Forward Classification on Data Streams

Peng Wang[1,2,3], Peng Zhang[3], Yanan Cao[3], Li Guo[3], and Bingxing Fang[4]

[1] Institute of Computing Technology, Chinese Academy of Science, China
[2] University of Chinese Academy of Science, China
[3] Institute of Information Engineering, Chinese Academy of Science, China
[4] Chinese Academy of Engineering, China
peng860215@gmail.com, {zhangpeng,caoyanan,guoli}@iie.ac.cn, fangbx@cae.cn

Abstract. In this paper, we explore a new research problem of predicting an incoming classifier on dynamic data streams, named as *forward classification*. The state-of-the-art classification models on data streams, such as the incremental and ensemble models, fall into the *retrospective classification* category where models used for classification are built from past observed stream data and constantly lag behind the incoming unobserved test data. As a result, the classification model and test data are temporally inconsistent, leading to severe performance deterioration when the concept (joint probability distribution) evolves rapidly. To this end, we propose a new *forward classification* method which aims to build the classification model which fits the current data. Specifically, forward classification first predicts the incoming classifier based on a line of recent classifiers, and then uses the predicted classifier to classify current data chunk. A learning framework which can adaptively switch between forward classification and retrospective classification is also proposed. Empirical studies on both synthetic and real-world data streams demonstrate the utility of the proposed method.

Keywords: Data stream classification, linear dynamic system, concept drifting.

1 Introduction

Data stream classification is an important tool for real-time applications. For example, data stream classification is popularly used in real-time intrusion detection, spam filtering, and malicious website monitoring. Compared to data mining models, data stream classification models face extra challenges from the unbounded stream data and the continuously evolving concept (joint probability distribution)[21,18] underneath stream data.

In data stream scenarios, the classification ability of a stream classification model generally decreases with time because of the concept evolving reality[1]. For example, in data streams, a classification model c_N built at time stamp N may classify its synchronous data chunk D_N accurately, but its accuracy on incoming data chunk D_{N+1} may deteriorate significantly. This is because that data distributions in D_{N+1} may be significantly different from the training samples collected at time stamp N (in this paper, samples, records, and instances are interchangeable terms). As a result, in data stream classification, it is important to build the classifier which fits the concept of current data.

Unfortunately, existing data stream classification models, including the incremental models[5,8] and ensemble models[12], are based on the assumption that same data is first used for training then for testing. However, in real-world applications, we have to classify the incoming data first and the labeled samples of the incoming data for training tend to lag behind (for example, for fraud behavior classification in bank, typically it will take days or weeks to find whether the user was actually a fraud or not.). As a result, the classifier does not temporally consistent with test data, as illustrated in Fig.1. This type of approaches regard concepts of data stream as sequence of recurring events, so they can only model the recurring probability of old concepts/classifiers, but cannot forecast a new classifier not showing up before. In this paper, we refer to this type of classification models as ***retrospective prediction***, i.e., uses models directly trained from past stream data to classify incoming data. To synchronize the

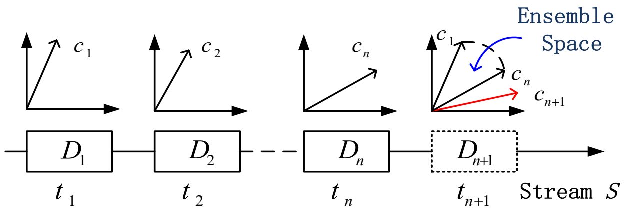

Fig. 1. The ensemble (retrospective) model can describe all the past concepts. However, it fails to describe the incoming concept(classifier) c_{N+1} that never appeared before.

classification model and test data on data streams, we present a novel ***forward classification*** method. Forward classification uses past classifiers to predict an incoming classifier, which is further used to classify incoming test data. Compared to the retrospective classification, the classifier used to classify incoming test data is not directly trained from historical stream records.

The main challenge of forward classification is to accurately predict the incoming classifier based on the past classifiers, which demands to model the evolution trend underneath the classifiers built from historical stream data. In this paper, we assume that concept evolution is a Markov process, i.e., the current concept of data stream is probabilistically determined by its previous state. This assumption is commonly used in data stream research [17]. Then, based on the observation that the classification boundaries of all the past classifiers can be represented as continuous vectors, we propose to use Linear Dynamic System (LDS)[3,11] as the solution. In this way, tracking the evolving concept is tantamount to learning the LDS model based on all the past observed classifiers (continuous vectors), and predicting the incoming classifier is equivalent to inferring the next state of the system.

