Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Learning Non-Linear Dynamics of Decision Boundaries for Maintaining Classification Performance

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 a method that involves a probabilistic model for learning future classifiers for tasks in which decision boundaries nonlinearly change over time. In certain applications, such as spam-mail classification, the decision boundary dynamically changes over time. Accordingly, the performance of the classifiers will deteriorate quickly unless the classifiers are updated using additional data. However, collecting such data can be expensive or impossible. The proposed model alleviates this deterioration in performance without additional data by modeling the non-linear dynamics of the decision boundary using Gaussian processes. The method also involves our developed learning algorithm for our model based on empirical variational Bayesian inference by which uncertainty of dynamics can be incorporated for future classification. The effectiveness of the proposed method was demonstrated through experiments using synthetic and real-world data sets.

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 malicious can vary over time (Ma et al. 2009). In activity recognition using sensor data, the decision boundary can vary over time since user activity patterns dynamically change (Abdallah et al. 2012). When we do not update classifiers for tasks in which the decision boundary evolves over time, the classification performance deteriorates quickly (Gama et al. 2014).

There have been many methods proposed for updating classifiers to maintain performance, such as online learning (Rosenblatt 1958; Crammer, Kulesza, and Dredze 2009; Crammer et al. 2010; Wang, Zhao, and C.Hoi 2012), forgetting algorithms (Klinkenberg 2004), ensemble learning (Wang et al. 2003; Kolter and Maloof 2007; Bach and Maloof 2010; Brzezinski and Stefanowski 2014), and domain adaptation (Hoffman, Darrell, and Saenko 2014). These methods require additional labeled and/or unlabeled data to update classifiers. However, using these data can be expensive or impossible. For example, collecting labeled data is quite expensive since labels need to be manually assigned by domain experts. Even using unlabeled data is impossible in some domains such as security. For example, anti-virus software vendors distribute their update files (data for updating classifiers) many times a day for immediately dealing with rapidly evolving malware. When software is used in an off-line environment, which has a risk of virus infection via physical media such as USB, it cannot receive the update files; therefore, it cannot update the classifiers.

To overcome this problem, a method for learning future decision boundaries given only labeled data with timestamps collected until the current time has been proposed (Kumagai and Iwata 2016). This method learns the dynamics of the decision boundary without additional data and predicts the future decision boundary. The dynamics of the decision boundary is modeled using vector autoregressive models, with which the decision boundary is assumed to depend linearly on the previous decision boundaries. However, this linearity assumption is too strong because decision boundaries would change more intricately in real-world data.

In this paper, we propose a method that involves a novel probabilistic model for learning future classifiers, whose dynamics is non-linear, when only labeled data with timestamps collected until the current time are available. With the proposed model, a decision boundary is defined by the classifier parameters, and the dynamics of each parameter is modeled by Gaussian processes (GPs) (Rasmussen and Williams 2006). By handling the dynamics of each parameter, our model can reflect the characteristics of each feature. For example, in spam-mail classification, words such as 'free', 'viagra', and 'DISCOUNT', have been used in spam mail for a long time; therefore, the parameters for such words (features) would be effective over time. In contrast, for words temporarily used in spam mail, such as new product names and celebrity's names, the parameters would become ineffective over time. The proposed method also involves our developed learning algorithm based on empirical variational Bayesian inference for optimizing classifier parameters and hyperparameters for GPs simultaneously. By using a Bayesian framework, the proposed method can classify data collected at a future time by taking account into the uncertainty of predicted future classifiers, which enables robust classification.

