Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Learning Future Classifiers without Additional Data

Atsutoshi Kumagai
NTT Secure Platform Laboratories,
NTT Corporation
3-9-11, Midori-cho, Musashino-shi, Tokyo, Japan
kumagai.atsutoshi@lab.ntt.co.jp

Tomoharu Iwata
NTT Communication Science Laboratories,
NTT Corporation
2-4, Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan
iwata.tomoharu@lab.ntt.co.jp

Abstract

We propose probabilistic models for predicting future classifiers given labeled data with timestamps collected until the current time. In some applications, the decision boundary changes over time. For example, in spam mail classification, spammers continuously create new spam mails to overcome spam filters, and therefore, the decision boundary that classifies spam or non-spam can vary. Existing methods require additional labeled and/or unlabeled data to learn a time-evolving decision boundary. However, collecting these data can be expensive or impossible. By incorporating time-series models to capture the dynamics of a decision boundary, the proposed model can predict future classifiers without additional data. We developed two learning algorithms for the proposed model on the basis of variational Bayesian inference. The effectiveness of the proposed method is demonstrated with experiments using synthetic and real-world data sets.

1 Introduction

The decision boundary dynamically changes over time in some applications. For example, in spam mail classification, spammers continuously create new spam mails to overcome spam filters, and therefore, the decision boundary that classifies spam or non-spam can vary (Fdez-Riverola et al. 2007). In activity recognition using sensor data, the decision boundary can vary as user activity patterns change (Abdallah et al. 2012). When we do not update classifiers for tasks where the decision boundary evolves over time, the performance of the classifiers deteriorate quickly (Gama et al. 2014).

Many methods have been proposed for updating classifiers to maintain performance, such as online learning (Rosenblatt 1958; Crammer, Dredze, and Pereira 2009; Crammer, Kulesza, and Dredze 2009), forgetting algorithms (Klinkenberg 2004), time-window methods (Babcock et al. 2002) and ensemble learning (Wang et al. 2003; Kolter and Maloof 2007). These methods require labeled data to update a time-evolving decision boundary. However, it would be quite expensive to continuously collect labeled data since labels need to be manually assigned by domain experts. Methods that use only unlabeled data for updating have been proposed (Shimodaira 2000; Dyer, Capo, and Polikar 2014). However, using unlabeled data might be difficult. For example, in the security domain, anti-virus software vendors periodically distribute update files (that is, data for updating classifiers) to their users. Users employing anti-virus software in an offline environment, which carries a risk of virus infections via physical mediums such as USB, cannot receive these files and therefore cannot update their classifiers.

In this paper, we propose a novel approach for classifying data in the future when only labeled data with timestamps collected until the current time are available. With the proposed approach, a decision boundary is defined by the parameters of classifiers, and the dynamics of the decision boundary are modeled by using time-series models. We present a probabilistic model to realize this approach, where vector autoregressive and logistic regression models are used for the time-series model and classifier, respectively. Since our proposed model takes into account the time evolution of the decision boundary, the proposed method can reduce the deterioration of classification performance without additional data.

We developed two learning algorithms for the proposed model on the basis of variational Bayesian inference. The first algorithm is a two-step algorithm, which learns a time-series model after the classifiers learned. The second algorithm is a one-step algorithm, which learns a time-series model and classifiers simultaneously.

The remainder of this paper is organized as follows. In Section 2, we briefly review related work. We formulate the proposed model in Section 3 and describe two learning algorithms for the proposed model on the basis of variational Bayesian inference in Section 4. In Section 5, we demonstrate the effectiveness of the proposed method with experiments using synthetic and real-world data sets. Finally, we present concluding remarks and a discussion of future work in Section 6.

2 Related Work

To capture a time-evolving decision boundary, updating classifiers with labeled data has been widely used. Online learning algorithms have been developed to update classifiers after the every arrival of labeled data (Rosenblatt 1958; Crammer, Dredze, and Pereira 2009; Crammer, Kulesza, and Dredze 2009). Forgetting algorithms learn the latest deci-

sion boundary by weighting training data with their age. This weighting assumes that recent data are influential for determining the decision boundary. Examples of weighting include linear decay (Koychev 2000), exponential decay (Klinkenberg 2004), and time-window techniques that use only training data obtained recently (Babcock et al. 2002). Ensemble learning combines multiple classifiers to create a better classifier (Wang et al. 2003; Kolter and Maloof 2007; Brzezinski and Stefanowski 2014). Its mixture weight is learned so that classifiers that capture a current decision boundary well have a large weight. In some methods, active learning which is a algorithm for determine data should be labeled is used for learning the time-evolving decision boundary (Zhu et al. 2010; Zliobaite et al. 2014). Methods for learning the boundary by using unlabeled data also have been proposed (Zhang, Zhu, and Guo 2009; Dyer, Capo, and Polikar 2014). All of these methods continuously require labeled and/or unlabeled data to capture the time-evolving decision boundary. However, the purpose of this study is to predict classifiers for data obtained in the future without additional labeled and/or unlabeled data, and our task is separate from the areas where these methods should be applied.

A method for predicting future probability distribution given data with timestamps collected until the current time has been proposed (Lampert 2015). Although this method predicts the future state of a time-varying probability distribution, our method predicts the future decision boundary.

Transfer learning utilizes data in a source domain to solve a related problem in a target domain (Pan and Yang 2010; Shimodaira 2000). Our task is a special case of transfer learning, which considers data generated until a certain time to be the source domain and data generated subsequently to be the target domain where we cannot use the data in the target domain for learning. Learning the dynamics of the decision boundary is considered to be learning the model for transferring the decision boundaries in the source domain to the decision boundary in the target domain.