Forward classification does not always outperform retrospective classification. As a model's performance is closely related to its version space [6,15], we design a flexible learning framework, which can adaptively switch between the forward classification and the retrospective classification, which is based on ensemble learning. In doing so, our learning framework is robust under different concept drifting patterns. We also demonstrate the effectiveness of the proposed method by experiments on both synthetic and real-world data streams.

The remainder of the paper is structured as follows: Section 2 introduces the modeling of the forward classification using Linear Dynamic System (LDS). Section 3 conducts the experiments. Section 4 surveys related work. We conclude the paper in Section 5.

2 Model for Forward Classification

In this section, we first describe concept evolution with a graphical model. Then, we discuss how to use Linear Dynamic System (LDS) as the solution. Finally, a forward classification framework is proposed. Consider a data stream S consisting of infinite number of records (x, y), where $x \in R^d$ is the feature vector and y is the class label. Assume the records arrive chunk-by-chunk. The records arrive at time stamp n are denoted as data chunk D_n. The classifier built on D_n is denoted as c_n. The concept at time stamp n is the joint probability distribution $p(x, y|n)$.

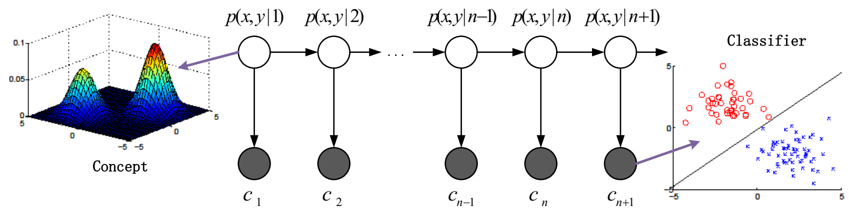

Fig. 2. Graphical model for concept description

Fig. 2 is the graphical model describing the concept evolution under the Markov assumption. The solid gray circles stand for the classifiers. The hollow circles represent the hidden concepts. The graph can be decomposed into two processes: a evolution process $p(x, y|n-1) \longrightarrow p(x, y|n)$ that describes the concept evolution between two neighboring concepts, and a modeling process $p(x, y|n) \longrightarrow c_n$ that describes the classifier training from the labeled training data. Noise is also involved in modeling process because the training set usually is a small biased data set sampled from the hidden concept. Based on graph model forward classification is formally defined as:

Forward classification: Given W historical classifiers $C = \{c_{N-W+1}, \cdots, c_N\}$ which are built consecutively from data stream S, the forward classification aims to predict the incoming classifier c_{N+1}:

$$f : (\{c_{N-W+1}, \cdots, c_N\}) \longrightarrow \widehat{c_{N+1}} \tag{1}$$

Here, c_{N+1} is the correct incoming classifier but cannot be known before hand while $\widehat{c_{N+1}}$ is our prediction. To solve the prediction problem, we use *Linear Dynamic System(LDS)* as the solution. In a deterministic LDS, a set of linear equations, governs the system evolution. Generally, the evolution of hidden concept is a stochastic process instead of a deterministic one. Thus, we add a random variable (denoted as w) to model the randomness as shown in Eq. (2).

$$z_{n+1} = A \cdot z_n + w_{n+1}. \tag{2}$$

To model the concept drifting in data stream with LDS, we assume the classifier model c_n can be converted to a vector of fixed length, such as linear classifier model $y = \omega x + b$ can be represented by the vector $[\omega, b]$. And we assume the concept $p(x, y)$ can be represented by a vector z. So the concept drifting process equates to the evolving of z. Moreover, to model the probabilistic dependence of $p(z_n|z_{n-1})$ and $p(c_n|z_n)$, we assume that probabilities $p(z_n|z_{n-1})$ and $p(c_n|z_n)$ follow Gaussian distributions:

$$p(z_n|z_{n-1}) = \mathcal{N}(z_n|A \cdot z_{n-1}, \varGamma) \tag{3}$$

$$p(c_n|z_n) = \mathcal{N}(c_n|B \cdot z_n, \varSigma) \tag{4}$$

where A represents the transform matrix that governs how the concept evolves, $A \cdot z_{n-1}$ is the mean value of z_n, $\varGamma$ is the covariance of the Gaussian noise incurred by the irregular concept evolution. B represents the transform matrix that governs how the latent concept maps to the classifier, $\varSigma$ is the Gaussian noise incurred by the biased sampled training data. Eqs. (3) and (4) can be described as noisy linear equations:

$$z_n = A \cdot z_{n-1} + w_n \tag{5}$$

$$c_n = B \cdot z_n + v_n \tag{6}$$

$$z_1 = \mu_0 + u \tag{7}$$

where Eq. (7) describe the initial state in LDS, $w_n \sim \mathcal{N}(w_n|0, \varGamma)$, $v_n \sim \mathcal{N}(v_n|0, \varSigma)$ and $u \sim \mathcal{N}(u|0, V_0)$ are the Gaussian noises.

We have described classifier prediction problem with LDS, then we will show how to learn the model and the resulting forward classification framework.

2.1 Model Learning

The learning problem [10,13] is to find the optimal parameter θ that maximizes the likelihood function of the observations $C = \{c_{N-W+1}, \cdots, c_N\}$, as in Eq. (8),

$$\widehat{\theta} = arg \max \log p(C|\theta). \tag{8}$$

where $p(C|\theta)$ is a marginal distribution of the joint distribution $p(C, Z|\theta)$ *w.r.t.* Z, as in Eqs. (9) and (10).

$$p(C|\theta) = \int_Z p(C, Z|\theta) dZ \tag{9}$$

$$p(C, Z|\theta) = p(z_1|\mu_0)\Big[\prod_{i=2}^{N} p(z_i|z_{i-1}, A)\Big]\prod_{j=1}^{N} p(c_j|z_j, B) \tag{10}$$

Eq. (10) is comprised of three parts. The first part is the probability of the initial state, the second part is probability of the concept evolution, and the last part is the probability of mapping the latent variables to classifiers. All the three parts are under the Gaussian distribution assumption.

Because $p(C|\theta) = \int_Z p(C, Z|\theta)dZ$ is very difficult to calculate, finding optimal solution is hardly achievable. Therefore, we use the Expectation Maximization (EM) algorithm to maximize $\log p(C|\theta)$. The EM algorithm starts with well-selected initial values for the parameters θ^{old}. Then, in the E-step, we use θ^{old} to find the posterior distribution of the latent variables $p(Z|C, \theta^{old})$. We then take the expectation of the log likelihood *w.r.t* the posterior distribution $p(Z|C, \theta^{old})$. In the M-step, we aim to find θ^{new} that maximizes $Q(\theta, \theta^{old}) = \mathbb{E}_{Z|\theta^{old}}[\ln p(C, Z|\theta)]$. The E-step and M-step are recursively executed until $Q(\theta, \theta^{old})$ converges. The details for the E-step and M-step of EM algorithm for learning LDS can be found in [3]. The basic process is summarized in Algorithm 1. The future classifier $\widehat{c_{N+1}}$ can be predicted based on the parameters θ and estimated current hidden state $\widehat{z_N}$ learned above:

$$\widehat{c_{N+1}} = B \cdot A \cdot \widehat{z_N} \tag{11}$$

Algorithm 1. Learning LDS

```
Require: A set of classification model in time sequential $C = \{c_{N-W+1}, \cdots, c_N\}$;
    Initial value $\theta^{old}$;
    The up bound of iterations $M$;
    The convergence threshold $\epsilon$.
Ensure: $\theta^{new} = \{\mu_0^{new}, V_0^{new}, A^{new}, \Gamma^{new}, B^{new}, \Sigma^{new}\}$;
    The expected latent state $\widehat{Z_N}$ for $N$ time block.
 1: Complete-data likelihood $Q$; $Q_{pre} \longleftarrow +\infty$,$Q_{new} \longleftarrow 0$;
 2: $i \longleftarrow 0$;
 3: while $|Q_{pre} - Q_{new}| > \epsilon$ AND $i < M$ do
 4:     $i \longleftarrow i + 1$
 5:     $Q_{pre} \longleftarrow Q_{new}$
 6:     $\{\mathbb{E}[z_n], \mathbb{E}[z_n z_{n-1}^T], \mathbb{E}[z_n z_n^T]\} \longleftarrow$ E-step$(\theta^{old}, C)$
 7:     $\theta^{new} \longleftarrow$ M-step$([\mathbb{E}[z_n], \mathbb{E}[z_n z_{n-1}^T], \mathbb{E}[z_n z_n^T]], C, \theta^{old})$
 8:     Evaluate $Q_{new} \longleftarrow Q(\theta^{new}, \theta^{old})$ based on $\theta^{new}, C$
 9: end while
10: return $\theta^{new}$, $\widehat{z_N} = \mathbb{E}[z_N]$
```

2.2 Method Comparison and Selection

In this part, we try to answer the following questions: does the forward classification always outperform retrospective classification (e.g., ensemble method)?