2 Related Work

To maintain the performance of classifiers, updating the classifiers with labeled data has been widely used. Online learning allows classifiers to be updated with sequentially arriving data (Rosenblatt 1958; Loosli, Canu, and Bottou 2007; Crammer, Kulesza, and Dredze 2009; Crammer et al. 2010; Wang, Zhao, and C.Hoi 2012). Forgetting algorithms learn the latest decision boundary by weighting training data with their age, which means recent data are important for learning (Koychev 2000; Klinkenberg 2004; Babcock et al. 2002). Ensemble learning combines multiple base classifiers to create a better classifier (Wang et al. 2003; Kolter and Maloof 2007; Bach and Maloof 2010; Brzezinski and Stefanowski 2014). Its mixture ratio is estimated so that base classifiers that effectively capture a recent decision boundary have a large ratio. Methods for learning the recent decision boundary by using unlabeled data have also been proposed (Zhang, Zhu, and Guo 2009; Dyer, Capo, and Polikar 2014; Hoffman, Darrell, and Saenko 2014; Haque, Khan, and Baron 2016). Some methods involve active learning, which is a method for determining what data should be labeled, to learn the time-evolving decision boundary (Zhu et al. 2010; Zliobaite et al. 2014). (Royer and Lampert 2015) proposed a method for classifying non-i.i.d test samples drawn form a time-evolving data distribution. (Zhou, Sohn, and Lee 2012) proposed an incremental feature learning algorithm for a time-varying data distribution. (Gao et al. 2014) used GPs for latent variables of samples in order to cope with concept drift. All these methods continuously require additional labeled and/or unlabeled data for adapting the time-varying decision boundary. However, the task we investigated in this study is learning future classifiers without additional labeled and/or unlabeled data.

There have been methods proposed that predict future states by using data collected until the current time. A method for predicting future probability distributions has been proposed (Lampert 2015). Although this method predicts the future state of a probability distribution, the proposed method predicts the future decision boundary. A method for learning future decision boundaries has also been proposed (Kumagai and Iwata 2016). Although it is assumed that the decision boundary depends linearly on the previous decision boundaries with this method, the proposed method does not rely on the such assumption due to GP regression, which is a non-linear non-parametric regression model. Though the previous method cannot take into account the uncertainty of dynamics, by using a Bayesian framework, the proposed method can. Moreover, since autoregressive models require decision boundaries at regular intervals without absence to predict future decision boundaries, the previous method requires training data to be collected continuously over a certain period. In contrast, the proposed method does not require this. These methods including the propose method use time-series models for predicting future states. Although time-series models usually require observed samples in the form of time-series, these methods only require independent samples from different time units.

The proposed method is related to transfer learning (Pan and Yang 2010). Transfer learning uses data in a source domain to solve a related problem in the target domain. In our task, the proposed method regards labeled data collected until a certain time point as data in the source domain, and data collected after the certain point as data in the target domain. Although transfer learning uses labeled or unlabeled data in the target domain, the proposed method does not use any data in the target domain for learning. Learning the dynamics of the decision boundaries is to learn how to transfer the decision boundaries in the source domain to those in the target domain.

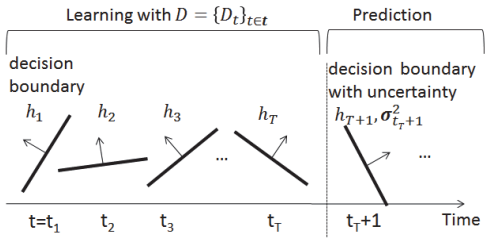


Figure 1: Illustration of our approach

3 Proposed Method

3.1 Model

We introduce the notations used in this paper and define the task we investigated. Let $\mathcal{D}_t{:=}\{(\boldsymbol{x}_n^t, y_n^t)\}_{n=1}^{N_t}$ 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 n-th 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. $\boldsymbol{t} := (t_1, \ldots, t_T)$, where $t_1 < t_2 < \cdots < t_T$, denotes times in which data are collected. Our goal is to find classifiers $h_t : \mathbb{R}^d \to \{0,1\}$, $\forall t > t_T$, which can precisely classify data at time t, given a set of training data $\mathcal{D}{:=}\{\mathcal{D}_t\}_{t\in\boldsymbol{t}}$. Figure 1 illustrates our approach. Using labeled data from time t_1 to t_T, the decision boundary for each time, which is defined by classifier h_t, and the dynamics of the decision boundary is learned. Then, the decision boundary at a future t, $t > t_T$, is predicted with uncertainty $\boldsymbol{\sigma}_t^2$ by using the learned decision boundary and dynamics.

It is assumed with our probabilistic model 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 $\boldsymbol{w}_t = (w_{t1}, \ldots, w_{tD}) \in \mathbb{R}^D$ is a parameter vector of h_t, σ 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 $1 - p(y_n^t = 1|\boldsymbol{x}_n^t, \boldsymbol{w}_t)$.

