Predicting Future Classifiers for Evolving Non-linear Decision Boundaries

Kanishka Khandelwal(✉), Devendra Dhaka, and Vivek Barsopia

NEC Corporation, Tokyo, Japan
{k_khandelwal,barsopiav}@nec.com, deven.dhaka@gmail.com

Abstract. In streaming data applications, the underlying concept often changes with time which necessitates the update of employed classifiers. Most approaches in the literature utilize the arriving labeled data to continually update the classifier system. However, it is often difficult/expensive to continuously receive the labels for the arriving data. Moreover, in domains such as embedded sensing, resource-aware classifiers that do not update frequently are needed. To tackle these issues, recent works have proposed to predict classifiers at a future time instance by additionally learning the dynamics of changing classifier weights during the initial training phase. This strategy bypasses the need to retrain/relearn the classifiers, and thus the additional labeled data is no longer required. However, the current progress is limited to the prediction of linear classifiers. As a step forward, in this work, we propose a probabilistic model for predicting future non-linear classifiers given time-stamped labeled data. We develop a variational inference based learning algorithm and demonstrate the effectiveness of our approach through experiments using synthetic and real-world datasets.

Keywords: Concept drift · Data streams · Classification

1 Introduction

In dynamically changing environments, the underlying concept that determines the response variable y, for the covariate $\mathbf{x}$, may vary over time. This variation is reflected as a change in the conditional distribution $p(y|\mathbf{x})$ which makes the environment non-stationary. For example, a user's preference for online news articles varies over time owing to the geopolitical/social factors. Consequently, the user's click pattern for news articles will change, even if the topics of published articles remain the same.

In applications where data arrives sequentially, this phenomenon is common and is referred to as *real* concept drift [10]. In a classification context, this would mean the underlying decision boundary is evolving over time. A classifier employed in such an environment should be updated to avoid a deterioration in the performance. In the above example, the news provider must update its

D.Dhaka—Contributed while working at NEC Corporation.

F. Hutter et al. (Eds.): ECML PKDD 2020, LNAI 12457, pp. 628–643, 2021.
https://doi.org/10.1007/978-3-030-67658-2_36

classifier to continue recommending articles in line with the user's preference; else the churn rate increases. To achieve this, most approaches in the literature propose to update the decision rule of classifier (or classifier system) in an online manner using the arriving labeled data. Of these, some are *passive* approaches that continuously update the decision rule while others are *active* approaches that use change detectors to trigger an update [2].

However, there are two issues associated with this online update strategy when the decision boundary is evolving. *Firstly*, it is often expensive/difficult to continuously acquire labels for the sequentially arriving data since labeling is done by domain experts in most cases. And if labels are unavailable, it is not possible to detect changes in $p(y|\mathbf{x})$ without taking any assumption [22,24]. *Secondly*, it could be impractical to allow such online updates in computationally constrained applications. For example, adaptive classifiers are needed in wearable embedded systems that collect physiological sensor data to perform tasks such as physical activity recognition [1,34]. Updating the decision rule in an online fashion becomes prohibitive in such systems with limited memory and processing capabilities [2,23].

The above issues motivate the research on classifier models that can avoid the online update step using labeled data yet cope with the situation of evolving (underlying) decision boundary. In this work, we propose a model to predict future classifier (i.e. weights of the classifier at a future time instance) to maintain the classification performance. We achieve this by jointly learning the dynamics of an evolving boundary along with the classifier weights using time-stamped labeled data during the training phase. The learned dynamics and past values of classifier weights can be used to predict the weights at a future time instance thus avoiding the need for an online update of the classifier weights using labeled data. The advantages of predicting future classifier are twofold: *first*, effective use of available training data makes it possible to (reasonably) approximate the evolving boundary for some future time instances which reduces the cost of labeling the arriving data; *second*, the online update step is no longer necessary and thus the classifier system can be deployed in resource-constrained applications.

The current progress in this direction is limited to the prediction of future linear classifier [20–22] which is expected to perform poorly if the (evolving) underlying decision boundary is non-linear (i.e. the classes are not linearly separable). As a step forward, we propose a probabilistic model to predict a non-linear classifier at a future time instance. We utilize the non-parametric framework of Dirichlet process mixture model [8] to model the joint distribution of label y and covariates x, in the form $p(x, y) = p(y|x)p(x)$ and allow $p(y|x)$ to change over time. Further, we develop a variational inference based learning algorithm and demonstrate the effectiveness of our approach through experiments using both synthetic and real datasets.

