Parameter Setting for TS^{3}VM

Four important parameters of the TS^{3}VM objective function in Eq.(9) are C_1, C_2, C, and C^*. From 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 let $C_1 \gg C_2$, i.e., $\frac{C_1}{C_2} > 10000$, 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 misclassified examples in T_1 and T_2, and the last parameter C^* controls the penalty of the misclassified examples in T_3.

In Fig.4, we report the accuracies of TS^{3}VM (the y-axis) with respect to different parameter values (the x-axis). All the results are based on the average over 100 data chunks each containing 500 examples (synthetic data stream). The concept drifting probability c is set to be 50%, and the percentage of labeled examples $p = 10\%$. From Figs.4(a) and (b), we can observe that increasing C_2 will result in a more noticeable performance deterioration than increasing C_1. This is consistent our assumption that task 1 is supposed

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



(a) (b) (c) (d)

Figure 4. The parameter study of the TS^{3}VM learning model. The y-axis denotes the prediction accuracies of TS^{3}VM, and the x-axis denotes the parameter values for (a) C_1, (b) C_2, (c) C^*, and (d) C, each varies from 0 to 10000.

to comply with the same distributions as the target domain. The results in Figs.4(a) and (b) also suggest that a reasonable setting for C_1 and C_2 is to ensure that $\frac{C_1}{C_2} = 10$. From Fig.4(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 to learn a classifier will significantly improve the performance. Based on the above observations, in following experiments, we set C_1 to be 10 while the remaining to be 1.

5.3 Sensitivity to the Concept Drifting

In order to investigate RK-TS^{3}VM's sensitivity with respect to different probabilities of the concept drifting, we collect a set of results from 100 data chunks with chunk size 500, and the percentage of labeled examples p is set to be 10%. According to our description in Section 2, the sizes of the Types I, II, III, and IV examples are partially determined by the concept drifting probability value c. Although we have the power to control the probability value c for synthetic streams, we do not, however, know the probability value c in real-world data streams. To bring the results one step closer to the real-world setting, we pretend that we do not know the probability value c and simply take a small portion (i.e., 10%) of instances at the tail of each training chunk as the Types I and III examples, and the rest of samples are taken as the Types II and IV examples.

In Table 1, we report the algorithm performance with respect to different concept drifting probability c values (for synthetic stream). For any specific c value, RK-TS^{3}VM outperforms its other peers and SVM also performs the worst. On the other hand, when comparing each algorithm with respect to different concept drifting probability values, we can observe that all the four algorithms suffer a loss in accuracy when c increases. This asserts that RK-TS^{3}VM is relatively stable in handling data streams with high concept drifting probabilities. We believe that the robustness of the RK-TS^{3}VM may be attributed to the following two facts. First, due to the difficult of identifying the concept drifting points, we cannot determine the proper chunk size to ensure that the concepts within a data chunk remain stable. So the concept drifting may actually happen within a data chunk. RK-TS^{3}VM considers that each data chunk is subject to the concept drifting and only a small portion of instances at the tail of each chunk have the same distribution as the target domain. This categorization closely simulates the real-world data stream, especially when the concept drifting probability value c is large. Second, the TS^{3}VM and RK learning models can properly utilize the four types of examples by learning from them separately. Consequently, the RK-TS^{3}VM is able to receive good performance even if it might not be able to accurately estimate the size of the Types I, III and Types II, IV examples.

Table 1. Synthetic stream: different concept drifting rates

Learning Models	Concept Drifting		
	c=10%	c=30%	c=60%
SVM	$0.8792_{\pm 0.07}$	$0.8644_{\pm 0.07}$	$0.8348_{\pm 0.08}$
TrSVM	$0.9215_{\pm 0.04}$	$0.9047_{\pm 0.04}$	$0.8912_{\pm 0.04}$
S^{3}VM	$0.9573_{\pm 0.04}$	$0.9427_{\pm 0.05}$	$0.9277_{\pm 0.05}$
RK-TS^{3}VM	$\mathbf{0.9610_{\pm 0.01}}$	$\mathbf{0.9548_{\pm 0.02}}$	$\mathbf{0.9473_{\pm 0.02}}$