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

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

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 $\boldsymbol{t} = (t_1, \ldots, t_T)$, the prior on the corresponding outputs $\boldsymbol{f}_d := (f_d(t_1), \ldots, f_d(t_T))$ 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 $\boldsymbol{K}_d \in \mathbb{R}^{T\times T}$ 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 a Gaussian kernel with a bias term as the kernel function,

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

where α_d, β_d, and γ_d are kernel hyperparameters. This kernel function describes the time series with a smooth shape well. By integrating out $\boldsymbol{f}_d$, we obtain the probability of the d-th component of the classifier at the input times $\boldsymbol{t}$, $\boldsymbol{w}_{\cdot d} := (w_{t_1 d}, \ldots, w_{t_T d}) \in \mathbb{R}^T$, 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 $\boldsymbol{C}_d \in \mathbb{R}^{T\times T}$ is defined by the following kernel function,

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

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

The joint distribution of labeled data $\mathcal{D}:=\{\mathcal{D}_t\}_{t\in\boldsymbol{t}}$ and classifier parameters $\boldsymbol{W} := (\boldsymbol{w}_{t_1}, \ldots, \boldsymbol{w}_{t_T})$ is written as

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

where $\boldsymbol{\theta} := (\alpha_1,\ldots, \alpha_D, \beta_1,\ldots, \beta_D, \gamma_1,\ldots, \gamma_D, \eta_1,\ldots, \eta_D)$, and $p(\boldsymbol{W}; \boldsymbol{\theta}) = \prod_{d=1}^{D} p(\boldsymbol{w}_{\cdot d})$. When alpha is infinitely large and gamma is zero, the classifier parameters in different time points are independent. This corresponds to learning classifiers for each time point independently. When alpha is zero, the classifier parameters are constant over time. This corresponds to learning a classifier by using all data and ignoring their time stamps. Our probabilistic model can represent various dynamics of the decision boundary by controlling the values of kernel hyperparameters.

3.2 Learning

We developed a learning algorithm for our probabilistic model based on empirical variational Bayesian inference (Bishop 2006). We consider obtaining an approximate posterior distribution, called the variational posterior $q(\boldsymbol{W})$, of the posterior distribution $p(\boldsymbol{W}|\mathcal{D}; \boldsymbol{\theta})$ since analytically calculating $p(\boldsymbol{W}|\mathcal{D}; \boldsymbol{\theta})$ is impossible. In addition, we estimate hyperparameters $\boldsymbol{\theta}$ from training data $\mathcal{D}$. The $q(\boldsymbol{W})$ and optimal hyperparameters $\boldsymbol{\theta}$ can be obtained by maximizing the following lower bound $L(q; \boldsymbol{\theta})$ with respect to q and $\boldsymbol{\theta}$,

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

We assume that $q(\boldsymbol{W})$ can be factorized as follows,

$$q(\boldsymbol{W}) = \prod_{t=t_1}^{t_T}\prod_{d=1}^{D} q(w_{td}). \tag{9}$$

We would like to maximize the lower bound $L(q; \boldsymbol{\theta})$. However, it is impossible because of the non-conjugacy of logistic regression. To solve this problem, we use the following inequality (Bishop 2006),

$$p(y_n^t|\boldsymbol{x}_n^t, \boldsymbol{w}_t) \geq e^{y_n^t a}\sigma(\xi_n^t)\exp\left(-\frac{a + \xi_n^t}{2} - h(\xi_n^t)(a^2 - {\xi_n^t}^2)\right), \tag{10}$$

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 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 Eq. (10) with $L(q; \boldsymbol{\theta})$, we obtain a new lower bound $L(q; \boldsymbol{\theta}, \boldsymbol{\xi})$. Since $L(q; \boldsymbol{\theta}, \boldsymbol{\xi})$ is the lower bound of $L(q; \boldsymbol{\theta})$, increasing the value of $L(q; \boldsymbol{\theta}, \boldsymbol{\xi})$ leads to an increased value of $L(q; \boldsymbol{\theta})$. By calculating the derivatives of $L(q; \boldsymbol{\theta}, \boldsymbol{\xi})$ with respect to q and ξ_n^t, we find that $q(\boldsymbol{W})$ takes the following form,

