Learning Dynamics of Decision Boundaries without Additional Labeled Data

Atsutoshi Kumagai
NTT Secure Platform Laboratories
atsutoshi.kumagai.ht@hco.ntt.co.jp

Tomoharu Iwata
NTT Communication Science Laboratories
tomoharu.iwata.gy@hco.ntt.co.jp

ABSTRACT

We propose a method for learning the dynamics of the decision boundary to maintain classification performance without additional labeled data. In various applications, such as spam-mail classification, the decision boundary dynamically changes over time. Accordingly, the performance of classifiers deteriorates quickly unless the classifiers are retrained using additional labeled data. However, continuously preparing such data is quite expensive or impossible. The proposed method alleviates this deterioration in performance by using newly obtained unlabeled data, which are easy to prepare, as well as labeled data collected beforehand. With the proposed method, the dynamics of the decision boundary is modeled by Gaussian processes. To exploit information on the decision boundaries from unlabeled data, the low-density separation criterion, i.e., the decision boundary should not cross high-density regions, but instead lie in low-density regions, is assumed with the proposed method. We incorporate this criterion into our framework in a principled manner by introducing the entropy posterior regularization to the posterior of the classifier parameters on the basis of the generic regularized Bayesian framework. We developed an efficient inference algorithm for the model based on variational Bayesian inference. The effectiveness of the proposed method was demonstrated through experiments using two synthetic and four real-world data sets.

CCS CONCEPTS

• **Computing methodologies → Transfer learning**;

KEYWORDS

Transfer learning, Semi-supervised learning, Concept drift

ACM Reference Format:
Atsutoshi Kumagai and Tomoharu Iwata. 2018. Learning Dynamics of Decision Boundaries without Additional Labeled Data. In *KDD 2018: 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219967

KDD 2018, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219967

1 INTRODUCTION

The decision boundary dynamically changes over time in certain applications. For example, in web-site classification, malicious web sites are uninterruptedly created to scam users. Therefore, the decision boundary that classifies a web site into malicious or not can vary over time [33]. In activity recognition using sensor data, the decision boundary can vary over time since user activity patterns dynamically change [1]. In recommender systems, the decision boundary for presenting goods or information suitable for users can vary over time because the interests of the users dynamically change [4]. When we do not retrain classifiers for tasks in which the decision boundary evolves over time, classification performance deteriorates quickly [11].

Many methods have been proposed for retraining classifiers to maintain classification performance, such as online learning [7, 46], forgetting algorithms [23], time-window methods [2], ensemble learning [14, 24, 45] and active learning [17, 52]. These methods require additional labeled data to retrain classifiers. However, it is quite expensive or impossible to continuously prepare labeled data since labels need to be manually annotated by domain experts.

To overcome this problem, methods for predicting future decision boundaries given only labeled data collected until the current time have been recently proposed [26, 28]. These methods learn the dynamics of the decision boundary with given labeled data. By using the learned dynamics, these methods can predict future decision boundaries, which will correctly classify samples obtained in the future, without additional data. Although these methods are designed to maintain classification performance, it is difficult to keep predicting the right decision boundaries for a long time because actual decision boundaries can change unexpectedly in real-world applications.

Though collecting labeled data is difficult, unlabeled data are easier to collect than labeled data since they do not require labels. For example, in web-site classification, newly created web sites would be easily available by crawling and used for learning. Since unlabeled data contain rich information on the decision boundary, semi-supervised learning methods can improve classification performance by using unlabeled data [3, 16, 21, 34]. However, these methods become inaccurate when the decision boundary changes over time.

In this paper, we propose a method for learning the dynamics of the decision boundary to maintain classification performance by using newly obtained unlabeled data as well as labeled data collected beforehand. With the proposed method, a decision boundary is defined by the classifier parameters, and the dynamics of each parameter is modeled by Gaussian processes (GPs), which are non-linear non-parametric regression models [37]. By handling the

dynamics of each parameter, our method can reflect the characteristics of each feature. Although labeled data are directly used for learning the classifier parameters and dynamics of each parameter, how to use unlabeled data is not trivial. To exploit information on the decision boundaries from unlabeled data, the low-density separation criterion, i.e., the decision boundary should not cross high-density regions, but instead lie in low-density regions, is assumed with the proposed method. We incorporate this criterion into our framework in a principled manner by introducing the entropy posterior regularization to the posterior of the classifier parameters on the basis of the generic regularized Bayesian inference (RegBayes) framework [51]. Since unlabeled data contain rich information on the decision boundaries, the proposed method can improve its ability to learn the dynamics of the decision boundary when labeled data are unavailable; thus, maintain classification performance for a long time. The proposed method also involves our developed inference algorithm based on variational Bayesian inference for optimizing classifier parameters and hyperparameters for GPs simultaneously. When the amount of training data is small or long-time prediction is executed, the uncertainty of the predicted classifiers becomes high. By using a Bayesian framework, the proposed method can learn the dynamics of the decision boundary while taking into account this uncertainty, which enables robust classification.

2 RELATED WORK

Retraining the classifiers with labeled data has been widely done to maintain the performance of classifiers. Online learning updates classifiers by using sequentially arriving labeled data [7, 46]. Some online learning methods use Kalman filters [19] for updating classifiers [41, 47]. Forgetting algorithms learn the decision boundary by weighting training data based on their age, which means recent data are influential for learning [2, 23, 25]. Ensemble learning combines multiple base classifiers to create a better classifier [14, 24, 45]. Its mixture weight is determined so that base classifiers that capture a recent decision boundary have a large weight. Some methods are aimed at adapting the time-evolving decision boundary when both labeled and unlabeled data are given sequentially [13, 29, 49]. Active learning based methods determine what data should be labeled using active learning for adaptation [17, 52]. Some studies proposed a framework for online transfer learning [44, 50]. Life-long learning methods use knowledge obtained from previous tasks to efficiently learn classifiers on the new task [8, 38, 48]. Gao et al. [12] uses GPs for latent variables of samples to cope with concept drift. All these methods continuously require additional labeled data for adapting the time-varying decision boundary. However, the task we investigate in this study is adapting the time-evolving decision boundary without additional labeled data.

Methods have been proposed for predicting future decision boundaries by using labeled data collected until the current time [26, 28]. These methods learn the dynamics of the decision boundary to predict future decision boundaries, which will classify samples collected at a future time. In [26], vector autoregressive models are used to model the linear dynamics of the decision boundary. For capturing the non-linear dynamics, GPs are used as time-series models [28]. Although these methods are designed to maintain classification performance without additional data, it is difficult to maintain it for a long time because the dynamics of the decision boundary sometimes change unexpectedly in real-world applications. The proposed method alleviates this deterioration in performance by using additional unlabeled data. Since the proposed method uses GPs for modeling the dynamics of the decision boundary, it is a natural semi-supervised extension of the method proposed in [28].

The proposed method is related to time-series models, which are used for analyzing or predicting time-series data [19, 37]. The proposed method uses them for modeling the dynamics of the decision boundary.

Semi-supervised learning methods use both labeled and unlabeled data for learning classifiers [3, 16, 21, 34]. However, it is implicitly assumed that the decision boundary does not change with these methods. Therefore, these methods are not suitable for tasks in which the decision boundary changes over time. The proposed method uses the low-density separation criterion, which has been successfully used in semi-supervised learning [16], to exploit information on the decision boundaries from unlabeled data.

The proposed method is related to transfer learning or domain adaptation [36]. These methods use data in a source domain to solve a related problem in the target domain. In our task, we can regard labeled data collected until a certain time point as data in the source domain, and unlabeled data collected after the certain point as data in the target domain. Importance weighting methods are widely used for transfer learning, which learn classifiers by weighting a training sample with its importance, which is defined by the ratio between the training and test distributions [20, 27, 39]. Many methods find the domain-invariant representations where the two domains are close [31, 32, 35]. Although it is usually assumed that probability distributions by which data are governed are different between the source and target domain with all these methods, each distribution is treated as stationary. This means that each decision boundary also does not change. In contrast, the situation we investigate in this paper is that the decision boundary evolves continuously.