5.4 Sensitivity to Labeled Examples

In Table 2 we report the algorithm performance with respect to different percentages of labeled examples. In the experiment, we use 100 data chunks each containing 500 examples. A fixed portion of 10% examples at the tail of the training chunk are selected as the same distribution examples (Types I & III). The results in Table 2 indicate that for a specific p value, S^{3}VM and RK-TS^{3}VM consistently outperforms other two methods which discard the unlabeled for learning. This is consistent with the common observations made from the semi-supervised learning, and further concludes that utilizing unlabeled samples is also an effective way for learning from data streams. When comparing four methods across different percentages of labeled examples, we can observe that all four methods receive significant improvements, especially for SVM and TrSVM, when the value p increases. This observation indicates that providing a sufficient number of labeled samples is crucial to

Table 2. Synthetic stream: different % of labeled exps.

Learning Models	Labeled example percentages		
	p=1%	p=5%	p=10%
SVM	$0.5818_{\pm 0.11}$	$0.7600_{\pm 0.10}$	$0.8605_{\pm 0.07}$
TrSVM	$0.5896_{\pm 0.14}$	$0.8387_{\pm 0.07}$	$0.9033_{\pm 0.05}$
S^3VM	$0.8917_{\pm 0.05}$	$0.9229_{\pm 0.04}$	$0.9416_{\pm 0.02}$
RK-TS^{3}VM	$\mathbf{0.9190_{\pm 0.04}}$	$\mathbf{0.9415_{\pm 0.03}}$	$\mathbf{0.9517_{\pm 0.02}}$

Table 3. Sensor stream: different chunk sizes

Learning Models	Chunk Size		
	n=100	n=500	n=1000
SVM	$0.6262_{\pm 0.11}$	$0.7655_{\pm 0.07}$	$0.7750_{\pm 0.07}$
TrSVM	$0.6985_{\pm 0.11}$	$0.7774_{\pm 0.06}$	$0.7860_{\pm 0.05}$
S^3VM	$0.7511_{\pm 0.08}$	$0.7994_{\pm 0.07}$	$0.8083_{\pm 0.06}$
RK-TS^{3}VM	$\mathbf{0.7729_{\pm 0.07}}$	$\mathbf{0.8050_{\pm 0.06}}$	$\mathbf{0.8094_{\pm 0.06}}$

ensure the accuracy of the prediction models, if learning has no mechanism of utilizing unlabeled samples or no unlabeled samples are available to boost the learning.

5.5 Results on Real-World Data Streams

In Tables 3 and 4, we report the algorithm performance on two real-world data streams with different chunk sizes (with p=10%). The results assert that comparing to its other peers, RK-TS^{3}VM can help build the most accurate prediction model. When comparing each method across different chunk sizes, we can see that the prediction models trained from RK-TS^{3}VM are the most robust ones with respect to different chunk sizes. Take the Sensor stream in Table 3 as an example, when the chunk size increases from 100 to 1000, RK-TS^{3}VM has the lowest change (only 0.0365) compared to SVM (0.1488), TrSVM (0.0875) and S^3VM (0.0472) methods. The above results assert that RK-TS^{3}VM not only helps build accurate prediction models from data streams, but also relatively less sensitive to different chunk sizes. In Tables 5 and 6, the four methods are compared with respect to different percentages of labeled instances (with n=500). For comparisons purposes, we also list the detailed chunk by chunk results of all four methods in Figs. 5 and 6. The results also validate our conclusion that RK-TS^{3}VM is the most robust model with respect to different chunk sizes, percentages of labeled instances, and concept drifting probabilities.

The above advantages render RK-TS^{3}VM a useful tool for practical usages, where real-world users may not have any priori knowledge on the proper chunk sizes, labeling percentages, and concept drifting rates in the data streams. So a method insensitive to these parameters is able to deliver stable and consistent mining results.