$$q(w_{td}) = \mathcal{N}(w_{td}|\mu_{td}, \lambda_{td}^{-1}). \tag{11}$$

The variables μ_{td}, λ_{td}, and ξ_t^n satisfy the following equations,

$$\begin{aligned} \mu_{td} &= \lambda_{td}^{-1}\left(\sum_{n=1}^{N_t}\{(y_n^t - \frac{1}{2})x_{nd}^t - 2h(\xi_n^t)\sum_{l\neq d}\mu_{tl}x_{nl}^t x_{nd}^t\} - \sum_{s\neq t}[\boldsymbol{C}_d^{-1}]_{ts}\mu_{sd}\right), \\ \lambda_{td} &= [\boldsymbol{C}_d^{-1}]_{tt} + 2\sum_{n=1}^{N_t} h(\xi_n^t)(x_{nd}^t)^2, \\ (\xi_n^t)^2 &= \boldsymbol{x}_n^{t\top}(\boldsymbol{\Lambda}_t^{-1} + \boldsymbol{\mu}_t\boldsymbol{\mu}_t^\top)\boldsymbol{x}_n^t, \end{aligned} \tag{12}$$

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 $\boldsymbol{x}$. We maximize $L(q; \boldsymbol{\theta}, \boldsymbol{\xi})$ with respect to $\boldsymbol{\theta}$ by using the quasi-Newton method (Liu and Nocedal 1989). The $L(q; \boldsymbol{\theta}, \boldsymbol{\xi})$ is represented as

$$\begin{aligned} L(q; \boldsymbol{\theta}, \boldsymbol{\xi}) = -\frac{1}{2}\sum_{d=1}^{D}[&\boldsymbol{\mu}_{\cdot d}^\top \boldsymbol{C}_d^{-1}\boldsymbol{\mu}_{\cdot d} + \mathrm{Tr}(\boldsymbol{C}_d^{-1}\boldsymbol{\Lambda}_d^{-1}) \\ &+ \log(\det(\boldsymbol{C}_d))] + const, \end{aligned} \tag{13}$$

where $\boldsymbol{\mu}_{\cdot d} := (\mu_{t_1 d}, \ldots, \mu_{t_T d})$, $\boldsymbol{\Lambda}_d := \mathrm{diag}(\lambda_{t_1 d}, \ldots, \lambda_{t_T d})$, Tr is the trace, det is the determinant, and $const$ is the terms that do not depend on $\boldsymbol{\theta}$. The gradients of $L(q; \boldsymbol{\theta}, \boldsymbol{\xi})$ with respect to θ_d, which represents one of the α_d, β_d, γ_d and η_d, are expressed as

$$\frac{\partial L(q; \boldsymbol{\theta}, \boldsymbol{\xi})}{\partial \theta_d} = \frac{1}{2}\boldsymbol{\mu}_{\cdot d}^\top \boldsymbol{C}_d^{-1}\frac{\partial \boldsymbol{C}_d}{\partial \theta_d}\boldsymbol{C}_d^{-1}\boldsymbol{\mu}_{\cdot d} + \frac{1}{2}\mathrm{Tr}\left(\boldsymbol{C}_d^{-1}\frac{\partial \boldsymbol{C}_d}{\partial \theta_d}(\boldsymbol{C}_d^{-1}\boldsymbol{\Lambda}_d^{-1} - \mathbf{I})\right), \tag{14}$$

where $\mathbf{I}$ is the identity matrix. The gradients of the Kernel $[\boldsymbol{C}_d]_{tt'}$ with respect to α_d, β_d, γ_d and η_d are given by

$$\frac{\partial[\boldsymbol{C}_d]_{tt'}}{\partial\alpha_d} = -\alpha_d|t-t'|^2\beta_d^2\exp\left(-\frac{1}{2}\alpha_d^2|t-t'|^2\right),$$
$$\frac{\partial[\boldsymbol{C}_d]_{tt'}}{\partial\beta_d} = 2\beta_d\exp\left(-\frac{1}{2}\alpha_d^2|t-t'|^2\right),$$
$$\frac{\partial[\boldsymbol{C}_d]_{tt'}}{\partial\gamma_d} = 2\gamma_d,\ \frac{\partial[\boldsymbol{C}_d]_{tt'}}{\partial\eta_d} = 2\delta_{tt'}\eta_d. \tag{15}$$

The $q(\boldsymbol{W})$ is estimated by updating variables μ_{td}, λ_{td}, and ξ_n^t by using Eq. (12) and the hyperparameters $\boldsymbol{\theta}$ are estimated using the quasi-Newton method with Eqs. (13), (14), and (15) in turn until some convergence criteria are satisfied.