3 Proposed Model

First, we introduce notations and define the task studied in this paper. Let $D_t{:=}\{(\boldsymbol{x}_n^t, y_n^t)\}_{n=1}^{N_t}$ be a set of training data collected at time t, where $\boldsymbol{x}_n^t \in \mathbb{R}^d$ is the d-dimensional feature vector of the nth sample at time t, $y_n^t \in \{0,1\}$ is its class label, and N_t is the number of training data collected at time t. We discretize time at regular intervals, and data collected in the same time unit are regarded as data in the same time. We denote $X_t{:=}\{\boldsymbol{x}_n^t\}_{n=1}^{N_t}$ and $Y_t{:=}\{y_n^t\}_{n=1}^{N_t}$. Our goal is to find classifiers $h_t : \mathbb{R}^d \to \{0,1\}$, $t \in \{T+1, T+2, \dots\}$, which can precisely classify data at time t, given a set of training data $D{:=}\{D_t\}_{t=1}^{T}$.

Next, we propose a probabilistic model for predicting future classifiers. Figure 1 illustrates our approach. Using labeled data from time 1 to T, the decision boundary for each time, which is defined by classifier h_t, and the dynamics of the decision boundary are learned. Then, the decision boundary in the future $t = T+1, T+2, \dots$ is predicted by using the learned decision boundary and dynamics.

We explain the proposed probabilistic model that realizes

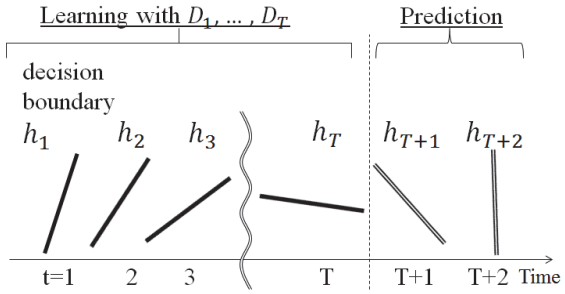


Figure 1: Illustration of our approach.

our approach. The proposed model assumes that the posterior probability of label y_n^t given feature vector $\boldsymbol{x}_n^t$ is modeled by logistic regression as.

$$p(y_n^t = 1 | \boldsymbol{x}_n^t, \boldsymbol{w}_t) = \sigma(\boldsymbol{w}_t^\top \boldsymbol{x}_n^t) = (1 + e^{-\boldsymbol{w}_t^\top \boldsymbol{x}_n^t})^{-1}, \quad (1)$$

where $\boldsymbol{w}_t \in \mathbb{R}^d$ is a parameter vector of the classifier h_t, $\sigma(\cdot)$ is the sigmoid function, and $\top$ denotes transposition. Note that the posterior probability that $y_n^t = 0$, $p(y_n^t = 0 | \boldsymbol{x}_n^t, \boldsymbol{w}_t)$, is equal to $1 - p(y_n^t = 1 | \boldsymbol{x}_n^t, \boldsymbol{w}_t)$.

We capture the dynamics of the classifier parameter $\boldsymbol{w}_t$ by using the vector autoregressive (VAR) model, which is one of the classic time-series models (Lütkepohl 2005). The mth order VAR model assumes that $\boldsymbol{w}_t$ depends linearly on the previous decision boundary $\boldsymbol{w}_{t-1}, \dots, \boldsymbol{w}_{t-m}$ as follows.

$$\boldsymbol{w}_t \sim \mathcal{N}\left(\boldsymbol{w}_t \,\middle|\, \sum_{k=1}^{m} A_k \boldsymbol{w}_{t-k} + A_0, \theta^{-1}\mathbf{I}_d\right), \quad (2)$$

where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, $A_1, \dots, A_k \in \mathbb{R}^{d \times d}$ are $d \times d$ matrices for defining dynamics, $A_0 \in \mathbb{R}^d$ is a bias term, $\theta \in \mathbb{R}_+$ denotes a hyper parameter, and $\mathbf{I}_d$ represents an $d \times d$ identity matrix. In this paper, we restrict $A_1, \dots, A_k \in \mathbb{R}^{d \times d}$ to diagonal matrices for simplicity. This means the ith component of classifier parameter $w_{t,i}$ depends only on its own previous parameter value $w_{t-1,i}, w_{t-2,i}, \cdots$. Since the mth order VAR model cannot be used as in case of $t \leq m$, we assume that $\boldsymbol{w}_t$ for $t \leq m$ is generated from $\mathcal{N}(\boldsymbol{w}_t | \mathbf{0}, \theta_0^{-1}\mathbf{I}_d)$.

The proposed model assumes that the dynamics and bias parameter A_k for $k = 0, \dots, m$ are generated from a normal distribution $\mathcal{N}(A_k | \mathbf{0}, \gamma_k^{-1}\mathbf{I}_d)$. The precision parameter γ_k for $k = 0, \dots, m$ is generated from a Gamma distribution, $\mathrm{Gam}(\gamma_k | a_k, b_k)$, which is its conjugate prior. The precision parameters for dynamics θ and θ_0 are generated from $\mathrm{Gam}(\theta | u, v)$ and $\mathrm{Gam}(\theta_0 | u_0, v_0)$, respectively.