If the answer is NO, then how to select proper methods for real-world data streams? Here we denote the predicted classifier in forward classification as c_f, and we use ensemble method as a typical example of retrospective method. As a classifier can be mapped to a point in hyperspace, the ensemble classifier c_e will lie at the center of the most recently W classifiers, i.e. $c_e = (1/W)\sum_{i=1}^{W} c_i$.

Intuitively, if a data stream evolves continuously with stable patterns, i.e., the transform matrix A is time-invariant, the classifier is predictable; otherwise, the predicted classifier will overfit to the fake pattern. Except the irregular evolving pattern, the random noise will also make it difficult to learn the correct evolving pattern. So forward classification will not dominate retrospective method on all data streams. What's worse, there is no analytic solution for LDS, so c_f cannot be directly compared to c_e. It raise the problem how to adaptively select between these methods in real-world data stream?

We solve the problem from the view of *version space*. Version space, in concept learning or induction, refers to a subset of all hypotheses that are consistent with the observed training examples. For data stream, all possible classifiers form the version space. As concept drifts continuously, for an incoming data chunk at time $N+1$, the classifier c_{N+1} may have many possibilities, namely the version space of c_{N+1} can be large. According to Tong's theory in [16], the larger version space is, the less accurate the classifier tends to be. We illustrate different version space on different concept evolving scenarios in Fig. 3. We can notice that, for data stream with clear evolving patterns, the version space of c_f is smaller than c_e; while for data stream **without** clear evolving pattern, the version space of c_f is bigger than c_e.

Based on the Gaussian noise assumption, c_f obeys Gaussian distribution, i.e. $c_f \sim \mathcal{N}(BA\widehat{z_N}, \Psi_f)$. We take the volume within standard deviation from $BA\widehat{z_N}$ as the version space, which is determined by Ψ_f. According the analysis in [3], we have $c_f = \widehat{c_{N+1}} = \mathcal{N}(BAz_N, BP_NB^T + \Sigma)$, where $P_N = AV_NA^T + \Gamma$. So $\Psi_f = BP_NB^T + \Sigma$. On the other hand, the covariance of c_e, denoted as Ψ_e, is $\Psi_e = (1/W)\sum_{i=1}^{W}(c_i - c_e)(c_i - c_e)^T$. The volume of version space of the classifier can be calculated by $\zeta = \prod \lambda_i$, where λ_i is the eigenvector of Ψ. By comparing the version space of c_f and c_e, we can adaptively decide which method to adopt for the data stream on hand.

2.3 The Learning Framework

In this part, we introduce the classification framework which combines retrospective and forward classification. In data streams, it is often very hard to immediately obtain labeled records for model updating. In contrast, the proposed framework can avoid such a shortcoming, by tracing the trend of concept drifting and forecasting the model that reflects the current concept, then select the proper classifier based on criteria of version space. Learning from data streams contains both training and testing processes. Our framework mainly focuses on the training process. The framework is summarized in Algorithm 2.

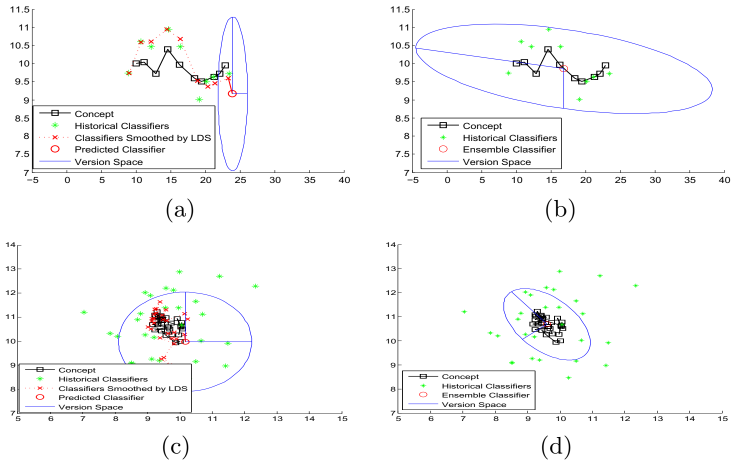

Fig. 3. Comparisons *w.r.t.* version space on different concept evolving scenarios. (a) c_f on stream with clear concept evolving pattern; (b) c_e on stream with clear concept evolving pattern; (c) c_f on stream **without** clear pattern; (d) c_e on stream **without** clear pattern