3.3 Prediction

We explain how to predict future classifiers by using the learned model. Assume we have a test input t_* and we would like to obtain the corresponding output (future decision boundary) $\boldsymbol{w}_{t_*}$. The probability of $\boldsymbol{w}_{t_*} = (w_{t_*1}, \ldots, w_{t_*D})$ is represented as follows,

$$p(\boldsymbol{w}_{t_*}) = \prod_{d=1}^{D} p(w_{t_*d}),$$
$$p(w_{t_*d}) = \int p(w_{t_*d}|\boldsymbol{w}_{\cdot d})q(\boldsymbol{w}_{\cdot d})d\boldsymbol{w}_{\cdot d} = \mathcal{N}(w_{t_*d}|m_{t_*d}, \sigma^2_{t_*d}),$$
$$m_{t_*d} = \boldsymbol{k}_d^\top \boldsymbol{C}_d^{-1}\boldsymbol{\mu}_{\cdot d},$$
$$\sigma^2_{t_*d} = k_d(t_*,t_*) + \eta_d^2 + \boldsymbol{k}_d^\top\left(\boldsymbol{C}_d^{-1}\boldsymbol{\Lambda}_d^{-1} - \mathbf{I}\right)\boldsymbol{C}_d^{-1}\boldsymbol{k}_d, \tag{16}$$

where $\boldsymbol{k}_d := (k_d(t_*,t_1), \ldots, k_d(t_*,t_T))$. The d-the component of the mean future decision boundary m_{t_*d} is given by a weighted sum of the estimated past decision boundaries $\boldsymbol{\mu}_{\cdot d}$, where the weights depend on the test input t_*. In contrast, the weights of the autoregressive models in (Kumagai and Iwata 2016) are constant. The variance $\sigma^2_{t_*d}$, which also depends on t_*, can be used for measuring the uncertainty of the prediction though the autoregressive models cannot take into account the uncertainty of their prediction. For classifying data at time t_*, we use Bayesian logistic regression, which is a method for classification taking variances of $p(\boldsymbol{w}_{t_*})$ into account. For details of Bayesian logistic regression, refer to (Bishop 2006). The posterior probability of label $y_n^{t_*} = 1$ given a sample $\boldsymbol{x}_n^{t_*}$ is given as follows,

$$p(y_n^{t_*} = 1|\boldsymbol{x}_n^{t_*}) = \sigma(\tau(\tilde{\sigma}^2)\tilde{\mu}),$$
$$\tilde{\mu} = \boldsymbol{m}_{t_*}^\top \boldsymbol{x}_n^{t_*},\ \tilde{\sigma}^2 = {\boldsymbol{x}_n^{t_*}}^\top \boldsymbol{\Sigma}_{t_*}\boldsymbol{x}_n^{t_*},$$
$$\tau(z) = (1+\pi z/8)^{-\frac{1}{2}}, \tag{17}$$

where $\boldsymbol{m}_{t_*} := (m_{t_*1}, \ldots, m_{t_*D})$, and $\boldsymbol{\Sigma}_{t_*}$ 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.

4.1 Synthetic Data

We used two synthetic data sets. For the first data set, called Linear, 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 $p(\boldsymbol{x}|y,t)$,

$$x_1 = 2.0\cdot\cos(\pi((t-1)/19 + 1 - y)) + \epsilon_1,$$
$$x_2 = 2.0\cdot\sin(\pi((t-1)/19 + 1 - y)) + \epsilon_2, \tag{18}$$

where ϵ_i for $i = 1, 2$ is a random variable with a standard Gaussian distribution. The decision boundary changes linearly in this data set. We changed t from 1 to 20 and generated 100 samples for each label y and each time t. For the second data set, called Non-linear, when $t \leq 11$, a sample $(x_1, x_2) \in \mathbb{R}^2$ with label $y \in \{0,1\}$ was generated from the following time-invariant distribution,

$$x_1 = 2.0\cdot\cos(\pi(1-y)) + \epsilon_1,$$
$$x_2 = 2.0\cdot\sin(\pi(1-y)) + \epsilon_2. \tag{19}$$

When $t \geq 12$, a sample (x_1, x_2) was generated from the distribution obtained by replacing t of (18) with $t - 10$. The decision boundary changes non-linearly in this data set. We changed t from 1 to 30 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]. 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. BLOG consists of $60,021$ samples and 281 features. For SPAM2, we added ham in SPAM1 (Concept drift Dataset1) to SPAM2 (Concept drift Dataset2) since the number of spams and hams 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 have 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 is more than zero.