The joint distribution of labeled data $D{:=}\{D_t\}_{t=1}^{T}$, classifier parameters $W{:=}(\boldsymbol{w}_1, \dots, \boldsymbol{w}_T)$, dynamics and bias parameters $A{:=}(A_0, A_1, \dots, A_m)$, precision parameters $\Gamma{:=}(\gamma_0, \gamma_1, \dots, \gamma_m)$, θ and θ_0 is written as:

$$\begin{aligned} &p(D, W, A, \Gamma, \theta, \theta_0) \\ &= p(D|W)p(W|A, \theta, \theta_0)p(A|\Gamma)p(\Gamma)p(\theta)p(\theta_0) \\ &= \prod_{t=1}^{T}\prod_{n=1}^{N_t} p(y_n^t | \boldsymbol{x}_n^t, \boldsymbol{w}_t) \cdot \prod_{t=1}^{m} \mathcal{N}(\boldsymbol{w}_t | \mathbf{0}, \theta_0^{-1}\mathbf{I}_d) \end{aligned}$$

$$\times \prod_{t=m+1}^{T} \mathcal{N}\left(\boldsymbol{w}_t \,\middle|\, \sum_{k=1}^{m} A_k \boldsymbol{w}_{t-k} + A_0, \theta^{-1}\mathbf{I}_d\right)$$
$$\times \prod_{k=0}^{m} \mathcal{N}(A_k|\mathbf{0}, \gamma_k^{-1}\mathbf{I}_d) \cdot \mathrm{Gam}(\gamma_k|a_k, b_k)$$
$$\times \mathrm{Gam}(\theta|u, v) \cdot \mathrm{Gam}(\theta_0|u_0, v_0). \quad (3)$$

Since our model takes into account the time development of the classifier parameter, our method improves the classification performance in a non-stationary environment. Note that though we employ the logistic regression model and the vector autoregressive model for describing our proposed model, it is possible to use other models such as boosting (Freund, Schapire, and others 1996) and Gaussian processes (Roberts et al. 2013).

4 Inference

We developed two learning algorithms for the proposed model on the basis of variational Bayesian inference (Bishop 2006). The first algorithm is a two-step algorithm, which learns a time-series model after the classifiers learned. The second algorithm is a one-step algorithm, which learns a time-series model and classifiers simultaneously.

4.1 Two-step Learning Algorithm

We explain the two-step algorithm. First, classifier parameters $\boldsymbol{w}_t$ is learned for each time t by the maximizing a posteriori estimation. Since dynamics parameters are unknown, we assume that the mean of the classifier parameters $\boldsymbol{w}_t$ is the parameters at the previous time $\boldsymbol{w}_{t-1}$, $p(\boldsymbol{w}_t|\boldsymbol{w}_{t-1}, \theta) = \mathcal{N}(\boldsymbol{w}_t|\boldsymbol{w}_{t-1}, \theta^{-1}\mathbf{I}_d)$. Then, the log posterior to be maximized is as follows.

$$\log p(\boldsymbol{w}_t|X_t, Y_t, \boldsymbol{w}_{t-1}, \theta)$$
$$= \log p(Y_t|\boldsymbol{w}_t, X_t) + \log p(\boldsymbol{w}_t|\boldsymbol{w}_{t-1}, \theta). \quad (4)$$

Classifier parameters W are obtained by repeating this procedure from $\boldsymbol{w}_1$ to $\boldsymbol{w}_T$, where $\boldsymbol{w}_0$ is a zero vector $\mathbf{0}$.

Next, we learn the time-series model on the basis of variational Bayesian inference. Using learned classifier parameters W as observed data, we consider the following joint distribution instead of (3).

$$p(W, A, \Gamma, \theta) = p(W|A, \theta)p(A|\Gamma)p(\Gamma)p(\theta)$$
$$= \prod_{t=m+1}^{T} \mathcal{N}\left(\boldsymbol{w}_t \,\middle|\, \sum_{k=1}^{m} A_k \boldsymbol{w}_{t-k} + A_0, \theta^{-1}\mathbf{I}_d\right)$$
$$\times \prod_{k=0}^{m} \mathcal{N}(A_k|\mathbf{0}, \gamma_k^{-1}\mathbf{I}_d) \cdot \mathrm{Gam}(\gamma_k|a_k, b_k) \cdot \mathrm{Gam}(\theta|u, v). \quad (5)$$

We obtain an approximate posterior distribution $q(A, \Gamma, \theta)$, called *the variational posterior*, of the posterior distribution $p(A, \Gamma, \theta|W)$ since analytically calculating $p(A, \Gamma, \theta|W)$ is impossible. It is known that the variational posterior $q(A, \Gamma, \theta)$ is a local maximum point of the following lower bound.

$$L(q) = \int q(A, \Gamma, \theta) \log \frac{p(W, A, \Gamma, \theta)}{q(A, \Gamma, \theta)} dA d\Gamma d\theta. \quad (6)$$

In this paper, the variational posterior $q(A, \Gamma, \theta)$ is assumed to factorize as follows.

$$q(A, \Gamma, \theta) = \prod_{k=0}^{m} q(A_k)q(\gamma_k)q(\theta). \quad (7)$$

By calculating the derivatives of the lower bound (6) with respect to $q(A_k)$, $q(\gamma_k)$ and $q(\theta)$, we find that each distribution of (7) has the following functional form.