Algorithm 2. Learning Framework

```
Require: Data stream $\mathcal{D}$;
    Time window size $\beta$;
    maximum size of classifier set $W$.

1:  Build initial classification model sets $C = \{c_1\}$;
2:  N $\longleftarrow$ 1;
3:  Calculate proper initial value $\theta^{init}$.
4:  while true do
5:      $[\theta^{new}, \mathbb{E}[z_N]] \longleftarrow \text{learnLDS}(C, \theta^{init})$;
6:      Predict Model $\widehat{c_{N+1}} \longleftarrow B^{new} A^{new} \mathbb{E}[z_N]$;
7:      Calculate $c_f$ and $c_e$ and their corresponding version space $\zeta_f$ and $\zeta_e$.
8:      Compared version space and send proper classifier to test process for
        classifying $N+1$ data chunk.
9:      Sleep during $N+1$ time window as labeled records cannot be get
        immediately.
10:     Build $c_{N+1}$ based on the labeled records in $N+1$ time window;
11:     $C \longleftarrow C \cup c_{N+1}$;
12:     if $|C| > W$ then
13:         $C \longleftarrow C - \{c_{N-W}\}$;
14:     end if
15:     $\theta^{init} \longleftarrow \theta^{new}$, $N \longleftarrow N+1$;
16: end while
```

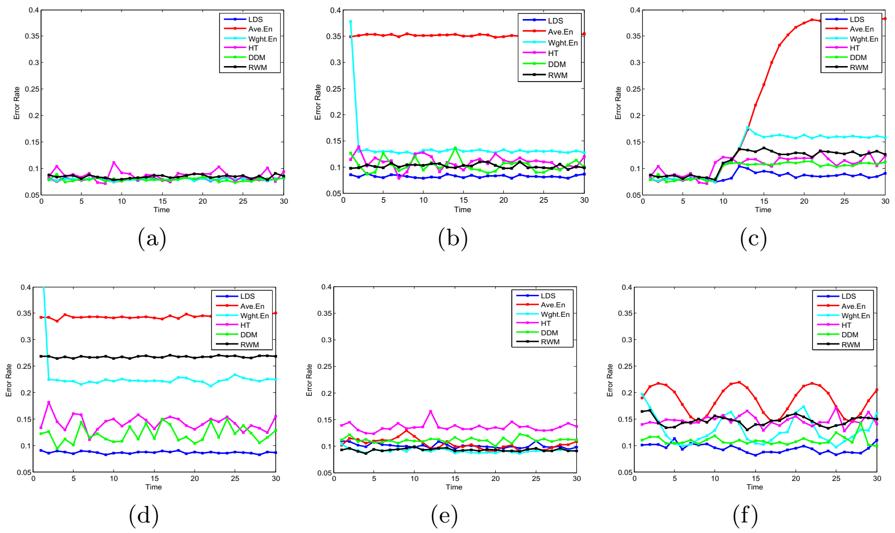

Fig. 4. Comparisons *w.r.t.* different concept evolving scenarios. (a) No concept drift; (b) Shift in one direction with speed $0.25/t$; (c) Hybrid drifting; (d) Spin with an angular speed $\frac{\pi}{3.6}/t$; (e) Random walk; (f) Period drift with a *sin* function.

Efficiency Analysis. The time cost of the proposed framework comes from two parts: training base classifiers, and predicting a future classifier. For simplicity, we take the cost of the first part as a constant value $O(1)$. The second part contains two sub-parts: an EM learning process and a prediction process. The E-step, which using the forward and backward recursions, has $O(Wd_zd_c)$ time cost, where W is the size of the historical classifier set, and d_z is the dimension of latent concept while d_c the classifier. The M-step directly updates the parameters, with time complexity of $O(W)$. Since we set the max number of iterations for EM as I, the time complexity of EM learning process is $O(WId_zd_c)$. In addition, from Eq.(11), the time complexity of predicting a future classifier is $O(1)$. To sum up, the total time complexity for a loop in the learning framework is $O(WId_zd_c)$.