4.3 Setting

To find the temporal variation of the classification performance in our experiments, we evaluated Area Under the Curve (AUC) by using data until a certain time unit for training, and the remaining for testing. This setting is the same as that in a previous study (Kumagai and Iwata 2016). For

[1] http://www.comp.dit.ie/sjdelany/Dataset.htm

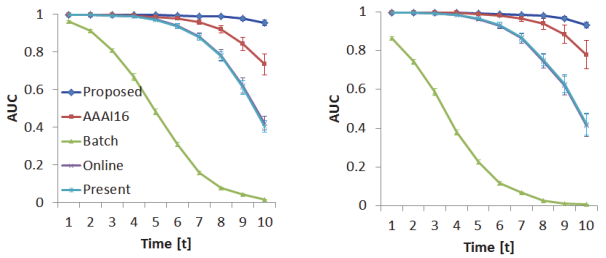
[2] http://www.inescporto.pt/~jgama/ales/ales_5.html

[3] https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity

[4] https://archive.ics.uci.edu/ml/datasets/BlogFeedback

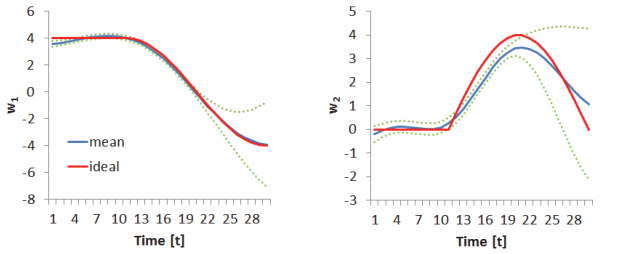
Table 1: Average and standard errors of AUCs over all test time units. Values in boldface are statistically better than others (in paired t-test, p=0.05)

	Proposed	AAAI16	Batch	Online	Present
Linear	**0.988±0.002**	0.940±0.011	0.443±0.036	0.860±0.020	0.860±0.020
Non-linear	**0.984±0.003**	0.952±0.012	0.302±0.031	0.851±0.021	0.854±0.020
SPAM2	**0.956±0.003**	0.907±0.006	0.930±0.005	0.928±0.004	0.915±0.005
ELEC2	0.921±0.005	**0.926±0.005**	0.916±0.004	0.904±0.005	0.903±0.005
ONP	**0.705±0.002**	0.533±0.004	0.630±0.002	0.622±0.002	0.622±0.002
BLOG	**0.838±0.003**	0.827±0.003	0.827±0.004	0.730±0.004	0.754±0.004



(a) Linear (b) Non-linear

Figure 2: Average and standard errors of AUCs for all test time units over ten runs



(a) classifier parameter for the 1st feature (b) classifier parameter for the 2nd feature

Figure 3: Classifier parameters estimated with the proposed method on Non-linear. Blue line is mean of probability of classifier (16) for each input value, and dotted green lines represent point-wise mean plus and minus two times standard deviation for each input value. Red line represents one ideal classifier, whose AUC is one for all time units

Linear, we set $t_T = 10$ and the remaining ten time units as test data. For Non-linear, we set $t_T = 20$ and the remaining ten time units as test data. We created ten different data sets for each synthetic data set. For SPAM2, we set two weeks as one time unit, $t_T = 14$, and the reminding ten time units as test data. For ELEC2, we set two weeks as one time unit, $t_T = 29$, and the reminding ten time units as test data. For ONP, we set one month as one time unit, $t_T = 15$, and the reminding ten time units as test data. For BLOG, we set two days as one time unit, $t_T = 20$, and the reminding ten time units as test data. We chose 80% of samples randomly at every training time unit to create ten different training data, and evaluated the average AUC by using these training data sets.