Recently, methods have been proposed to learn classifiers with sequentially arriving unlabeled data. One method uses a generalization of convex hull called α-shape to incorporate unlabeled data into the system [9]. Souza et al. [40] proposed a method that uses a clustering followed by a classification step applied to adapt the time-evolving decision boundary. Importance weighted least squares probabilistic classifiers are proposed for adaptation [43]. One method iteratively adapts object detectors from images to videos by retraining them with automatically discovered target examples [42]. Some domain adaptation methods incrementally find new domain-invariant representations for the source and time-varying target domains [6, 18]. These methods sequentially update classifiers by using newly obtained unlabeled data. However, they do not explicitly model the dynamics of the decision boundary. Although unlabeled data have rich information on the decision boundary, their classifiers tend to be inaccurate since unlabeled data do not have labels and estimated errors tend to be accumulated. The proposed method alleviates this risk by modeling the dynamics of the decision boundary, which also work as the regularization

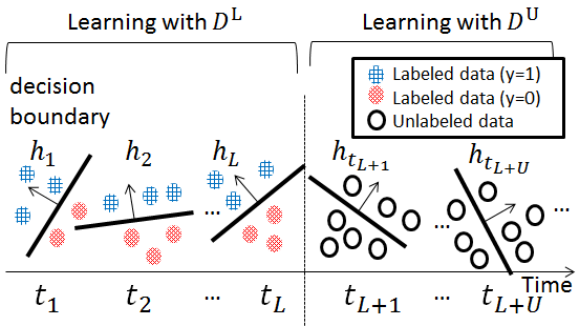


Figure 1: Illustration of our method

for preventing the learning of the decision boundaries far from the estimated dynamics. Accordingly, by exploiting useful information on the decision boundaries from both the dynamics and unlabeled data, the proposed method can effectively maintain classification performance compared with these methods, which is discussed in Section 4.

To the best of our knowledge, the proposed method is the first attempt at adapting the time-varying decision boundaries by learning the dynamics of the decision boundary with newly obtained unlabeled data as well as labeled data collected beforehand.

3 PROPOSED METHOD

We introduce the notations used in this paper and define the task we investigate. Let $\mathcal{D}_t^{\mathrm{L}}:=\{(\boldsymbol{x}_n^t, y_n^t)\}_{n=1}^{N_t}$ be a set of labeled data collected at time t, where $\boldsymbol{x}_n^t \in \mathbb{R}^D$ is the D-dimensional feature vector of the n-th sample at time t, $y_n^t \in \{0,1\}$ is its class label, and N_t is the number of labeled data collected at time t. Although the proposed method can be applied to multi-class classification straightforwardly, we treat binary-classification for simplicity. The term $\boldsymbol{t}^{\mathrm{L}}:=(t_1,\ldots,t_L)$, where $t_1 < t_2 < \cdots < t_L$, denotes the times at which labeled data are collected. $\mathcal{D}_t^{\mathrm{U}}:=\{\boldsymbol{x}_m^t\}_{m=1}^{M_t}$ is a set of unlabeled data (test data) collected at time t, where $\boldsymbol{x}_m^t \in \mathbb{R}^D$ is the D-dimensional feature vector of the m-th sample at time t, and M_t is the number of unlabeled data collected at time t. The term $\boldsymbol{t}^{\mathrm{U}}:=(t_{L+1},\ldots,t_{L+U})$, where $t_L < t_{L+1} < \cdots < t_{L+U}$, denotes the times at which unlabeled data are obtained. Note that all unlabeled data are collected after labeled data are collected, and each sample can be obtained at irregular intervals.

Our goal is to find classifiers $h_t : \mathbb{R}^D \to \{0,1\}$, $\forall t > t_L$ that can precisely classify samples at time t, given a set of labeled data and unlabeled data, $\mathcal{D} := \mathcal{D}^{\mathrm{L}} \bigcup \mathcal{D}^{\mathrm{U}}$, where $\mathcal{D}^{\mathrm{L}} := \{\mathcal{D}_t^{\mathrm{L}}\}_{t\in\boldsymbol{t}^{\mathrm{L}}}$ and $\mathcal{D}^{\mathrm{U}} := \{\mathcal{D}_t^{\mathrm{U}}\}_{t\in\boldsymbol{t}^{\mathrm{U}}}$. The proposed method can predict classifiers at any time t even if there are no labeled and unlabeled training data at that time. Figure 1 illustrates our method. Using labeled data from time t_1 to t_L and unlabeled data from time t_{L+1} to t_{L+U}, the decision boundary for each time, which is defined by classifier h_t, and the dynamics of the decision boundary is learned.

3.1 Probabilistic Model for Dynamics of Decision Boundary

Our probabilistic model assumes that the probability of label y_n^t given feature vector $\boldsymbol{x}_n^t$ is modeled by logistic regression as follows,

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

where $\boldsymbol{w}_t = (w_{t1},\ldots,w_{tD}) \in \mathbb{R}^D$ is a parameter vector of h_t, σ is the sigmoid function, and $\top$ is transposition. Note that the probability that $y_n^t = 0$, $p(y_n^t = 0|\boldsymbol{x}_n^t, \boldsymbol{w}_t)$, is equal to $1 - p(y_n^t = 1|\boldsymbol{x}_n^t, \boldsymbol{w}_t)$.

Our probabilistic model assumes that the d-th component of classifier parameter w_{td} is generated by mapping input time t using a non-linear function,

$$w_{td} = f_d(t) + \epsilon_d, \tag{2}$$

where f_d is the non-linear function for the d-th feature, ϵ_d is Gaussian noise, $\epsilon_d \sim \mathcal{N}(0, \eta_d^2)$, and η_d^2 is a variance parameter. We use a GP for a prior distribution of f_d. Specifically, given any finite subset of input times $\boldsymbol{t} := (t_1,\ldots,t_{L+U})$, the prior on the corresponding outputs $\boldsymbol{f}_d := (f_d(t_1),\ldots,f_d(t_{L+U}))$ is represented as a zero-mean multivariate Gaussian distribution on $\boldsymbol{t}$,

$$p(\boldsymbol{f}_d) = \mathcal{N}(\boldsymbol{f}_d|\boldsymbol{0}, \boldsymbol{K}_d), \tag{3}$$

where the covariate matrix $\boldsymbol{K}_d \in \mathbb{R}^{(L+U)\times(L+U)}$ is constructed from a covariance (or kernel) function k_d , that is, the (t,t') element of $\boldsymbol{K}_d$ is the value of k_d between t and t', $[\boldsymbol{K}_d]_{tt'} := k_d(t,t')$. Usually the kernel function depends on certain hyperparameters that control the smoothness property of f_d. In this paper, we use the following Gaussian kernel with the addition of constant and linear terms as the kernel function,

$$k_d(t,t') = \beta_d^2 \exp\left(-\frac{1}{2}\alpha_d^2|t-t'|^2\right) + \gamma_d^2 + \zeta_d^2 tt', \tag{4}$$

where α_d, β_d, γ_d, and ζ_d are kernel hyperparameters. This kernel function is widely used for GP regression and effectively describes the time series with a smooth shape [5]. By integrating out $\boldsymbol{f}_d$, we obtain the probability of the d-th component of the classifier at the input times $\boldsymbol{t}$, $\boldsymbol{w}_{\cdot d} := (w_{t_1 d},\ldots,w_{t_{L+U} d}) \in \mathbb{R}^{L+U}$, as follows,

$$p(\boldsymbol{w}_{\cdot d}) = \int p(\boldsymbol{w}_{\cdot d}|\boldsymbol{f}_d)p(\boldsymbol{f}_d)d\boldsymbol{f}_d = \mathcal{N}(\boldsymbol{w}_{\cdot d}|\boldsymbol{0}, \boldsymbol{C}_d), \tag{5}$$

where the covariance matrix $\boldsymbol{C}_d \in \mathbb{R}^{(L+U)\times(L+U)}$ is defined as follows,

$$c_d(t,t') := k_d(t,t') + \delta_{tt'}\eta_d^2, \tag{6}$$

where $\delta_{tt'} = 1$ if $t = t'$, and $\delta_{tt'} = 0$ otherwise.

The joint distribution of labeled data $\mathcal{D}^{\mathrm{L}}$ and classifier parameters at times $\boldsymbol{t}$, $\boldsymbol{W} := (\boldsymbol{w}_{t_1},\ldots,\boldsymbol{w}_{t_{L+U}})$, is written by

$$\begin{aligned} p(\mathcal{D}^{\mathrm{L}}, \boldsymbol{W}; \boldsymbol{\theta}) &= p(\mathcal{D}^{\mathrm{L}}|\boldsymbol{W})p(\boldsymbol{W};\boldsymbol{\theta}) \\ &= \prod_{t=t_1}^{t_L}\prod_{n=1}^{N_t} p(y_n^t|\boldsymbol{x}_n^t, \boldsymbol{w}_t) \cdot \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{w}_{\cdot d}|\boldsymbol{0}, \boldsymbol{C}_d), \end{aligned} \tag{7}$$

where $\boldsymbol{\theta} := (\alpha_1,\ldots,\alpha_D,\beta_1,\ldots,\beta_D,\gamma_1,\ldots,\gamma_D,\zeta_1,\ldots,\zeta_D,\eta_1,\ldots,\eta_D)$, and $p(\boldsymbol{W};\boldsymbol{\theta}) = \prod_{d=1}^{D} p(\boldsymbol{w}_{\cdot d})$. When α is infinitely large, and γ and ζ are zero, the classifier parameters in different time points are independent. This corresponds to learning classifiers for each time point independently. When α and ζ are zero, and β or γ are infinitely large, the classifier parameters are almost constant over time. This

corresponds to learning a single classifier for all time points by using all data and ignoring their time stamps. Our probabilistic model can represent various dynamics of the decision boundary by varying the values of kernel hyperparameters. Although many time-series models such as vector autoregressive models require data to be collected at regular intervals without absence, the proposed model does not require this owing to the characteristics of GPs. Note that though we employ the logistic regression as classifiers, it is possible to use other classifiers such as neural networks [15] and boosting [10] for our framework.