3 Experimental Study

In this section, we first introduce the benchmark methods, followed by the test-bed. The test results on both synthetic and real-world data sets and the analysis will be given in the end.

We compare our method with four state-of-the-art classification method on data stream: ensemble learning[2,19], incremental learning[14], drift detection method(DDM)[7] and random walk model[9]. All of them fall into the category of retrospective learning.

3.1 Data Stream Test-Bed

In our experiment, we adopt both synthetic data stream generator and real world data streams as our test-bed.

Evolving Gaussian Generator. As we have described in the previous sections, the concept of data stream is a joint distribution $p(x, y)$. So we can generate a evolving data stream by generating records according to certain distribution and changes the distribution as time passes. For simplicity, we assume a binary classification problem, where the positive and negtive records are generated according to the Gaussian distribution. To simulate the concept evolving, we gradually change mean of the distribution as time passes. Particularly, we generate data stream with 6 types of concept drifting: *stay still*, *shift*, *hybrid*, *spin*, *random walk* and *periodic variation*.

Rotating Hyperplane is used as a test bed for CVFDT[8] models. A hyperplane in a d-dimensional space is a set of points x that satisfies $\sum_{i=1}^{d} w_i x_i = w_0$, where x_i is the i^{th} dimension of the vector x. Records satisfying $\sum_{i=1}^{d} w_i x_i \geq w_0$ are labeled as positive. Otherwise, negative. To simulate concept evolution, we let each weight attribute $w_i = w_i + d\sigma$ change with time, where σ denotes the probability that the direction of change is reversed and d denotes that the change degree.

Sensor Stream. Sensor stream[20] contains information (temperature, humidity, light, and sensor voltage) collected from 54 sensors deployed in Intel Berkeley Research Lab. The learning task is to correctly identify the sensor ID based on the sensor data. This dataset can be downloaded from website[1].

Power Supply. Power Supply stream [20] contains hourly power supply of an Italian electricity company. The learning task is to classify the time the current power supply belongs to. This dataset can be downloaded from website[2].

3.2 Results

In Fig.4 we report the algorithm performance *w.r.t.* different types of concept drifting scenarios. From Fig.4(a), we can observe that our model is as good as the classic methods if there is no concept drifting in the stream data. Fig.4(b) indicates that for concept shifting data stream, our method outperforms others. This is because when concept drifting follows stable pattern, our method can track the pattern of the changes and more accurately predict future classifiers. In Fig.4(c), the drifting pattern is unstable. For example, before $t = 10$, the concept stays still, classifiers $\{c_1, \cdots, c_{10}\}$ are determined by the parameter θ_{still} while $\{c_{10}, \cdots, c_N\}$ are determined by θ_{drift}. When $\theta_{still} \longrightarrow \theta_{drift}$, the error rate of LDS arises, as the classifier predicted with θ_{still} for the future concept. As the drifting records increase, the LDS model can gradually tracking the concept evolution. So the error rate gradually decreases. In Fig.4(d), the concept's evolving rate is fast. We can observe that LDS significantly outperforms other methods for the fast evolving data stream. In Fig.4(e), the concept of data stream evolves in a random walk manner. We can see that LDS can handle the noise factor well for it can switch to retrospective method when the evolving