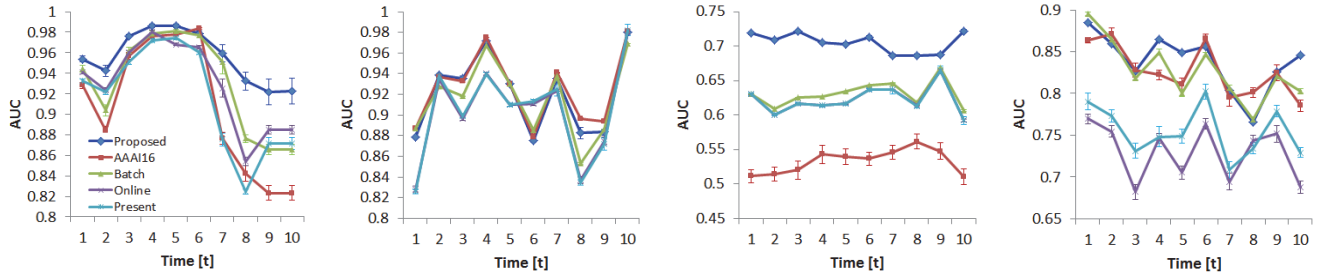
4.4 Comparison methods

We compared the proposed method with four other methods: batch logistic regression (Batch), online logistic regression (Online), present logistic regression (Present) and a method for future classifiers (AAAI16). Batch learns a classifier by using all training data $\mathcal{D}$ at once. Online learns a classifier $\boldsymbol{w}_t$ by maximizing the log likelihood of data $\mathcal{D}_t$ with a prior $p(\boldsymbol{w}_t) = \mathcal{N}(\boldsymbol{w}_t|\boldsymbol{w}_{t-1}, c\mathbf{I})$, where c is a regularization parameter, in turn from $\boldsymbol{w}_{t_1}$ to $\boldsymbol{w}_{t_T}$. Online is considered as a weighing method, where the past data are used in the form of priors with weights. Present estimates a classifier with only recent training data $\mathcal{D}_{t_T}$. Present is also considered as a weighting method, where there is a weight only at the current time. The AAAI16 method is for learning future classifiers by using the one-step algorithm (Kumagai and Iwata 2016). With the proposed method, we ran the learning four times with different initial conditions, and selected the result with the highest lower bound. For Batch, Online and Present, we chose the regularization parameter from $\{10^{-1}, 1, 10^{1}\}$ in terms of which average AUC over all test time units was the best. For AAAI16, we set the parameters for the gamma priors as the same values as those in a previous study (Kumagai and Iwata 2016), ran the learning four times with different initial conditions, and changed the order of vector autoregressive models from $1, \ldots, 9$ in terms of which average AUC over all test time units was the best. For the proposed method and AAAI16, the number of iterations for learning was 2000 in all experiments.

4.5 Results

Table 1 shows the average and standard errors of AUCs over all test time units for all data sets. The proposed method achieved the highest AUC for all data sets except ELEC2. For ELEC2, the proposed method outperformed Batch, Online and Present although AAAI16 performed slightly better than the proposed method. Nevertheless, this results show that the proposed method can better maintain classification performance compared with the other methods.

Figure 2 shows the average and standard errors of AUCs for all test time units in the synthetic data sets. For both data sets, the proposed method maintained AUC at almost one over all test time units, although the AUCs of the others decreased quickly over time. Figure 3 shows an example of



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

Figure 4: Average and standard errors of AUCs for all test time units over ten runs

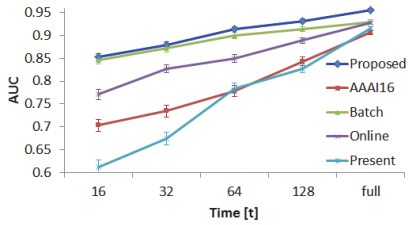


Figure 5: Average and standard errors of AUCs with proposed method with different numbers of training data on SPAM2

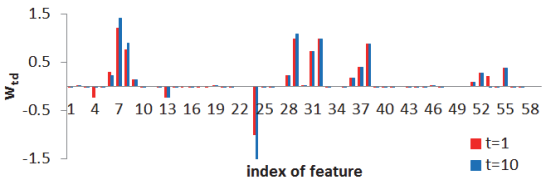


Figure 6: Classifier parameters estimated with proposed method on ONP. t represents time after training

classifier parameters estimated with the proposed method on Non-linear. The variances σ^2_{td} in Eq. (16) for the test inputs ($t \geq 21$) were large, although they were small for the training inputs ($t \leq 20$). This is the property of GP regression. We found that the estimated classifiers (the means m_{td} in Eq. (16)) nearly matches the ideal classifiers, whose AUC is one for all time units. From these results, we confirmed that the proposed method works effectively when the decision boundary dynamically changes over time.