$$q(\boldsymbol{\alpha}_k) = \mathcal{N}(\boldsymbol{\alpha}_k|\boldsymbol{\mu}_k, \boldsymbol{S}_k^{-1}), \quad k = 0, \cdots, m,$$
$$q(\gamma_k) = \mathrm{Gam}(\gamma_k|a_k^\gamma, b_k^\gamma), \quad k = 0, \cdots, m,$$
$$q(\theta) = \mathrm{Gam}(\theta|u^\theta, v^\theta), \quad (8)$$

where $\boldsymbol{\alpha}_k \in \mathbb{R}^d$ is a diagonal vector $(A_{k1}, \ldots, A_{kd})$ of A_k, $\boldsymbol{\mu}_k \in \mathbb{R}^d$, $\boldsymbol{S}_k \in \mathbb{R}^{d\times d}$, and $a_k^\gamma, b_k^\gamma, u^\theta, v^\theta \in \mathbb{R}_+$. These parameters are expressed as

$$\boldsymbol{S}_0 = \left(\frac{a_0^\gamma}{b_0^\gamma} + (T-m)\frac{u^\theta}{v^\theta}\right)\mathbf{I}_d,$$
$$\boldsymbol{\mu}_0 = \boldsymbol{S}_0^{-1}\frac{u^\theta}{v^\theta}\sum_{t=m+1}^{T}\left(\boldsymbol{w}_t - \sum_{k=1}^{m}\mathrm{dg}(\boldsymbol{\mu}_k)\boldsymbol{w}_{t-k}\right),$$
$$\boldsymbol{S}_k = \frac{a_k^\gamma}{b_k^\gamma}\mathbf{I}_d + \frac{u^\theta}{v^\theta}\sum_{t=m+1}^{T}\mathrm{dg}(\boldsymbol{w}_{t-k})^2,$$
$$\boldsymbol{\mu}_k = \boldsymbol{S}_k^{-1}\frac{u^\theta}{v^\theta}\sum_{t=m+1}^{T}\mathrm{dg}\left(\boldsymbol{w}_t - \sum_{l\neq k}\mathrm{dg}(\boldsymbol{\mu}_l)\boldsymbol{w}_{t-l} - \boldsymbol{\mu}_0\right)\boldsymbol{w}_{t-k},$$
$$a_k^\gamma = a_k + \frac{1}{2}d, \quad b_k^\gamma = b_k + \frac{1}{2}(\|\boldsymbol{\mu}_k\|^2 + \mathrm{Tr}(\boldsymbol{S}_k^{-1})), \quad k = 1, \ldots, m,$$
$$u^\theta = u + \frac{1}{2}(T-m)d,$$
$$v^\theta = v + \frac{1}{2}\sum_{t=m+1}^{T}(\|\boldsymbol{w}_t - \sum_{l=1}^{m}\mathrm{dg}(\boldsymbol{\mu}_l)\boldsymbol{w}_{t-l} - \boldsymbol{\mu}_0\|^2$$
$$+ \mathrm{Tr}(\boldsymbol{S}_0^{-1}) + \sum_{l=1}^{m}\boldsymbol{w}_{t-l}^\top \boldsymbol{S}_l^{-1}\boldsymbol{w}_{t-l}), \quad (9)$$

where $\mathrm{dg}(\boldsymbol{x})$ returns a diagonal matrix whose diagonal elements are $\boldsymbol{x} = (x_1, \ldots, x_d)$, $\|\cdot\|$ is ℓ_2-norm, and Tr represents the trace. In variational Bayesian inference, the variational posterior (7) is calculated by updating the variables in (8) by using equations (9) until some convergence criterion are satisfied.

4.2 One-step Learning Algorithm

With the one-step algorithm, we learn the time-series model and classifiers simultaneously.

The lower bound $L(q)$ of the log marginal likelihood of the proposed model is expressed as:

$$L(q) = \int q(W, A, \Gamma, \theta, \theta_0)\log\frac{p(D, W, A, \Gamma, \theta, \theta_0)}{q(W, A, \Gamma, \theta, \theta_0)} dW dA d\Gamma d\theta d\theta_0. \quad (10)$$

We assume that the variational posterior $q(W, A, \Gamma, \theta, \theta_0)$ is factorized as follows.

$$q(W, A, \Gamma, \theta, \theta_0) = \prod_{t=1}^{T}\prod_{i=1}^{d} q(w_{t,i}) \prod_{k=0}^{m} q(A_k)q(\gamma_k)q(\theta)q(\theta_0). \quad (11)$$

We would like to maximize the lower bound (10) in a similar manner to the two-step algorithm. However, $p(y_n^t|\boldsymbol{x}_n^t, \boldsymbol{w}_t)$ cannot be calculated analytically because of its nonlinearity. To deal with this problem, we use the following inequality based on (Jaakkola and Jordan 2000).

$$p(y_n^t|\boldsymbol{x}_n^t,\boldsymbol{w}_t) \geq -e^{y_n^t a}\sigma(\xi_n^t)\left(\frac{a+\xi_n^t}{2} + h(\xi_n^t)(a^2 - {\xi_n^t}^2)\right), \quad (12)$$

where $a := \boldsymbol{w}_t^\top \boldsymbol{x}_n^t$, $\xi_n^t \in \mathbb{R}$ is a parameter that is associated with each training data $(\boldsymbol{x}_n^t, y_n^t)$ and determines the accuracy of the approximation, and $h(\xi_n^t) := \frac{1}{2\xi_n^t}\left(\sigma(\xi_n^t) - \frac{1}{2}\right)$. By substituting the term on the right side of (12) to the lower bound $L(q)$, we obtain a new lower bound $L(q, \xi)$. Note that since $L(q) \geq L(q, \xi)$ for all q and ξ_n^t, increasing the value of $L(q, \xi)$ leads to an increased value of $L(q)$. By calculating the derivatives of $L(q, \xi)$ with respect to $q(w_{t,i})$, $q(A_k)$, $q(\gamma_k)$, $q(\theta)$, $q(\theta_0)$, and ξ_n^t, we find that the variational posterior becomes as follows.