3.2 RegBayes Framework with Entropy Posterior Regularization

To obtain the posterior distribution $p(\boldsymbol{W}|\mathcal{D}^{\mathrm{L}};\boldsymbol{\theta})$ and the hyperparameters $\boldsymbol{\theta}$, we need to calculate the model evidence $p(\mathcal{D}^{\mathrm{L}};\boldsymbol{\theta}) = \int p(\mathcal{D}^{\mathrm{L}},\boldsymbol{W};\boldsymbol{\theta})d\boldsymbol{W}$. However, unfortunately, this integral is intractable. Instead, we treat the evidence lower bound (ELBO), which is the lower bound of $p(\mathcal{D}^{\mathrm{L}};\boldsymbol{\theta})$. Specifically, the ELBO is defined as

$$\mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta}) := \int q(\boldsymbol{W}) \log \frac{p(\mathcal{D}^{\mathrm{L}},\boldsymbol{W};\boldsymbol{\theta})}{q(\boldsymbol{W})} d\boldsymbol{W}, \tag{8}$$

where $q(\boldsymbol{W})$ is an approximate posterior distribution of the posterior distribution $p(\boldsymbol{W}|\mathcal{D}^{\mathrm{L}};\boldsymbol{\theta})$. We can obtain $q(\boldsymbol{W})$ by maximizing the ELBO with respect to q since minimizing the Kullback-Leibler divergence between $q(\boldsymbol{W})$ and $p(\boldsymbol{W}|\mathcal{D}^{\mathrm{L}};\boldsymbol{\theta})$ is equivalent to maximizing the ELBO. The hyperparameters $\boldsymbol{\theta}$ can be estimated by maximizing the ELBO with respect to $\boldsymbol{\theta}$. Note that this ELBO does not depend on unlabeled data $\mathcal{D}^{\mathrm{U}}$.

To incorporate useful information of the unlabeled data into the model, we use the entropy minimization principle, which encourages the decision boundaries to lie in low-density regions. Specifically, we define the entropy posterior regularization term for the classifier parameter $\boldsymbol{w}_t$ for $t \in \boldsymbol{t}^{\mathrm{U}}$ as,

$$R_t(q) := \int \sum_{m=1}^{M_t} H(p(y|\boldsymbol{x}_m^t,\boldsymbol{w}_t))q(\boldsymbol{W})d\boldsymbol{W}, \tag{9}$$

where $H(\cdot)$ is the entropy function of class-conditional distribution defined as,

$$H(p(y|\boldsymbol{x}_m^t,\boldsymbol{w}_t)) := -\sum_{y\in\{0,1\}} p(y|\boldsymbol{x}_m^t,\boldsymbol{w}_t)\log p(y|\boldsymbol{x}_m^t,\boldsymbol{w}_t). \tag{10}$$

Since (10) takes a small value when the sample $\boldsymbol{x}_m^t$ is located far from the decision boundary $\boldsymbol{w}_t$, the decision boundary minimizing this entropy function is encouraged to pass through the low density regions. The entropy minimization principle is widely used in semi-supervised learning and shows good performance in various tasks; therefore, we chose this regularization to incorporate unlabeled data into the model. Of course, it is possible to use other regularizations, such as manifold regularization, in our framework [3]. Since there are no labeled data at times $\boldsymbol{t}^{\mathrm{U}}$, the uncertainty of the classifier parameters $\{\boldsymbol{w}_t\}_{t\in\boldsymbol{t}^{\mathrm{U}}}$ becomes high. To handle this uncertainty appropriately, the proposed method takes the expectation of the entropy function with respect to $\boldsymbol{W}$ in (9). By minimizing (9) with respect to $q(\boldsymbol{w}_t)$, the classifier parameter $\boldsymbol{w}_t$ is learned so as to pass through the low-density regions of the unlabeled samples $\mathcal{D}_t^{\mathrm{U}}$ while following the dynamics of the decision boundary.

By using the sum of the entropy posterior regularization (9) as a posterior regularization term $R(q) := \sum_{t=t_{L+1}}^{t_{L+U}} R_t(q)$, we consider the following RegBayes framework:

$$\begin{aligned}
&\max_{q(\boldsymbol{W}),\boldsymbol{\theta}} \mathcal{L}(q;\boldsymbol{\theta}) := \mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta}) - \frac{\rho}{M}R(q) \\
&= \max_{q(\boldsymbol{W}),\boldsymbol{\theta}} \int q(\boldsymbol{W}) \log \frac{p(\mathcal{D}^{\mathrm{L}},\boldsymbol{W};\boldsymbol{\theta})}{q(\boldsymbol{W})} d\boldsymbol{W} \\
&- \frac{\rho}{M}\int \sum_{t=t_{L+1}}^{t_{L+U}} \sum_{m=1}^{M_t} H(p(y|\boldsymbol{x}_m^t,\boldsymbol{w}_t))q(\boldsymbol{W})d\boldsymbol{W},
\end{aligned} \tag{11}$$

where $M := \sum_{t=t_{L+1}}^{t_{L+U}} M_t$ and ρ is the positive regularization parameter. The GP prior $p(\boldsymbol{W};\boldsymbol{\theta})$ has the effect of connecting the classifier parameters $\{\boldsymbol{w}_t\}_{t\in\boldsymbol{t}^{\mathrm{L}}}$, which are mainly estimated with labeled data, and the classifier parameters $\{\boldsymbol{w}_t\}_{t\in\boldsymbol{t}^{\mathrm{U}}}$, which are mainly estimated with unlabeled data. Note that, when $\rho = 0$, the entropy posterior regularization is unused and the objective function is equivalent to those in [28] except that the GP prior depends on $\{\boldsymbol{w}_t\}_{t\in\boldsymbol{t}^{\mathrm{U}}}$ as well as $\{\boldsymbol{w}_t\}_{t\in\boldsymbol{t}^{\mathrm{L}}}$. Thus, our method is a natural semi-supervised extension of the method proposed in [28].

3.3 Inference

We present our efficient inference algorithm for the proposed model based on variational Bayesian inference. We assume that variational posterior $q(\boldsymbol{W})$ can be factorized as follows:

$$q(\boldsymbol{W}) = \prod_{t=t_1}^{t_{L+U}} \prod_{d=1}^{D} q(w_{td}). \tag{12}$$

For the case of $t \in \boldsymbol{t}^{\mathrm{U}}$, we assume the functional form of $q(w_{td})$ as a Gaussian distribution, $q(w_{td}) = \mathcal{N}(w_{td}|\mu_{td},\sigma_{td}^2)$, where μ_{td} is a mean and σ_{td}^2 is a variance. In the variational Bayesian inference, the proposed method maximizes the objective function in (11) by iteratively updating each variational posterior $q(w_{td})$ and the kernel hyper parameter $\boldsymbol{\theta}$.