Table 4. PowerSup. stream: different chunk sizes

Learning Models	Chunk Size		
	n=100	n=500	n=1000
SVM	$0.6032_{\pm 0.11}$	$0.6919_{\pm 0.07}$	$0.6923_{\pm 0.07}$
TrSVM	$0.6076_{\pm 0.12}$	$0.6957_{\pm 0.05}$	$0.7105_{\pm 0.06}$
S^3VM	$0.6148_{\pm 0.11}$	$0.7002_{\pm 0.05}$	$0.7195_{\pm 0.05}$
RK-TS^{3}VM	$\mathbf{0.6700_{\pm 0.09}}$	$\mathbf{0.7124_{\pm 0.04}}$	$\mathbf{0.7226_{\pm 0.05}}$

Table 5. Sensor stream: different % of labeled examples

Learning Models	Percentage of Labeled Examples		
	p=1%	p=5%	p=10%
SVM	$0.5406_{\pm 0.10}$	$0.7063_{\pm 0.09}$	$0.7655_{\pm 0.07}$
TrSVM	$0.6175_{\pm 0.11}$	$0.7475_{\pm 0.10}$	$0.7774_{\pm 0.06}$
S^3VM	$0.7826_{\pm 0.08}$	$0.7913_{\pm 0.07}$	$0.7994_{\pm 0.07}$
RK-TS^{3}VM	$\mathbf{0.7967_{\pm 0.08}}$	$\mathbf{0.7980_{\pm 0.07}}$	$\mathbf{0.8050_{\pm 0.06}}$

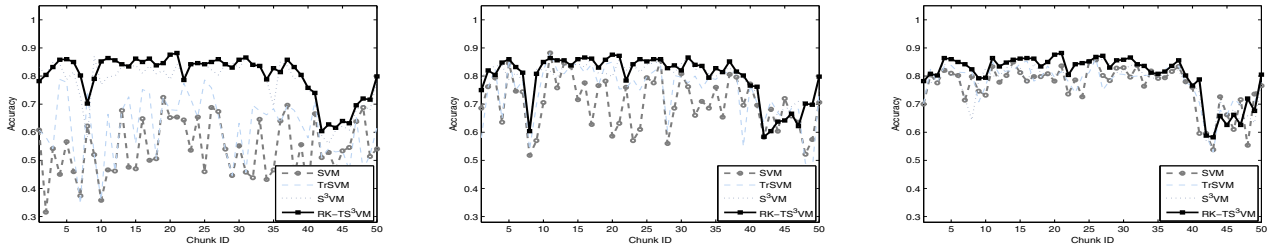
6 Conclusions

For years, stream data mining research has been primarily focused on the data volumes and the concept drifting challenges, under assumption that the stream data are fully labeled. Although the advancement in the hardware and networking technologies has made the data collection easier than ever before, labeling is still a rather expensive process and instance labels are hardly immediately available for stream data which constantly flow with large volumes. The mining algorithms should therefore consider a threefold change, labeled/unlabled samples, data volumes, and concept drifting, to build accurate prediction models. In this paper, we proposed a relational *k*-means based transfer semi-supervised SVM learning framework (RK-TS^{3}VM) for data streams containing both labeled and unlabeled samples. Our essential goal is to leverage labeled and unlabeled examples to boost the learning. This goal is achieved through the combination of a transfer semi-supervised SVM learning paradigm and a relational *k*-means based learning model. Empirical studies on both synthetic and real-world data streams demonstrated that RK-TS^{3}VM is superior to other simple approaches such as the classical supervised SVM, Transfer SVM, and Semi-Supervised SVM models.

The contribution of the work reported in the paper is fourfold: (1) we characterize instances in data streams con-

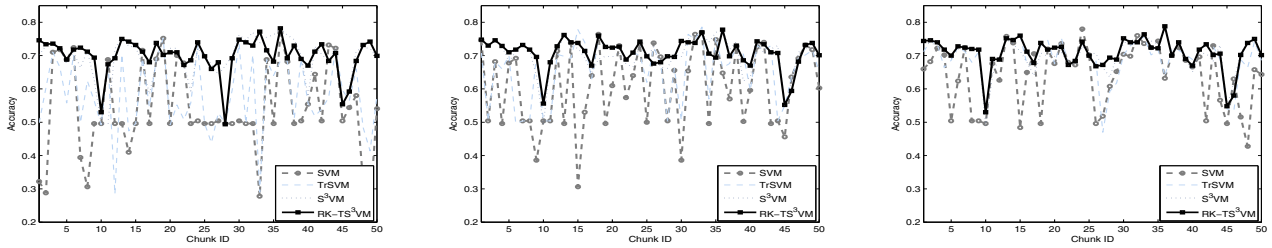
Table 6. PowerSup. stream: different % of labeled exps.