$$\begin{aligned}
q(w_{t,i}) &= \mathcal{N}(w_{t,i}|\eta_{t,i}, \lambda_{t,i}^{-1}), \quad t = 1, \ldots, T,\ i = 1, \ldots, d,\\
q(\boldsymbol{\alpha}_k) &= \mathcal{N}(\boldsymbol{\alpha}_k|\boldsymbol{\mu}_k, \boldsymbol{S}_k^{-1}), \quad k = 0, \ldots, m,\\
q(\gamma_k) &= \mathrm{Gam}(\gamma_k|a_k^\gamma, b_k^\gamma), \quad k = 0, \ldots, m,\\
q(\theta) &= \mathrm{Gam}(\theta|u^\theta, v^\theta),\\
q(\theta_0) &= \mathrm{Gam}(\theta_0|u^{\theta_0}, v^{\theta_0}), \quad (13)
\end{aligned}$$

where $\eta_{t,i} \in \mathbb{R}, \lambda_{t,i}, a_k^\gamma, b_k^\gamma, u^\theta, v^\theta, u^{\theta_0}, v^{\theta_0} \in \mathbb{R}_+, \boldsymbol{\mu}_k \in \mathbb{R}^d$ and $\boldsymbol{S}_k \in \mathbb{R}^{d\times d}$. These parameters are determined by

$$\begin{aligned}
\boldsymbol{S}_0 &= \left(\frac{a_0^\gamma}{b_0^\gamma} + (T-m)\frac{u^\theta}{v^\theta}\right)\mathbf{I}_d,\\
\boldsymbol{\mu}_0 &= \boldsymbol{S}_0^{-1}\frac{u^\theta}{v^\theta}\sum_{t=1}^{T}\left(\boldsymbol{\eta}_t - \sum_{k=1}^{m}\mathrm{dg}(\boldsymbol{\mu}_k)\boldsymbol{\eta}_{t-k}\right),\\
\boldsymbol{S}_k &= \frac{a_k^\gamma}{b_k^\gamma}\mathbf{I}_d + \frac{u^\theta}{v^\theta}\sum_{t=m+1}^{T}\mathrm{dg}(\boldsymbol{\eta}_{t-k})^2 + \boldsymbol{\Lambda}_{t-k}^{-1},\\
\boldsymbol{\mu}_k &= \boldsymbol{S}_k^{-1}\frac{u^\theta}{v^\theta}\sum_{t=m+1}^{T}\mathrm{dg}\left(\boldsymbol{\eta}_t - \sum_{l\neq k}\mathrm{dg}(\boldsymbol{\mu}_l)\boldsymbol{\eta}_{t-l} - \boldsymbol{\mu}_0\right)\boldsymbol{\eta}_{t-k},\\
a_k^\gamma &= a_k + \frac{1}{2}d, \quad b_k^\gamma = b_k + \frac{1}{2}(\|\boldsymbol{\mu}_k\|^2 + \mathrm{Tr}(\boldsymbol{S}_k^{-1})),\\
&\qquad\qquad k = 1, \ldots, m,
\end{aligned}$$