First, we consider deriving the update rules for $\{q(\boldsymbol{w}_t)\}_{t\in\boldsymbol{t}^{\mathrm{L}}}$. Since the variational posterior $\{q(\boldsymbol{w}_t)\}_{t\in\boldsymbol{t}^{\mathrm{L}}}$ depends only on $\mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta})$ in (11), we can derive the update rules for $\{q(\boldsymbol{w}_t)\}_{t\in\boldsymbol{t}^{\mathrm{L}}}$ by calculating the derivatives of $\mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta})$ with respect to $\{q(\boldsymbol{w}_t)\}_{t\in\boldsymbol{t}^{\mathrm{L}}}$. However, it is impossible because of the non-conjugacy of $p(y_n^t|\boldsymbol{x}_n^t,\boldsymbol{w}_t)$. To cope with this problem, we use the following inequality [5],

$$p(y_n^t|\boldsymbol{x}_n^t,\boldsymbol{w}_t) \geq \sigma(\xi_n^t)\exp\left(y_n^t a_n^t - \frac{a_n^t+\xi_n^t}{2} - h(\xi_n^t)({a_n^t}^2 - {\xi_n^t}^2)\right), \tag{13}$$

where $a_n^t := \boldsymbol{w}_t^\top \boldsymbol{x}_n^t$, $\xi_n^t \in \mathbb{R}$ is a parameter that is associated with each sample $(\boldsymbol{x}_n^t, y_n^t)$ 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 (13) with $\mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta})$, we obtain a new ELBO $\mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta},\boldsymbol{\xi})$. Since $\mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta},\boldsymbol{\xi})$ is the lower bound of $\mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta})$, increasing the value of $\mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta},\boldsymbol{\xi})$ leads to an increased value of $\mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta})$. By calculating the derivatives of $\mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta},\boldsymbol{\xi})$ with respect to $\{q(\boldsymbol{w}_t)\}_{t\in\boldsymbol{t}^{\mathrm{L}}}$ and ξ_n^t, we find that $q(w_{td})$ for $t \in \boldsymbol{t}^{\mathrm{L}}$ and $d = 1,\ldots,D$ takes the following form,

$$q(w_{td}) = \mathcal{N}(w_{td}|\mu_{td},\lambda_{td}^{-1}). \tag{14}$$

The update rules for μ_{td}, λ_{td}, and ξ_t^n are given by

$$\begin{aligned}\mu_{td} \leftarrow & \lambda_{td}^{-1}\Bigg(\sum_{n=1}^{N_t}\{(y_n^t-\frac{1}{2})x_{nd}^t - 2h(\xi_n^t)\sum_{l\neq d}\mu_{tl}x_{nl}^t x_{nd}^t\} \\ & -\sum_{s\neq t}[C_d^{-1}]_{ts}\mu_{sd}\Bigg), \\ \lambda_{td} \leftarrow & [C_d^{-1}]_{tt} + 2\sum_{n=1}^{N_t}h(\xi_n^t)(x_{nd}^t)^2, \\ (\xi_n^t)^2 \leftarrow & \boldsymbol{x}_n^{t\top}(\boldsymbol{\Lambda}_t^{-1}+\boldsymbol{\mu}_t\boldsymbol{\mu}_t^\top)\boldsymbol{x}_n^t, \end{aligned} \tag{15}$$

where $\boldsymbol{\mu}_t := (\mu_{t1},\ldots,\mu_{tD})$, and $\boldsymbol{\Lambda}_t := \mathrm{diag}(\lambda_{t1},\ldots,\lambda_{tD})$, in which $\mathrm{diag}(\boldsymbol{x})$ means a diagonal matrix whose diagonal elements are $\boldsymbol{x}$.

Second, we consider updating $\{q(\boldsymbol{w}_t)\}_{t\in\boldsymbol{t}^{\mathrm{U}}}$. Since the entropy posterior regularization term $R(q)$ is intractable, it is impossible to analytically obtain the derivative of the objective function $\mathcal{L}(q;\boldsymbol{\theta},\boldsymbol{\xi}) := \mathcal{L}^{\mathrm{L}}(q;\boldsymbol{\theta},\boldsymbol{\xi}) - \frac{\rho}{M}R(q)$ with respect to $\{q(\boldsymbol{w}_t)\}_{t\in\boldsymbol{t}^{\mathrm{U}}}$. To solve this problem, we use the reparametrization trick, which is a technique for approximating expectation with a low variance [22]. This trick expresses a continuous random variable $\boldsymbol{w}$ as a deterministic variable $\boldsymbol{w} = g_\phi(\boldsymbol{\epsilon})$, where $\boldsymbol{\epsilon}$ is an auxiliary variable with $p(\boldsymbol{\epsilon})$, g_ϕ is some vector-valued function parameterized by ϕ. By this expression, the expectation with respect to $\boldsymbol{w}$ is transformed to it with respect to $\boldsymbol{\epsilon}$. Since $\boldsymbol{\epsilon}$ is irrelevant for ϕ, the approximated expectation by sampling from $p(\boldsymbol{\epsilon})$ is differentiable with respect to ϕ. Since we assumed that $q(\boldsymbol{w}_t)$ for $t \in \boldsymbol{t}^{\mathrm{U}}$ is the Gaussian distribution, the reparametrization trick can be applied to the entropy posterior regularization term $R(q)$. Specifically, by using the equation: $\boldsymbol{w}_t^{(j)} := \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \odot \boldsymbol{\epsilon}_t^{(j)}$, $\boldsymbol{\epsilon}_t^{(j)} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, where $\odot$ is an element-wise product, $\mathbf{I}$ is the identity matrix, and $\boldsymbol{\sigma}_t := (\sigma_{t1},\ldots,\sigma_{tD})$, the terms depend on $\{\boldsymbol{w}_t\}_{t\in\boldsymbol{t}^{\mathrm{U}}}$ in the objective function $\mathcal{L}(q;\boldsymbol{\theta},\boldsymbol{\xi})$ are approximated as follows:

$$\begin{aligned}\mathcal{L}^{\mathrm{U}}(q) = & \frac{1}{J}\frac{\rho}{M}\sum_{t=t_{L+1}}^{t_{L+U}}\sum_{j=1}^{J}\sum_{y\in\{0,1\}}\sum_{m=1}^{M_t}p(y|\boldsymbol{x}_m^t,\boldsymbol{w}_t^{(j)})\log p(y|\boldsymbol{x}_m^t,\boldsymbol{w}_t^{(j)}) \\ & -\frac{1}{2}\sum_{d=1}^{D}\sum_{t=t_{L+1}}^{t_{L+U}}\left([C_d^{-1}]_{tt}(-\mu_{td}^2+\sigma_{td}^2) + 2\sum_{s=t_1}^{t_{L+U}}[C_d^{-1}]_{st}\mu_{sd}\mu_{td}\right) \\ & +\frac{1}{2}\sum_{d=1}^{D}\sum_{t=t_{L+1}}^{t_{L+U}}(1+\log\sigma_{td}^2), \end{aligned} \tag{16}$$

where J is the number of samples of the reparametrization trick. We introduce new variables υ_{td} that satisfy the equation $\upsilon_{td} = \log\sigma_{td}$. By maximizing (16) with respect to the parameters $\boldsymbol{\mu}_t$ and $\boldsymbol{\upsilon}_t := (\upsilon_{t1},\ldots,\upsilon_{tD})$, we obtain the updated distributions $q(\boldsymbol{w}_t)$ for $t \in \boldsymbol{t}^{\mathrm{U}}$. To achieve this, we use the the quasi-Newton method [30], which requires the information of the gradient of (16). These gradients with respect to μ_{td} and υ_{td} are expressed as

$$\begin{aligned}\frac{\partial\mathcal{L}^{\mathrm{U}}(q)}{\partial\mu_{td}} &= \frac{1}{J}\frac{\rho}{M}\sum_{j,m}F(\boldsymbol{w}_t^{(j)},\boldsymbol{x}_m^t)x_{md}^t - \sum_{s=t_1}^{t_{L+U}}[C_d^{-1}]_{ts}\mu_{sd}, \\ \frac{\partial\mathcal{L}^{\mathrm{U}}(q)}{\partial\upsilon_{td}} &= \frac{1}{J}\frac{\rho}{M}\sum_{j,m}F(\boldsymbol{w}_t^{(j)},\boldsymbol{x}_m^t)x_{md}^t\sigma_{td}\epsilon_{td}^{(j)} - [C_d^{-1}]_{tt}\sigma_{td}^2 + 1, \end{aligned} \tag{17}$$

where $F(\boldsymbol{w}_t^{(j)},\boldsymbol{x}_m^t) := \sigma(\boldsymbol{w}_t^{(j)\top}\boldsymbol{x}_m^t)(1-\sigma(\boldsymbol{w}_t^{(j)\top}\boldsymbol{x}_m^t))(\boldsymbol{w}_t^{(j)\top}\boldsymbol{x}_m^t)$.