[1] `http://www.cse.fau.edu/~xqzhu/Stream/sensor.arff`

[2] `http://www.cse.fau.edu/~xqzhu/Stream/powersupply.arff`

Table 1. Error rate comparisons among different methods

DataSet	Average EN	Weighted EN	H.T.	DDM	RWM	LDS
Gaussian Static	$\mathbf{0.077}_{\pm\mathbf{0.001}}$	$0.079_{\pm 0.001}$	$0.087_{\pm 0.004}$	$0.079_{\pm 0.001}$	$0.0841_{\pm 0.001}$	$0.079_{\pm 0.002}$
Gaussian Shift	$0.351_{\pm 0.003}$	$0.138_{\pm 0.017}$	$0.109_{\pm 0.004}$	$0.103_{\pm 0.004}$	$0.102_{\pm 0.003}$	$\mathbf{0.082}_{\pm\mathbf{0.004}}$
Gaussian Hybrid	$0.085_{\pm 0.002}$	$0.221_{\pm 0.046}$	$0.087_{\pm 0.005}$	$\mathbf{0.085}_{\pm\mathbf{0.002}}$	$0.094_{\pm 0.003}$	$0.111_{\pm 0.010}$
Gaussian Spin	$0.343_{\pm 0.001}$	$0.230_{\pm 0.012}$	$0.143_{\pm 0.005}$	$0.121_{\pm 0.006}$	$0.267_{\pm 0.001}$	$\mathbf{0.086}_{\pm\mathbf{0.007}}$
Gaussian Random	$0.093_{\pm 0.003}$	$0.091_{\pm 0.002}$	$0.105_{\pm 0.008}$	$0.095_{\pm 0.018}$	$\mathbf{0.087}_{\pm\mathbf{0.001}}$	$0.090_{\pm 0.005}$
Gaussian Period	$0.218_{\pm 0.031}$	$0.113_{\pm 0.014}$	$0.117_{\pm 0.004}$	$0.094_{\pm 0.003}$	$0.096_{\pm 0.003}$	$\mathbf{0.089}_{\pm\mathbf{0.003}}$
Hyperplane	$0.173_{\pm 0.043}$	$0.110_{\pm 0.027}$	$0.176_{\pm 0.031}$	$0.107_{\pm 0.021}$	$0.103_{\pm 0.024}$	$\mathbf{0.100}_{\pm\mathbf{0.023}}$
Power Supply	$0.064_{\pm 0.027}$	$0.070_{\pm 0.028}$	$0.077_{\pm 0.035}$	$0.055_{\pm 0.019}$	$0.073_{\pm 0.033}$	$\mathbf{0.052}_{\pm\mathbf{0.026}}$
Sensor	$0.042_{\pm 0.006}$	$0.039_{\pm 0.004}$	$0.037_{\pm 0.009}$	$0.034_{\pm 0.003}$	$0.079_{\pm 0.022}$	$\mathbf{0.032}_{\pm\mathbf{0.020}}$

pattern is blurred. Random walk model also perform well and it can filter out the random noise of $\{c_1, \cdots, c_N\}$, and track the proper concept, as in this model, $A \equiv I$ so it will not learn false drifting patterns. In Fig.4(f), the evolving pattern of the data stream is changing periodically. the LDS model is robust to this kind of concept drifting. In summary, our learning framework outperforms other benchmark methods for data streams with regular evolving patterns.

In Table 1, we summarize and compare the performance of different methods on all data streams. Overall, for data streams having regular evolving patterns, the performance of the proposed LDS model outperforms other benchmark methods. For data streams whose evolving is not a stable Markov process, learning methods such as drift detection and random walk perform better.

Efficiency. From our experiments, we observe that the EM algorithm converges within 100 iterations, so M is manually set to 100. In most cases, when $W > 50$, the predicted classifier is stable, so W is manually set to 50. With these settings, on our PC with 2.8G Hz CPU, the time cost for predicting the classifier is less than 1 minute. In real-world applications, it usually takes hours for the concept having detectable change, so the framework is efficient.

4 Related Work

Existing data stream classification models can be categorized into three groups: online / incremental models[8,14], ensemble learning[12,18,19] and drift detection methods [7]. We briefly describe them based on the development trace. In the simplest situation, where the concept of data stream remains stable, for each time window, the classifier has prediction variance due to limited training samples. We can use majority voting method to ensemble these classifiers, because random variance tends to compensate each other. This is the fundamental framework for retrospective classification. For data stream with concept drifting, the majority voting for ensemble is inappropriate. An alternative solution is to use weighted ensemble, where each classifier is weighted according to their consistence with the most recent observed training data. For incremental or DDM method, they keep updating the classifier using newly arriving records, enabling the learning model to adapt to new concepts. For retrospective learning models [17,4], they regard concepts of data stream as a sequence of recurring events and use the most probable concept in the future to classify incoming stream data. Unfortunately, existing weighing and updating approaches, including the proactive learning framework, cannot forecast a completely new classifier. Thus, they cannot synchronize the classifier to the evolving stream data.

Forward classifier prediction method, on the other hand, uses probabilistic model to define time evolution of the concept. By using the probabilistic model, we can approximate the distribution that maximizes the posterior probability of the model [3]. Moreover, we can predict the optimal incoming classifier by adopting the inference method. That is to say, forward classifier prediction method is able to derive better classification results.