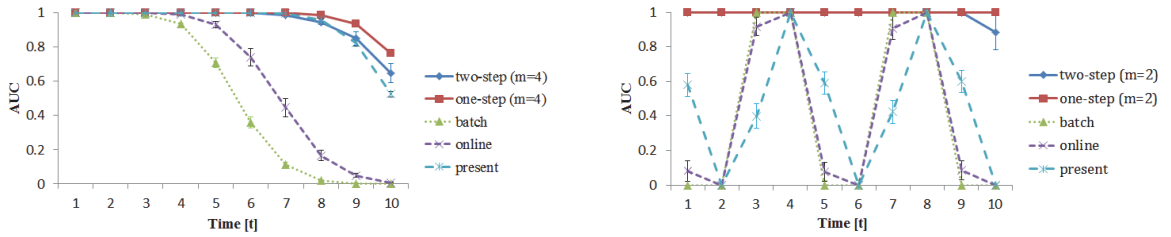
$$\begin{aligned}
u^\theta &= u + \frac{1}{2}(T-m)d,\\
v^\theta &= v + \frac{1}{2}\sum_{t=m+1}^{T}\{\mathrm{Tr}(\boldsymbol{S}_0^{-1} + \boldsymbol{\Lambda}_t^{-1}) + \sum_{l=1}^{m}\boldsymbol{\eta}_{t-l}^\top \boldsymbol{S}_l^{-1}\boldsymbol{\eta}_{t-l}\\
&\quad + \|\boldsymbol{\eta}_t - \sum_{l=1}^{m}\mathrm{dg}(\boldsymbol{\mu}_l)\boldsymbol{\eta}_{t-l} - \boldsymbol{\mu}_0\|^2\\
&\quad + \sum_{l=1}^{m}\mathrm{Tr}(\mathrm{dg}(\boldsymbol{\mu}_l)^2\boldsymbol{\Lambda}_{t-l}^{-1}) + \mathrm{Tr}(\boldsymbol{S}_l^{-1}\boldsymbol{\Lambda}_{t-l}^{-1})\},\\
u^{\theta_0} &= u_0 + \frac{1}{2}md, \quad v^{\theta_0} = v_0 + \frac{1}{2}\sum_{t=1}^{T}\|\boldsymbol{\eta}_t\|^2 + \mathrm{Tr}(\boldsymbol{\Lambda}_t^{-1}),\\
\boldsymbol{\Lambda}_t &= \frac{u^\theta}{v^\theta}\sum_{l=1}^{K_1^t}(\mathrm{dg}(\boldsymbol{\mu}_{m-t+l})^2 + \boldsymbol{S}_{m-t+l}^{-1})\\
&\quad + \frac{u^{\theta_0}}{v^{\theta_0}}\mathbf{I}_d + 2\sum_{n=1}^{N_t}h(\xi_n^t)\mathrm{dg}(\boldsymbol{x}_{\mathrm{n}}^{\mathrm{t}})^2,\\
\eta_{t,i} &= \lambda_{t,i}^{-1}\left(\frac{u^\theta}{v^\theta}\sum_{l=1}^{K_1^t}(\eta_{m+l,i} - \mu_{0,i})\mu_{m+l-t,i}\right.\\
&\quad - \frac{u^\theta}{v^\theta}\sum_{l=1}^{K_1^t}\sum_{l'\neq m+l-t}^{m}\eta_{m+l-l',i}\cdot\mu_{l',i}\cdot\mu_{m+l-t,i}\\
&\quad \left. + \sum_{n=1}^{N_t}\{(y_n^t - \frac{1}{2})x_{n,i}^t - 2h(\xi_n^t)\sum_{l\neq i}\eta_{t,l}x_{n,l}^t x_{n,i}^t\}\right),\\
&\qquad\qquad t = 1, \ldots, m, \quad i = 1, \ldots, d,\\
\boldsymbol{\Lambda}_t &= \frac{u^\theta}{v^\theta}\sum_{l=1}^{K_2^t}(\mathrm{dg}(\boldsymbol{\mu}_l)^2 + \boldsymbol{S}_l^{-1}) + \frac{u^\theta}{v^\theta}\mathbf{I}_d + 2\sum_{n=1}^{N_t}h(\xi_n^t)\mathrm{dg}(\boldsymbol{x}_{\mathrm{n}}^{\mathrm{t}})^2,\\
\eta_{t,i} &= \lambda_{t,i}^{-1}\left(\frac{u^\theta}{v^\theta}\sum_{l=1}^{m}\mu_{l,i}\eta_{t-l,i} + \frac{u^\theta}{v^\theta}\sum_{l=1}^{K_2^t}(\eta_{t+l,i} - \mu_{0,i})\mu_{l,i}\right.\\
&\quad - \frac{u^\theta}{v^\theta}\sum_{l=1}^{K_2^t}\sum_{l'\neq l}^{m}\eta_{t+l-l',i}\cdot\mu_{l',i}\cdot\mu_{l,i} + \frac{u^\theta}{v^\theta}\mu_{0,i}\\
&\quad \left. + \sum_{n=1}^{N_t}\{(y_n^t - \frac{1}{2})x_{n,i}^t - 2h(\xi_n^t)\sum_{l\neq i}\eta_{t,l}x_{n,l}^t x_{n,i}^t\}\right),\\
&\qquad\qquad t = m+1, \ldots, T, \quad i = 1, \ldots, d,\\
(\xi_n^t)^2 &= \boldsymbol{x}_n^{t\top}(\boldsymbol{\Lambda}_t^{-1} + \boldsymbol{\eta}_t\boldsymbol{\eta}_t^\top)\boldsymbol{x}_n^t, t = 1, \ldots, T,\ n = 1, \ldots, N_t
\end{aligned} \quad (14)$$

where $\boldsymbol{\eta}_t := (\eta_{t,1}, \ldots, \eta_{t,d})$, $\boldsymbol{\mu}_t := (\mu_{t,1}, \ldots, \mu_{t,d})$, $\boldsymbol{\Lambda}_t := \mathrm{dg}(\lambda_{\mathrm{t},1}, \ldots, \lambda_{\mathrm{t,d}})$, $K_1^t := \min(t, T-m)$, and $K_2^t := \min(m, T-t)$. We calculate the variational posterior (11) by updating the variables in (13) in turn via the equations (14) until some convergence criterion are satisfied.

Finally, we can predict the classifier parameters $\boldsymbol{w}_t$ for $t = T+1, T+2, \ldots$ using the learned VAR model (2).

Table 1: Average and standard error of AUC over entire test time units. Values in boldface are statistically better than the others (in paired t-test, p=0.05)

	Proposed (two-step)	Proposed (one-step)	Batch	Online	Present
Gradual	0.942±0.018	**0.967±0.010**	0.512±0.062	0.632±0.058	0.930±0.021
Drastic	**0.988±0.010**	**1.000±0.000**	0.400±0.070	0.407±0.066	0.459±0.055
ELEC2	**0.925±0.005**	**0.925±0.005**	0.915±0.005	0.904±0.006	0.898±0.007
Chess.com	**0.850±0.020**	**0.852±0.019**	**0.850±0.018**	0.837±0.018	0.834±0.018



(a) *Gradual* (b) Drastic

Figure 2: Average and standard error of AUC for each test time unit over ten runs.

5 Experiments

We conducted experiments using two synthetic and two real-world data sets to confirm the effectiveness of the proposed method.