Finally, we consider updating $\boldsymbol{\theta}$ by maximizing $\mathcal{L}(q;\boldsymbol{\theta},\boldsymbol{\xi})$ with respect to $\boldsymbol{\theta}$ using the quasi-Newton method. The terms depend on $\boldsymbol{\theta}$ in the objective function $\mathcal{L}(q;\boldsymbol{\theta},\boldsymbol{\xi})$ are represented as

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{2}\sum_{d=1}^{D}[\boldsymbol{\mu}_{\cdot d}^\top C_d^{-1}\boldsymbol{\mu}_{\cdot d} + \mathrm{Tr}(C_d^{-1}\Lambda_d^{-1}) + \log(\det(C_d))], \tag{18}$$

where $\boldsymbol{\mu}_{\cdot d} := (\mu_{t_1 d},\ldots,\mu_{t_{L+U}d})$, $\boldsymbol{\Lambda}_d := \mathrm{diag}(\lambda_{t_1 d},\ldots,\lambda_{t_{L+U}d})$, Tr is the trace, and det is the determinant. The gradients of $\mathcal{L}(\boldsymbol{\theta})$ with respect to θ_d, which represents one of the α_d, β_d, γ_d, ζ_d and η_d, are expressed as

$$\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\theta_d} = \frac{1}{2}\boldsymbol{\mu}_{\cdot d}^\top C_d^{-1}\frac{\partial C_d}{\partial\theta_d}C_d^{-1}\boldsymbol{\mu}_{\cdot d} + \frac{1}{2}\mathrm{Tr}\left(C_d^{-1}\frac{\partial C_d}{\partial\theta_d}(C_d^{-1}\Lambda_d^{-1}-\mathbf{I})\right). \tag{19}$$

The gradients of the kernel $[C_d]_{tt'}$ with respect to α_d, β_d, γ_d. ζ_d and η_d are given by

$$\begin{aligned}\frac{\partial[C_d]_{tt'}}{\partial\alpha_d} &= -\alpha_d|t-t'|^2\beta_d^2\exp\left(-\frac{1}{2}\alpha_d^2|t-t'|^2\right), \\ \frac{\partial[C_d]_{tt'}}{\partial\beta_d} &= 2\beta_d\exp\left(-\frac{1}{2}\alpha_d^2|t-t'|^2\right), \\ \frac{\partial[C_d]_{tt'}}{\partial\gamma_d} &= 2\gamma_d,\ \frac{\partial[C_d]_{tt'}}{\partial\zeta_d} = 2\zeta_d tt',\ \frac{\partial[C_d]_{tt'}}{\partial\eta_d} = 2\delta_{tt'}\eta_d. \end{aligned} \tag{20}$$

The $\{q(\boldsymbol{w}_t)\}_{t\in\boldsymbol{t}^{\mathrm{L}}}$ are estimated by updating variables μ_{td}, λ_{td}, and ξ_n^t by using (15), the $\{q(\boldsymbol{w}_t)\}_{t\in\boldsymbol{t}^{\mathrm{U}}}$ are estimated by updating variables $\boldsymbol{\mu}_t$ and $\boldsymbol{\upsilon}_t$ by using the quasi-Newton method with (16) and (17), and the hyperparameters $\boldsymbol{\theta}$ are estimated by using the quasi-Newton method with (18), (19), and (20) in turn until some convergence criteria are satisfied. The pseudo-code of our inference algorithm of the proposed method is presented in Algorithm 1.

3.4 Prediction

We explain how to obtain classifiers of arbitrary time by using the learned model. When we would like to classify samples at time $t_* \in \boldsymbol{t}^{\mathrm{U}}$, we can use the variational posterior $q(\boldsymbol{w}_{t_*})$ as the distribution of the classifier parameter. When classifying sample at time $t_* \notin \boldsymbol{t}$, where there are no labeled and unlabeled training data, we can obtain the corresponding distribution of the classifier parameter $q(\boldsymbol{w}_{t_*})$ in usual GPs manner. Specifically, the probability of $\boldsymbol{w}_{t_*} = (w_{t_*1},\ldots,w_{t_*D})$ is represented as follows:

$$\begin{aligned} p(\boldsymbol{w}_{t_*}) &= \prod_{d=1}^{D}p(w_{t_*d}), \\ p(w_{t_*d}) &= \int p(w_{t_*d}|\boldsymbol{w}_{\cdot d})q(\boldsymbol{w}_{\cdot d})d\boldsymbol{w}_{\cdot d} = \mathcal{N}(w_{t_*d}|m_{t_*d},\sigma_{t_*d}^2), \\ m_{t_*d} &= \boldsymbol{k}_d^\top C_d^{-1}\boldsymbol{\mu}_{\cdot d}, \\ \sigma_{t_*d}^2 &= k_d(t_*,t_*) + \eta_d^2 + \boldsymbol{k}_d^\top\left(C_d^{-1}\Lambda_d^{-1}-\mathbf{I}\right)C_d^{-1}\boldsymbol{k}_d, \end{aligned} \tag{21}$$

where $\boldsymbol{k}_d := (k_d(t_*,t_1),\ldots,k_d(t_*,t_{L+U}))$. For classifying samples at time t_*, we use Bayesian logistic regression, which is a method for classification taking variances of $p(\boldsymbol{w}_{t_*})$ into account [5]. The

Algorithm 1 Inference algorithm of the proposed method

Require: A set of labeled and unlabeled data $\mathcal{D}$, the regularization parameter ρ, and the sample size of the reparametrization trick L

Ensure: Variational parameters $\{\boldsymbol{\mu}_t\}_{t\in\boldsymbol{t}^{\mathrm{L}}\cup\boldsymbol{t}^{\mathrm{U}}}$, $\{\boldsymbol{\Lambda}_t\}_{t\in\boldsymbol{t}^{\mathrm{L}}}$, $\{\boldsymbol{\sigma}_t\}_{t\in\boldsymbol{t}^{\mathrm{U}}}$ and hyperparameters $\boldsymbol{\theta}$

```
Initialize {μ_t}_{t∈t^L ∪ t^U}, {Λ_t}_{t∈t^L}, {σ_t}_{t∈t^U}, and θ
repeat
  for t = t_1 to t_L do
    for d = 1 to D do
      update μ_td and λ_td by using (15)
    end for
    for n = 1 to N_t do
      update ξ_n^t by using (15)
    end for
  end for
  update {μ_t}_{t∈t^U} and {σ_t}_{t∈t^U} so as to minimize (16) using (17)
  update θ so as to minimize (18) using (19) and (20)
until end condition is true
return {μ_t}_{t∈t^L ∪ t^U}, {Λ_t}_{t∈t^L}, {σ_t}_{t∈t^U}, and θ
```

posterior probability of label $y_n^{t_*}=1$ given a sample $\boldsymbol{x}_n^{t_*}$ is given as,

$$p(y_n^{t_*}=1|\boldsymbol{x}_n^{t_*})=\sigma(\tau(\tilde{\sigma}^2)\tilde{\mu}),$$
$$\tilde{\mu}=\boldsymbol{m}_{t_*}^\top\boldsymbol{x}_n^{t_*},\ \tilde{\sigma}^2=\boldsymbol{x}_n^{t_*\top}\Sigma_{t_*}\boldsymbol{x}_n^{t_*},\ \tau(z)=(1+\pi z/8)^{-\frac{1}{2}}, \tag{22}$$

where $\boldsymbol{m}_{t_*}:=(m_{t_*1},\ldots,m_{t_*D})$, and Σ_{t_*} is a diagonal matrix whose diagonal elements are $(\sigma^2_{t_*1},\ldots,\sigma^2_{t_*D})$.

4 EXPERIMENTS

We conducted experiments using two synthetic and four real-world data sets to confirm the effectiveness of the proposed method. In our experiments, we discretized time at regular intervals, and data collected in the same time unit were regarded as data at the same time as that used in previous studies [26, 28].

4.1 Synthetic Data

We used two synthetic data sets. For the first data set, called Synth1, 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 $p(\boldsymbol{x}|y,t)=0.5p_1(\boldsymbol{x}|y,t)+0.5p_2(\boldsymbol{x}|y,t)$,

$$\begin{cases}x_1=7\cdot\cos(\pi((t-1)/19+1-y))+\epsilon_1,\\ x_2=7\cdot\sin(\pi((t-1)/19+1-y))+\epsilon_2 \quad \text{in } p_1(\boldsymbol{x}|y,t),\end{cases}$$
$$\begin{cases}x_1=7\cdot\cos(\pi((t-1)/19+1.5-y))+\epsilon_1,\\ x_2=7\cdot\sin(\pi((t-1)/19+1.5-y))+\epsilon_2 \quad \text{in } p_2(\boldsymbol{x}|y,t),\end{cases} \tag{23}$$

where ϵ_i for $i=1,2$ is a random variable with a standard Gaussian distribution. The decision boundary turns around the origin at a constant (linear) angular velocity in this data set. We changed time t from 1 to 20 and generated 200 samples for each label y and each time t.

For the second data set, called Synth2, 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 $p(\boldsymbol{x}|y,t)$,

$$x_1=2\cdot\cos\left(\pi\left((1.1)^t/5+1-y\right)\right)+\epsilon_1,$$
$$x_2=2\cdot\sin\left(\pi\left((1.1)^t/5+1-y\right)\right)+\epsilon_2, \tag{24}$$

where ϵ_i for $i=1,2$ is a random variable with a standard Gaussian distribution. The decision boundary turns around the origin at gradually increasing (non-linear) angular velocity in this data set. We changed time t from 1 to 20 and generated 100 samples for each label y and each time t.