Learning Models	Percentage of Labeled Examples		
	p=1%	p=5%	p=10%
SVM	$0.5408_{\pm 0.13}$	$0.6026_{\pm 0.12}$	$0.6919_{\pm 0.07}$
TrSVM	$0.5792_{\pm 0.10}$	$0.6529_{\pm 0.10}$	$0.6957_{\pm 0.05}$
S^3VM	$0.6785_{\pm 0.06}$	$0.6991_{\pm 0.06}$	$0.7002_{\pm 0.05}$
RK-TS^{3}VM	$\mathbf{0.7002_{\pm 0.05}}$	$\mathbf{0.7011_{\pm 0.05}}$	$\mathbf{0.7124_{\pm 0.04}}$



(a) p=1% (b) p=5% (c) p=10%

Figure 5. Chunk by chunk comparison of the first 50 data chunks (Sensor stream, n=500)



(a) p=1% (b) p=5% (c) p=10%

Figure 6. Chunk by chunk comparison of the first 50 data chunks (Power supply stream, n=500)

taining labeled and unlabeled samples into four categories. Such a categorization not only provides a clear concept drifting definition, but also may motivate interested readers to devise efficient and effective solutions for data stream with labeled and unlabeled samples; (2) we proposed a relational k-means based learning model, which is useful for semi-supervised learning when training and test data have different distributions; (3) we proposed a transfer semi-supervised model to leverage labeled and unlabeled samples for learning. This model is generally useful when training examples share similar distributions to the test data; and (4) although we used SVM-like model as the learning algorithm, our principle of combining transfer and semi-supervised learning and relational clustering can be further extended to other learning algorithms for mining data streams contained labeled and unlabeled samples.

References

[1] C. Aggarwal, J. Han, J. Wang, and Y. Philip. A framework for clustering evolving data streams. In *Proc. of VLDB, 2003*.

[2] O. Chapelle, V. Sindhwani, and S. Keerthi. Optimization techniques for semi-supervised support vector mahchines. *J. of Machine Learning Research*, 9:203–233, 2008.

[3] R. Collobert, F. Sinz, and J. Weston. Large scale transductive svms. *J. of Machine Learning Research*, 7:1687–1712, 2006.

[4] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. of ACM KDD, 2000*, pages 71 – 80.

[5] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proc.of ACM KDD, 2004*, pages 109 – 117.

[6] W. Fan. Systematic data selection to mine concept-drifting data streams. In *Proc. of ACM KDD, 2004*, pages 128–137.

[7] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proc. of ACM KDD, 2001*.

[8] S. Laxman, P. Sastry, and K. Unnikrishnan. A fast algorithm for finding frequent episodes in event streams. In *Proc. of ACM KDD, 2007*, pages 410–419.

[9] J. Gao, W. Fan, and J. Han. On appropriate assumptions to mine data streams:analysis and practice. In *Proc. of IEEE ICDM, 2007*.

[10] M. Mohammad, J. Gao, L. Khan, and J. Han. A practical approach to classify evolving data streams:training with limited amount of labeled data. In *Proc. of IEEE ICDM, 2008*.

[11] B. Scholkopf, A. Smola, and K. Muller. Kernel principal component analysis. In *Proc. of ICANN, 1997*.

[12] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proc. of ACM KDD, 2003*.

[13] A. Yuille and A. Rangarajan. The concave-convex procedure. In *Advances in NIPS*, 2002.

[14] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, U. of Wisconsin-Madison, 2005.

[15] P. Zhang, X. Zhu, and Y. Shi. Categorizing and mining concept drifting data streams. In *Proc. of ACM KDD, 2008*.

[16] X. Zhu, P. Zhang, X. Lin, and Y. Shi. Active learning from data streams. In *Proc. of IEEE ICDM, 2007*.

[17] X. Zhu, and R. Jin. Multiple Information Sources Cooperative Learning. In *Proc. of IJCAI, 2007*.

[18] X. Zhu. Stream Data Mining Repository. `http://www.cse.fau.edu/~xqzhu/stream.html`. 2009.