5.1 Synthetic Data

We used two synthetic data sets. A sample $(x_1, x_2) \in \mathbb{R}^2$ with label $y \in \{0, 1\}$ and time t was generated from the following distribution $p(\boldsymbol{x}|y, t)$.

$$x_1 = 4.0 \cdot \cos(\pi(t/r + 1 - y)) + \epsilon_1,$$
$$x_2 = 4.0 \cdot \sin(\pi(t/r + 1 - y)) + \epsilon_2, \qquad (15)$$

where ϵ_i for $i = 1, 2$ is a random variable with a standard normal distribution. We can generate various data sets by changing the value of r. Note that the data sets generated by (15) have periodicity (a period is equal to $2r$). *Gradual* is a data set obtained with $r = 20$. The decision boundary of this data gradually changed. *Drastic* is a data set obtained with $r = 2$. The decision boundary of this data drastically changed. In this experiments, we changed t from 1 to 20 and generated 100 samples for each label y and each time t.

5.2 Real-world Data

We used two real-world data sets: *ELEC2* and *Chess.com*. Both data sets are public benchmark data sets for evaluating stream data classification or concept drift. *ELEC2* contains 45312 instances and has eight features. *Chess.com* contains 573 instances and has seven features. For details on these data sets, refer to the paper (Gama et al. 2014). In *ELEC2*, we removed instances that have missing values since missing values are not the focus of our task. We removed instances with draw labels in *Chess.com*. As a result, there were 27549 instances for *ELEC2* and 503 instances for *Chess.com*.

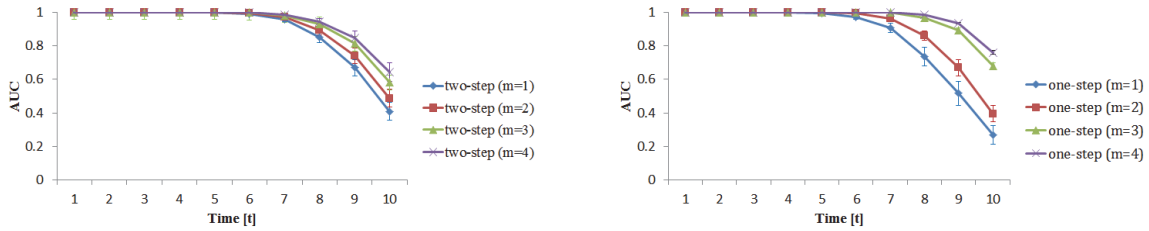
5.3 Setting

In our experiments, to find time variations in the performance of each method, we evaluated AUC by using data until a certain time unit for training, and the remaining for testing. For synthetic data sets, we created ten different sets every synthetic data set. We set $T = 10$ and the remaining ten time units as test data. For *ELEC2*, we set two weeks as one time unit, and then $T = 29$ and the remaining ten time units as test data. For *Chess.com*, we set 20 games as one time unit, $T = 15$ and the remaining ten time units for test data. For the real-world data sets, we chose 80% of instances randomly every time unit to create different training sets (five sets) and evaluated the average AUC by using these sets.

To demonstrate the effectiveness of the proposed method, we compared it with several methods: online logistic regression (*online*), batch logistic regression (*batch*) and present logistic regression (*present*). *Batch* learns a classifier by using all training data D at once. *Online* learns a classifier by maximizing the log posterior (4) in turn from $\boldsymbol{w}_1$ to $\boldsymbol{w}_T$. *Present* is a method that learns a classifier with only recent training data D_T. We chose the regularization parameter for these three methods from $\{10^{-1}, 1, 10^1\}$ in terms of which average AUC over the entire test periods was the best. With the proposed method, we set the hyperparameters as $a_k = u = u_0 = 1$, $b_k = v = v_0 = 0.1$ for all data sets and fixed m as 3 for the real-world data sets from preliminary experiments. In addition, the regularization parameter for the two-step algorithm was set to the same value for *online*.

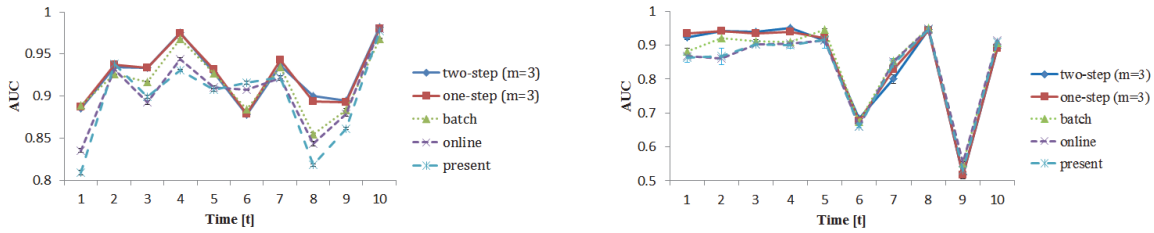
5.4 Results

Table 1 shows averages and standard errors of AUC over the entire test time units for all data sets. For all data sets, the proposed method (especially *one-step*) achieved the highest AUC. This result means that the proposed method captured



(a) two-step learning algorithm (b) one-step learning algorithm

Figure 3: Average and standard error of AUC with the proposed method for each test time unit over ten runs.



(a) *ELEC2* (b) *Chess.com*

Figure 4: Average and standard error of AUC for each test time unit over five runs.

a time-evolving decision boundary better compared with the others.

Figure 2 shows the averages and standard errors of AUC for each test time unit in the synthetic data sets. For *Gradual*, the AUC of *batch* and *online* rapidly decreased over time since these methods do not have a mechanism to capture the time change of a decision boundary. However, the proposed method relatively maintained the classification performance. For *Drastic*, the AUC of the proposed method was almost one at every time, though the AUC of *batch*, *online*, and *present* unsteadily varied from zero to one over time. We confirmed that the proposed method can work in a non-stationary environment effectively.