4.2 Real-world Data

We used four real-world data sets: SPAM2[1], ELEC2[2], ONP[3], and BLOG [4]. These data sets were used in a previous study [28]. SPAM2 and ELEC2 are public benchmark data sets for evaluating concept drift. SPAM2 contains 11,905 samples and has 166,047 features, ELEC2 contains 45,312 samples and has eight features, ONP consists of 39,797 samples and 61 features, and BLOG consists of 60,021 samples and 281 features. We conducted the following preprocessing, which is the same as that in a previous study [28]. For SPAM2, we added non-spam in SPAM1 (Concept drift Dataset1) to SPAM2 (Concept drift Dataset2) since the number of spam and non-spam were very skewed. In addition, we reduced the features to 200-dimensional features by principal component analysis (PCA), preserving 99.9% of the accumulated proportion. For ELEC2, samples that contain missing values were removed since missing values were not the focus of our task. As a result, SPAM2 contained 14,623 samples and ELEC2 contained 27,549 samples. For ONP, we transformed the regression task into a binary classification as the author instructs. For BLOG, we also converted the regression task into a binary classification based on whether the target value was more than zero, and used samples in which there are time stamp information. As a result, BLOG contained 7,624 samples. In addition, we rescaled the range of features to scale the range $[0,1]$ for SPAM2 and BLOG, and normalized each feature vector by ℓ_2-normalization for ELEC2 and ONP.

4.3 Setting

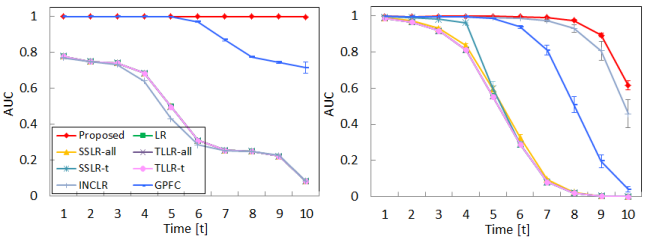
To find the temporal variation in classification performance in our experiments, we evaluated the area under the curve (AUC), which is a well-used evaluation measure for classification tasks, by using data until a certain time unit for labeled training data, and the remaining for test data, which are also used as unlabeled training data. This setting is the same as that in previous studies [26, 28]. In our experiments, test data were the same as unlabeled training data. Although the one-step ahead evaluation, with which classification performance is evaluated in predicting one step ahead, is widely used in concept drift studies [11], we did not use this evaluation procedure because our task was focused on the long-time prediction without additional labeled data. For Synth1 and Synth2, we set $t_L=10$ and the remaining ten time units as test time units. We created ten different data sets for each synthetic data set. For SPAM2, we set two weeks as one time unit, $t_L=14$, and the remaining ten

[1] http://www.comp.dit.ie/sjdelany/Dataset.htm
[2] http://www.inescporto.pt/~jgama/ales/ales_5.html
[3] https://archive.ics.uci.edu/ml/data sets/Online+News+Popularity
[4] https://archive.ics.uci.edu/ml/data sets/BlogFeedback

time units as test time units. For ELEC2, we set two weeks as one time unit, $t_L = 29$, and the remaining ten time units as test time units. For ONP, we set one month as one time unit, $t_L = 15$, and the remaining ten time units as test time units. For BLOG, we set two days as one time unit, $t_L = 20$, and the remaining ten time units as test time units. We evaluated classification performance by varying the amount of labeled training data N for the real-world data sets. We fixed the amount of unlabeled data (test data) M, which was 80% of the original samples at test time units. For labeled training data, we chose the same number of samples from each time unit. Specifically, when $N/M = 0.2$, the number of labeled samples at each time unit for SPAM2, ELEC2,ONP, and BLOG were 100,35,240, and 20, respectively. We created ten different data sets by randomly choosing samples at every time unit according to the above procedure and evaluated the average AUC by using these data sets.

4.4 Comparison Methods

We compared the proposed method with five other methods: logistic regression (LR), semi-supervised logistic regression (SSLR), transfer learning logistic regression (TLLR), incremental logistic regression (INCLR), and Gaussian processes future classifiers (GPFC). LR learns a classifier by using all labeled data $\mathcal{D}^{\mathrm{L}}$ without time stamp information. SSLR is a semi-supervised learning extension of logistic regression [16]. SSLR learns classifiers with the minimum entropy regularization, which is also used in the proposed method. In our experiments, we evaluated two types of SSLR: SSLR-t and SSLR-all. SSLR-t learns a classifier with labeled data $\mathcal{D}^{\mathrm{L}}$ and unlabeled data at time t, $\mathcal{D}^{\mathrm{U}}_t$ when classifying samples at time t. SSLR-all learns a classifier with labeled data $\mathcal{D}^{\mathrm{L}}$ and all unlabeled data $\mathcal{D}^{\mathrm{U}}$ when classifying samples at time t. We included SSLR-all in our evaluation because the proposed method also uses all unlabeled data $\mathcal{D}^{\mathrm{U}}$ for learning the dynamics of the decision boundary. TLLR is a transfer learning extension of logistic regression, which is one of the most representative approaches in transfer learning as stated in Section 2. TLLR can learn unbiased classifiers in a situation called *covariate shift*, where training and test distributions of the feature vectors differ but the class label distribution given the feature vectors is constant. Specifically, TTLR learns classifiers with the the unconstrained least-square importance fitting approach (uLSIF), which is one of the importance weighting method [20]. Since the uLSIF has shown to have excellent numerical stability and efficient runtime solution, we chose this as the comparison method. Note that although a method taking into account the both the covariate shift and emergence of new features has been proposed recently [27], this method is equivalent to TLLR in our experiments since the number of features in the data sets is constant. As with SSLR, we evaluated two types of TLLR: TLLR-t and TLLR-all. INCLR, which is introduced by us in this paper, updates classifiers sequentially given unlabeled data, which is a representative of recently proposed methods [6, 9, 18, 40, 43]. Specifically, INCLR updates a classifier $\boldsymbol{w}_t$ for unlabeled data $\mathcal{D}^{\mathrm{U}}_t$ by minimizing the following objective function, $\|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}\|^2 + \frac{\rho}{M_t}\sum_{m=1}^{M_t} H(p(y|\boldsymbol{x}^t_m, \boldsymbol{w}_t))$,where $\|\cdot\|$ denotes the Frobenius norm, ρ is the positive regularization parameter, and $H(\cdot)$ is the entropy function of class-conditional distribution defined in Section 3. An initial classifier $\boldsymbol{w}_{t_L}$ is learned by logistic regression



(a) Synth1 (b) Synth2

Figure 2: Temporal variation in average and standard errors of AUCs for all test time units with synthetic data sets

with all labeled data $\mathcal{D}^{\mathrm{L}}$. GPFC learns the dynamics of the decision boundary with only labeled data $\mathcal{D}^{\mathrm{L}}$ and predicts future classifiers by using the learned dynamics [28]. Although LR and GPFC do not use unlabeled data for learning, SSLR, TLLR, INCLR, and the proposed method use them for learning. To evaluate the effectiveness of taking into account the dynamics of the decision boundary with labeled and unlabeled data, we used logistic regression for classifiers with all methods including the proposed method.

To select the hyperparameter values, we randomly chose labeled samples from the most recent time t_L without overlapping the labeled training data as validation data. We considered the following variations: the regularization parameter for logistic regression from $\{10^{-2}, 10^{-1}, 1, 10^1, 10^2\}$ in LR, SSLR-all, SSLR-t, TLLR-all, TLLR-t, and INCLR, the regularization parameter for incorporating unlabeled data $\rho \in \{0.1, 1, 10, \{0.5 \cdot 10^n, 10^n\}_{n=2}^3\}$ in SSLR-all, SSLR-t, INCLR, and proposed method, the regularization parameter for importance from $\{0.1, 1, 10, \{0.5 \cdot 10^n, 10^n\}_{n=2}^3\}$ in TLLR-all and TLLR-t. The band width of the Gaussian RBF kernel for estimating the importances is determined by median trick, that is, it is set by the median of squared distance between labeled and unlabeled samples in TLLR-all and TLLR-t. The number of samples for the reparametrization trick J is set as one for all data sets in the proposed method.

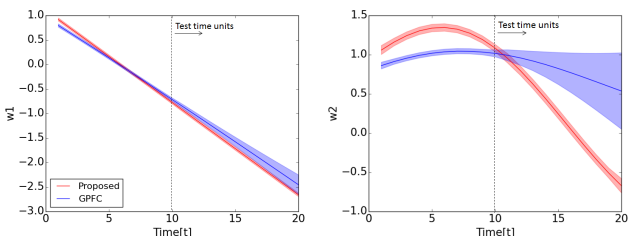
5 RESULTS

Table 1 shows the average and standard errors of AUCs over all test time units for all data sets. For all data sets, the proposed method achieved the highest AUC with all N/M.