5 Conclusions

In this paper, we present a novel *forward classification* method for classifying evolving stream data. Due to the temporal evolving of data streams, simply learning classification models from historical data, as existing *retrospective classification* methods do, is inadequate and inaccurate. So a proper classification design is to capture the evolution trend underneath stream data and use it to predict a future classifier for classification. With this vision and the assumption that the concept evolving can be characterized by Markov process, we propose to model the trend of classifiers using the *Linear Dynamic System*, through which we can model the concept drifting and predict incoming classifier. We also notice that forward classification is not overwhelmingly better than retrospective classification. Then we design the learning framework, which adaptively switches between the forward classification based on LDS and basic ensemble learning method, so it is robust to different types of data streams. Experiments on synthetic and real-world streams demonstrate that our framework outperforms other methods in different types of concept drifting scenarios.

Acknowledgments. This work was supported by the NSFC (No. 61003167), IIE Chinese Academy of Sciences (No. Y3Z0062101), 863 projects (No. 2011AA010703 and 2012AA012502), 973 project (No. 2013CB329606), and the Strategic Leading Science and Technology Projects of Chinese Academy of Sciences (No. XDA06030200).

References

1. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: On demand classification of data streams. In: Proc. of 10th ACM SIGKDD, New York, USA, pp. 503–508 (2004)
2. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: Proc. of the 15th ACM SIGKDD, New York, USA, pp. 139–148 (2009)
3. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer Science+Business Media (2006)
4. Chen, S., Wang, H., Zhou, S., Yu, P.S.: Stop chasing trends: discovering high order models in evolving data. In: Proc. of the 24th IEEE International Conference on Data Engineering, ICDE 2008, pp. 923–932. IEEE (2008)

5. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proc. of the Sixth ACM SIGKDD, KDD 2000, pp. 71–80. ACM, New York (2000)
6. Dubois, V., Quafafou, M.: Concept learning with approximation: Rough version spaces. In: Alpigini, J.J., Peters, J.F., Skowron, A., Zhong, N. (eds.) RSCTC 2002. LNCS (LNAI), vol. 2475, pp. 239–246. Springer, Heidelberg (2002)
7. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 286–295. Springer, Heidelberg (2004)
8. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proc. of the Seventh ACM SIGKDD, KDD 2001, pp. 97–106. ACM, New York (2001)
9. Jaakkola, M.S.T., Szummer, M.: Partially labeled classification with markov random walks. In: Advances in Neural Information Processing Systems (NIPS), vol. 14, pp. 945–952 (2002)
10. Kalman, R.E.: A new approach to linear filtering and prediction problems. Journal of Basic Engineering 82(5311910), 35–45 (1960)
11. Katok, A., Hasselblatt, B.: Introduction to the modern theory of dynamical systems, Cambridge (1996)
12. Opitz, D., Maclin, R.: Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research 11, 169–198 (1999)
13. Zarchan, P., Musoff, H.: Fundamentals of Kalman filtering: a practical approach. American Institute of Aeronautics Astronautics (2005)
14. Pfahringer, B., Holmes, G., Kirkby, R.: New options for hoeffding trees. In: Orgun, M., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 90–99. Springer, Heidelberg (2007)
15. Sverdlik, W., Reynolds, R.: Dynamic version spaces in machine learning. In: Proceedings of the Fourth International Conference on Tools with Artificial Intelligence, TAI 1992, pp. 308–315 (November 1992)
16. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. The Journal of Machine Learning Research 2, 45–66 (2002)
17. Yang, Y., Wu, X., Zhu, X.: Combining proactive and reactive predictions for data streams. In: Proc. of the Eleventh ACM SIGKDD, KDD 2005, pp. 710–715. ACM (2005)
18. Zhang, P., Zhu, X., Shi, Y.: Categorizing and mining concept drifting data streams. In: Proceedings of the 14th ACM SIGKDD, KDD 2008, pp. 812–820. ACM, New York (2008)
19. Zhang, P., Zhu, X., Shi, Y., Guo, L., Wu, X.: Robust ensemble learning for mining noisy data streams. Decision Support Systems 50(2), 469–479 (2011)
20. Zhu, X.: Stream data mining repository (2010), `http://www.cse.fau.edu/~xqzhu/stream.html`
21. Zliobaite, I.: Learning under concept drift: an overview. Technical Report (2009)