Figure 3 shows how the performance of the proposed method changes against the hyperparameter m with *Gradual*. The AUC of the proposed method was improved by increasing the value of m. This is because the proposed model become more flexible by increasing the values of m.

Figure 4 shows the averages and standard errors of AUC for each test time unit in the real-world data sets. The proposed method outperformed the others expect t=6 in *ELEC2*. For *Chess.com*, the proposed method achieved the highest AUC until t=4. However, there was not much difference from the others after t=5. In general, it is known that long-time prediction is a very difficult task in time-series analysis. Thus, this result is in accordance with this fact. Long-time prediction is a future challenge.

We compared the proposed two-step algorithm and proposed one-step one with *ELEC2*. Figure 5 shows the averages and standard errors of AUC with different numbers of training data at time t. Here, we used the same hyperparameters used before. *One-step* provided better classification performance than did *two-step* when the number of training

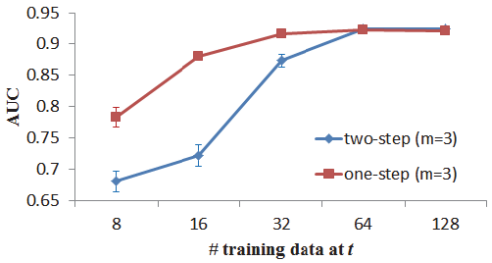


Figure 5: Average and standard error of AUC by the proposed method with different numbers of training data.

data was small. Since *two-step* learns the time-series model after the classifiers learned, the parameters of the time-series model are sensitive to the learned classifiers, which are difficult to learn with a small number of training data. In comparison, *one-step* learned the classifiers and a time-series model simultaneously and therefore become more robust.

6 Conclusion and Future Work

We proposed probabilistic models for predicting future classifiers given labeled data with timestamps collected until the current time. We developed two learning algorithms for the proposed model on the basis of variational Bayesian inference. In experiments, we confirmed that the proposed method can reduce the deterioration of the classification ability better compared with existing methods. As future work, we will apply a non-linear time-series model to the proposed framework for data's dynamics is non-linear.

References

Abdallah, Z. S.; Gaber, M. M.; Srinivasan, B.; and Krishnaswamy, S. 2012. Streamar: incremental and active learning with evolving sensory data for activity recognition. In *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, volume 1, 1163–1170. IEEE.

Babcock, B.; Babu, S.; Datar, M.; Motwani, R.; and Widom, J. 2002. Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 1–16. ACM.

Bishop, C. M. 2006. *Pattern recognition and machine learning*. springer.

Brzezinski, D., and Stefanowski, J. 2014. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *Neural Networks and Learning Systems, IEEE Transactions on* 25(1):81–94.

Crammer, K.; Dredze, M.; and Pereira, F. 2009. Exact convex confidence-weighted learning. In *Advances in Neural Information Processing Systems*, 345–352.

Crammer, K.; Kulesza, A.; and Dredze, M. 2009. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems*, 414–422.

Dyer, K. B.; Capo, R.; and Polikar, R. 2014. Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. *Neural Networks and Learning Systems, IEEE Transactions on* 25(1):12–26.

Fdez-Riverola, F.; Iglesias, E. L.; Díaz, F.; Méndez, J. R.; and Corchado, J. M. 2007. Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications* 33(1):36–48.

Freund, Y.; Schapire, R. E.; et al. 1996. Experiments with a new boosting algorithm. In *ICML*, volume 96, 148–156.

Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; and Bouchachia, A. 2014. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)* 46(4):44.

Jaakkola, T. S., and Jordan, M. I. 2000. Bayesian parameter estimation via variational methods. *Statistics and Computing* 10(1):25–37.

Klinkenberg, R. 2004. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis* 8(3):281–300.

Kolter, J. Z., and Maloof, M. A. 2007. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research* 8:2755–2790.

Koychev, I. 2000. Gradual forgetting for adaptation to concept drift. Proceedings of ECAI 2000 Workshop on Current Issues in Spatio-Temporal Reasoning,.

Lampert, C. H. 2015. Predicting the future behavior of a time-varying probability distribution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 942–950.

Lütkepohl, H. 2005. *New introduction to multiple time series analysis*. Springer Science & Business Media.

Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on* 22(10):1345–1359.

Roberts, S.; Osborne, M.; Ebden, M.; Reece, S.; Gibson, N.; and Aigrain, S. 2013. Gaussian processes for time-series modeling. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 371(1984):20110550.

Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* 65(6):386.

Shimodaira, H. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference* 90(2):227–244.

Wang, H.; Fan, W.; Yu, P. S.; and Han, J. 2003. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 226–235. ACM.

Zhang, P.; Zhu, X.; and Guo, L. 2009. Mining data streams with labeled and unlabeled training examples. In *Data Mining, 2009, ICDM'09, Ninth IEEE International Conference on*, 627–636. IEEE.

Zhu, X.; Zhang, P.; Lin, X.; and Shi, Y. 2010. Active learning from stream data using optimal weight classifiers ensemble. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 40(6):1607–1621.

Zliobaite, I.; Bifet, A.; Pfahringer, B.; and Holms, G. 2014. Active learning with drifting streaming data. *Neural Networks and Learning Systems, IEEE Transactions on* 25(1):27–39.