Figure 2 shows the temporal variation in the average and standard errors of AUCs in the synthetic data sets. For Synth1, the proposed method could maintain the AUC to almost one for all test time units although the AUCs of other methods quickly become poor. Though GPFC showed a high AUC for a short time, it did not maintain it for a long because the long-time prediction of the future decision boundary was difficult. In contrast, the proposed method could maintain a high AUC for a long time since unlabeled data helped it to learn the dynamics of the decision boundary correctly. For Synth2, the proposed method and INCLR showed better AUC over time compared with the other methods. Since INCLR updates classifiers using newly obtained unlabeled data, it relatively succeeded in tracking the time-evolving decision boundary. Nevertheless, the proposed method could better maintain a high

Table 1: Average and standard errors of AUCs over all test time units with N labeled samples and M fixed unlabeled (test) samples. Values in boldface are statistically better than others (in paired t-test, p=0.05)

Data	N/M	Proposed	LR	SSLR-all	TLLR-all	SSLR-t	TLLR-t	INCLR	GPFC
Synth1	100%	**0.999** (0.001)	0.458 (0.025)	0.458 (0.025)	0.458 (0.025)	0.460 (0.025)	0. 458 (0.025)	0.442 (0.025)	0.908 (0.012)
Synth2	100%	**0.946** (0.012)	0.463 (0.041)	0.478 (0.041)	0.464 (0.041)	0.493 (0.044)	0.464 (0.041)	0.913 (0.019)	0.746 (0.035)
SPAM2	20%	**0.928** (0.006)	0.897 (0.008)	0.907 (0.007)	0.894 (0.008)	0.906 (0.007)	0.896 (0.008)	0.901 (0.008)	0.919 (0.007)
	40%	**0.963** (0.003)	0.925 (0.006)	0.927 (0.006)	0.904 (0.007)	0.925 (0.006)	0.907 (0.007)	0.916 (0.007)	0.957 (0.007)
ELEC2	20%	**0.908** (0.005)	0.906 (0.005)	0.905 (0.005)	0.903 (0.005)	0.898 (0.006)	0.901 (0.005)	0.899 (0.006)	**0.908** (0.005)
	40%	**0.910** (0.005)	0.906 (0.005)	0.906 (0.006)	0.904 (0.005)	0.904 (0.005)	0.901 (0.005)	0.887 (0.007)	**0.909** (0.005)
ONP	20%	**0.568** (0.006)	0.547 (0.005)	0.547 (0.005)	**0.564** (0.003)	0.547 (0.005)	**0.563** (0.003)	0.546 (0.005)	**0.569** (0.004)
	40%	**0.584** (0.003)	0.535 (0.004)	0.535 (0.004)	0.564 (0.002)	0.535 (0.004)	0.560 (0.002)	0.534 (0.004)	**0.584** (0.003)
BLOG	20%	**0.787** (0.005)	0.761 (0.004)	0.761 (0.004)	0.763 (0.004)	0.761 (0.004)	0.763 (0.004)	0.735 (0.006)	0.771 (0.006)
	40%	**0.797** (0.004)	0.757 (0.004)	0.730 (0.005)	0.730 (0.005)	0.737 (0.005)	0.730 (0.005)	0.729 (0.005)	0.786 (0.005)



(a) Classifier parameters for the 1st feature **(b) Classifier parameters for the 2nd feature**

Figure 3: Classifier parameters estimated with proposed method and GPFC on Synth1. Solid line is mean of probability of predicted classifiers for each time unit, and shaded region corresponds to plus and minus two times standard deviations. Classifier parameters estimated with proposed method achieves that AUC is one for all time units

AUC compared with INCLR. Figure 3 shows an example of classifier parameters estimated with the proposed method and GPFC on Synth1. Although GPFC showed high variances $\sigma^2_{t_* d}$ in (21) for the time units ($t > 10$) since GPFC do not use any data collected at the times for learning, the proposed method learned correct classifier parameters, whose AUC was one, with low variances. Overall, we found that the proposed method could better maintain classification performance compared with the other methods.

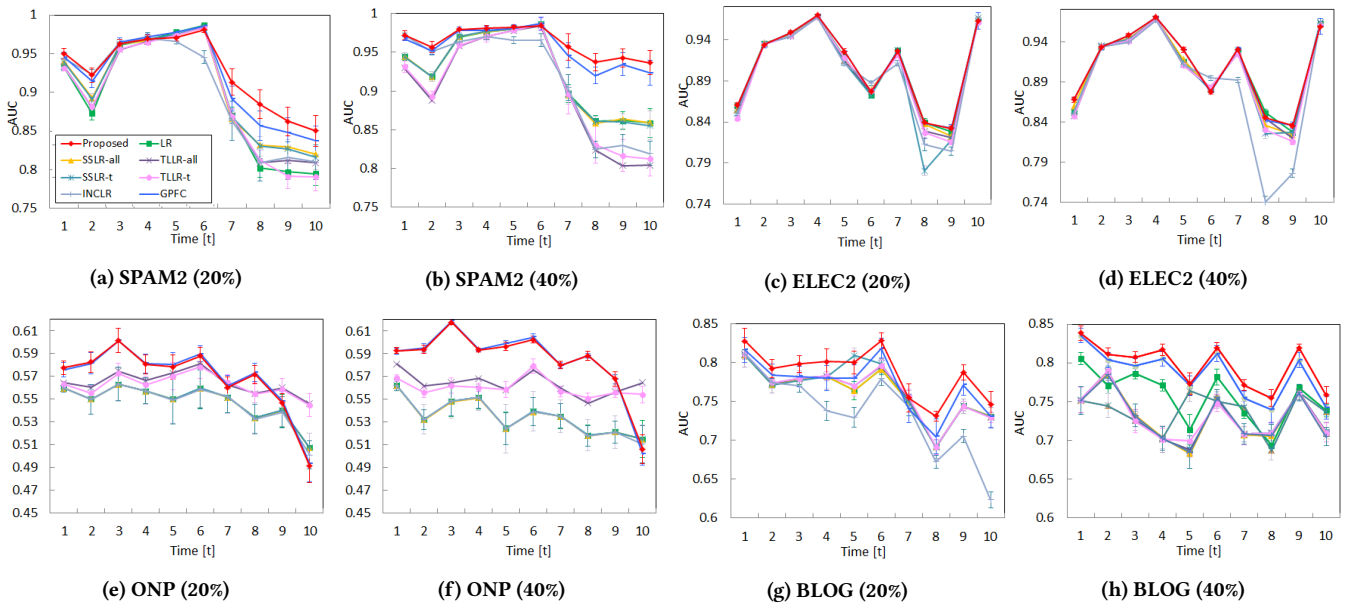
Figure 4 shows the temporal variation in the average and standard errors of AUCs in the real-world data sets. For SPAM2 and BLOG, the proposed method outperformed the others over almost all test time units. The proposed method showed better classification performance compared with GPFC. This means that the proposed method can learn the more correct dynamics of the decision boundary by using unlabeled data as well as labeled data. For ELEC2 and ONP, the proposed method and GPFC achieved the highest AUC over almost all test time units. When $t = 10$ in ONP, the proposed method and GPFC deteriorated in performance. One of the reasons for this deterioration is that the nature of data at $t = 10$ was quite different from those of labeled training data; therefore, predicting the right decision boundary was also difficult. Nevertheless, the proposed method and GPFC were effective since they showed good performance at almost all test time units. For all real-world data sets except ONP, SSLR-all, TLLR-all, SSLR-t, TLLR-t, and INCLR did not significantly improve in performance compared with LR, though they also used unlabeled data for learning. Especially, INCLR greatly deteriorated in performance for ELEC2 and BLOG. This suggests that unlabeled data have a risk of deteriorating performance, even though they also have useful information. The proposed method succeeded in exploiting useful information on the decision boundaries from unlabeled data by modeling the dynamics of the decision boundary. As a result, the proposed method improved in classification performance.

Overall, through the experiments, we confirmed two important observations; The first is that modeling the dynamics of the decision boundary is quite effective for maintaining classification performance. This is because the proposed method and GPFC were almost consistently superior to the other methods. The second is that using unlabeled data to learn the dynamics of the decision boundary is also quite effective for maintaining performance, as also shown in our experimental results.

Finally, we investigated how the classification performance of the proposed method changed against the regularization parameter ρ, which controls the effect of the entropy posterior regularization. Figure 5 represents the average of AUCs for all test time units when varying ρ. For SPAM2 and ELEC2, the proposed method improved the AUC compared with LR for all ρ values. For ONP and BLOG, the proposed method improved the AUC in the range where the ρ value was not too large. Although the effectiveness of unlabeled data varied depending on the data set, the proposed method was able to select good ρ by using validation data.