Figure 4 shows the average and standard errors of AUCs for all test time units in the real-world data sets. The proposed method outperformed the others over all the test time units in SPAM2 and ONP. Note that the AUC of AAAI16 was worse than Batch, Online and Present. One of the reasons for the poor performance of AAAI16 is that it could not effectively capture the dynamics of the time-evolving decision boundary since they are probably complex in SPAM2 and ONP. In contrast, the proposed method performed well since GP regression is a flexible non-linear regression model; therefore, it could effectively capture the complex dynamics of the decision boundary. For BLOG, the proposed method outperformed the others in many test time units. For ELEC2, AAAI16 performed slightly better than the proposed method, while the proposed method outperformed Batch, Online, and Present. This indicates that the linearity assumption on AAAI16 matches ELEC2 well.

Next, we discuss the classification performance when varying the number of training data. Figure 5 shows the average and standard errors of AUCs with different number of training data at time t on SPAM2. The hyperparameters of all methods were the same as before. The proposed method outperformed the others whether the number of training data was small or large. These results suggest that the proposed method is robust against the number of training data.

Finally, we show that the proposed method can model the various dynamics of the classifier parameters. Figure 6 shows an example of classifier parameters for each feature w_{td}, where $t = 1$ and $t = 10$, on ONP. We found that many classifier parameters are very small, some parameters, such as those for the 4-th, 7-th, and 24-th features, dynamically change over time, and some parameters, such as those for the 31-st and 37-th features, are almost non-zero constant. Since the classifier parameters that do not affect the classification performance become small, the proposed method can select features that are useful for classification.

5 Conclusions

We propose a method involving a probabilistic model for learning future classifiers, whose dynamics is non-linear, given labeled data with timestamps collected until the current time, and a learning algorithm for the model based on empirical variational Bayesian inference by which uncertainty of dynamics can be incorporated for future classification. We conducted experiments, and showed that the proposed method can better maintain the classification performance over time compared with current methods. For future work, we will apply other kernel functions or deep learning methods to the framework of learning future classifiers.

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

2012 IEEE 24th International Conference on Tools with Artificial Intelligence, 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.

Bach, S., and Maloof, M. 2010. A bayesian approach to concept drift. In *Advances in Neural Information Processing Systems*, 127–135.

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.; Even-Dar, E.; Mansour, Y.; and Vaughan, J. W. 2010. Regret minimization with concept drift. *COLT 2010* 168.

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.

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.

Gao, J.; Ling, H.; Hu, W.; and Xing, J. 2014. Transfer learning based visual tracking with gaussian processes regression. In *European Conference on Computer Vision*, 188–203. Springer.

Haque, A.; Khan, L.; and Baron, M. 2016. Sand: semi-supervised adaptive novel class detection and classification over data stream. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Hoffman, J.; Darrell, T.; and Saenko, K. 2014. Continuous manifold based adaptation for evolving visual domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 867–874.

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,.

Kumagai, A., and Iwata, T. 2016. Learning future classifiers without additional data. In *Thirtieth AAAI Conference on Artificial Intelligence*.

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.

Liu, D. C., and Nocedal, J. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming* 45(1-3):503–528.

Loosli, G.; Canu, S.; and Bottou, L. 2007. Training invariant support vector machines using selective sampling. *Large Scale Kernel Machines* 301–320.

Ma, J.; Saul, L. K.; Savage, S.; and Voelker, G. M. 2009. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 681–688. ACM.

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

Rasmussen, C., and Williams, C. 2006. *Gaussian process for machine learning*. MITPress.

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

Royer, A., and Lampert, C. H. 2015. Classifier adaptation at prediction time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1401–1409.

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.

Wang, J.; Zhao, P.; and C.Hoi, S. 2012. Exact soft confidence-weighted learning. In *Proceedings of the 29th International Conference on Machine Learning*, 121–128.

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

Zhou, G.; Sohn, K.; and Lee, H. 2012. Online incremental feature learning with denoising autoencoders. In *International Conference on Artificial Intelligence and Statistics*, 1453–1461.

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.