6 CONCLUSION

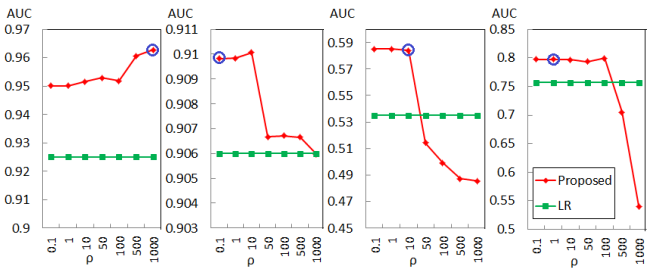
We proposed a method for learning the dynamics of the decision boundary to maintain classification performance by using newly obtained unlabeled data as well as labeled data collected beforehand. With the proposed method, the dynamics of the decision boundary is modeled by Gaussian processes. To incorporate unlabeled data into our probabilistic model, the proposed method introduces the entropy posterior regularization on the basis of the generic RegBayes framework. In addition, we presented an inference algorithm based on variational Bayesian inference. Through experiments, we



(a) SPAM2 (20%) (b) SPAM2 (40%) (c) ELEC2 (20%) (d) ELEC2 (40%)

(e) ONP (20%) (f) ONP (40%) (g) BLOG (20%) (h) BLOG (40%)

Figure 4: Temporal variation in average and standard errors of AUCs for all test time units with real-world data sets



(a) SPAM2 (40%) (b) ELEC2 (40%) (c) ONP (40%) (d) BLOG (40%)

Figure 5: Average of AUCs for all test time units when varying regularization parameter ρ. Blue circle in each graph denotes AUC chosen using validation data

showed that the proposed method can better maintain the classification performance over time compared with seven other methods. For future work, we will apply other regularizations, such as the manifold regularization, to our framework.

REFERENCES

[1] Zahraa Said Abdallah, Mohamed Medhat Gaber, Bama Srinivasan, and Shonali Krishnaswamy. 2012. StreamAR: incremental and active learning with evolving sensory data for activity recognition. In *IEEE 24th International Conference on Tools with Artificial Intelligence*, Vol. 1. IEEE, 1163–1170.

[2] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. 2002. Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. ACM, 1–16.

[3] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *JMLR* 7, Nov (2006), 2399–2434.

[4] Daniel Billsus and Michael J Pazzani. 2000. User modeling for adaptive news access. *User Modeling and User-adapted Interaction* 10, 2-3 (2000), 147–180.

[5] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

[6] Adeleh Bitarafan, Mahdieh Soleymani Baghshah, and Marzieh Gheisari. 2016. Incremental evolving domain adaptation. *TKDE* 28, 8 (2016), 2128–2141.

[7] Koby Crammer, Alex Kulesza, and Mark Dredze. 2009. Adaptive regularization of weight vectors. In *NIPS*.

[8] Thang D. Bui Richard E. Turner Cuong V. Nguyen, Yingzhen Li. 2018. Variational continual learning. In *ICLR*.

[9] Karl B Dyer, Robert Capo, and Robi Polikar. 2014. Compose: a semisupervised learning framework for initially labeled nonstationary streaming data. *Neural Networks and Learning Systems, IEEE Transactions on* 25, 1 (2014), 12–26.

[10] Yoav Freund, Robert E Schapire, et al. 1996. Experiments with a new boosting algorithm. In *ICML*.

[11] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 44.

[12] Jin Gao, Haibin Ling, Weiming Hu, and Junliang Xing. 2014. Transfer learning based visual tracking with gaussian processes regression. In *ECCV*.

[13] Andrew B Goldberg, Ming Li, and Xiaojin Zhu. 2008. Online manifold regularization: a new learning setting and empirical study. In *ECML PKDD*.

[14] Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. 2017. Adaptive random forests for evolving data stream classification. *Machine Learning* (2017), 1–27.

[15] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.

[16] Yves Grandvalet and Yoshua Bengio. 2004. Semi-supervised learning by entropy minimization. In *NIPS*.

[17] Ahsanul Haque, Latifur Khan, and Micheal Baron. 2016. SAND: semi-supervised adaptive novel class detection and classification over data stream. In *AAAI*.

[18] Judy Hoffman, Trevor Darrell, and Kate Saenko. 2014. Continuous manifold based adaptation for evolving visual domains. In *CVPR*.

[19] Rudolph Emil Kalman et al. 1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering* 82, 1 (1960), 35–45.

[20] Takafumi Kanamori, Shohei Hido, and Masahi Sugiyama. 2009. Efficient direct density ratio estimation for non-stationarity adaptation and outlier detection. In *NIPS*.

[21] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *NIPS*.

[22] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *ICLR* (2014).

[23] Ralf Klinkenberg. 2004. Learning drifting concepts: example selection vs. example weighting. *Intelligent Data Analysis* 8, 3 (2004), 281–300.

[24] J Zico Kolter and Marcus A Maloof. 2007. Dynamic weighted majority: An ensemble method for drifting concepts. *JMLR* 8 (2007), 2755–2790.

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

[26] Atsutoshi Kumagai and Tomoharu Iwata. 2016. Learning future classifiers without additional data. In *AAAI*.

[27] Atsutoshi Kumagai and Tomoharu Iwata. 2017. Learning latest classifiers without additional labeled data. In *IJCAI*.

[28] Atsutoshi Kumagai and Tomoharu Iwata. 2017. Learning non-linear dynamics of decision boundaries for maintaining classification performance. In *AAAI*.

[29] Peipei Li, Xindong Wu, and Xuegang Hu. 2010. Mining recurring concept drifts with limited labeled streaming data. In *ACML*.

[30] Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming* 45, 1-3 (1989), 503–528.

[31] Mingsheng Long, Jianmin Wang, and Michael I Jordan. 2017. Deep transfer learning with joint adaptation networks. In *ICML*.

[32] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2016. Unsupervised domain adaptation with residual transfer networks. In *NIPS*.

[33] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. 2009. Identifying suspicious urls: an application of large-scale online learning. In *ICML*.

[34] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2016. Distributional smoothing with virtual adversarial training. In *ICLR*.

[35] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *TNN* 22, 2 (2011), 199–210.

[36] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on* 22, 10 (2010), 1345–1359.

[37] Carl Rasmussen and Chris Williams. 2006. *Gaussian process for machine learning*. MITPress.

[38] Paul Ruvolo and Eric Eaton. 2013. ELLA: an efficient lifelong learning algorithm. In *ICML*.

[39] Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *JSPI* 90, 2 (2000), 227–244.

[40] Vinícius MA Souza, Diego F Silva, João Gama, and Gustavo EAPA Batista. 2015. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In *SDM*.

[41] Peter Sykacek and Stephen J Roberts. 2003. Adaptive classification by variational Kalman filtering. In *NIPS*.

[42] Kevin Tang, Vignesh Ramanathan, Li Fei-Fei, and Daphne Koller. 2012. Shifting weights: adapting object detectors from image to video. In *NIPS*.

[43] Muhammad Umer, Robi Polikar, and Christopher Frederickson. 2017. LEVEL IW: Learning extreme verification latency with importance weighting. In *IJCAN*.

[44] Boyu Wang and Joelle Pineau. 2015. Online boosting algorithms for anytime transfer and multitask learning. In *AAAI*.

[45] Haixun Wang, Wei Fan, Philip S Yu, and Jiawei Han. 2003. Mining concept-drifting data streams using ensemble classifiers. In *SIGKDD*.

[46] Jialei Wang, Peilin Zhao, and Steven C.Hoi. 2012. Exact soft confidence-weighted learning. In *ICML*.

[47] Ji Won Yoon, Stephen J Roberts, Matt Dyson, and John Q Gan. 2009. Adaptive classification for brain computer interface systems using sequential monte carlo sampling. *Neural Networks* 22, 9 (2009), 1286–1294.

[48] Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *ICML*.

[49] Peng Zhang, Xingquan Zhu, and Li Guo. 2009. Mining data streams with labeled and unlabeled training examples. In *ICDM*. IEEE, 627–636.

[50] Peilin Zhao and Steven C Hoi. 2010. OTL: a framework of online transfer learning. In *ICML*.

[51] Jun Zhu, Ning Chen, and Eric P Xing. 2014. Bayesian inference with posterior regularization and applications to infinite latent svms. *JMLR* 15 (2014), 17–99.

[52] Indre Zliobaite, Albert Bifet, Bernhard Pfahringer, and Graham Holms. 2014. Active learning with drifting streaming data. *Neural Networks and Learning Systems, IEEE Transactions on* 25, 1 (2014), 27–39.