In the next section, we discuss the related work in this domain and highlight the challenges in designing methods for future non-linear classifier prediction. In Sect. 3, the task is formally defined which is followed by a description of our model. Next, we develop a variational-inference based learning algorithm

in Sect. 4. We present the evaluation results of our approach on synthetic and real-world datasets in Sect. 5 and summarize this paper in Sect. 6.

2 Related Work

Changing data distribution in streaming data applications is a well-studied problem and many methods have been proposed to maintain the classifier performance in such non-stationary conditions. Online algorithms [6,33], forgetting algorithms [17,19,27], ensemble methods [5,18], evolving fuzzy systems [3,30] etc. usually capitalize on arriving labeled data for updating the classifier. However, these methods are not well suited for applications where acquiring labeled data is difficult/expensive. Active learning algorithms [36,37] look to optimize the labeling cost by querying labels for specific instances. Some methods have been proposed [13–15] that use unlabeled data with/without the labeled data. However, all of the above methods look to update the decision rule in an online manner using the arriving data. As explained in the previous section, this strategy could be prohibitive in applications where resources are limited.

To alleviate these issues, a possible *proactive* strategy is to predict future classifiers using the labeled training data at hand. In this direction, Kumagai and Iwata [20–22] have proposed probabilistic models for prediction of future classifiers by learning the dynamics of evolving decision boundary during the initial training phase. The intuition behind their approach is as follows. The weights of a classifier employed in an environment where the decision boundary is evolving ought to change with time to maintain the performance. Using the time-stamped labeled data available in the initial training phase, the classifier can be trained at different time instances in the past. A time series model fitted over these weights can capture the dynamics of the decision boundary. Thus, the past values of classifier weights and the time series model can be used to predict the classifier at future time instances.

In their initial work [20], logistic regression is used as a classifier model and autoregressive (AR) process as a time series model over classifier weights. Since the AR process could only capture the linear dynamics of the decision boundary, in [21] the authors proposed to use Gaussian Process (GP) as the time series model over the weights in logistic regression model for capturing the non-linear dynamics of the decision boundary. The GP-based model was further extended in [22] to utilize the unlabeled data available at test (future) time instances. Low-density separation criterion, i.e. decision boundary should pass through low density regions, is used to regularize the posterior of the classifier parameters which may improve the prediction of future classifiers in some scenarios. On the other hand, waiting for a few unlabeled batches of data (belonging to test time instances) makes this approach inappropriate for the real-time prediction task.

In all of these works, logistic regression is used for the classification task which limits their applicability to linearly separable settings and is bound to perform poorly in scenarios where the decision boundary is non-linear. Extending these models using a non-linear classifier in place of logistic regression classifier is difficult. A non-parametric non-linear classification algorithm (like GP classification,

nearest neighbors, SVM, etc.) learns a data-dependent classifier. Since the data samples can be different at different time instances, establishing a correspondence between the learned weights across time instances is not obvious; thus the time series model cannot be applied straightforwardly. On the other hand, with parametric classifiers (such as Multilayer perceptron) there is an issue of the huge parameter space. The model formulation and training would be difficult since the model should also identify the subset of parameters to be modeled using time series.

3 Proposed Method

3.1 Task

We consider a binary classification task with an available set $\mathcal{D}$ of datasets $\mathcal{D}_t := \{(\mathbf{x}_{t,i}, y_{t,i})\}_{i=1}^{N_t}$ collected at regular intervals with time-stamp denoted using $t \in \{1, 2 \dots T\}$. Here, $\mathbf{x}_{t,i} \in \mathbb{R}^D$ is the D-dimensional covariate vector of the i-th sample at time t, with $y_{t,i} \in \{0, 1\}$ as the class label and N_t is the number of samples available at time t. Note that sequential time-stamped data can be represented in this format by discretizing at regular intervals and considering the data falling within same interval to have same time-stamp. Given the set of training data $\mathcal{D} = \{D_t\}_{t=1}^{T}$, our aim is to predict binary classifier $h_t : \mathbb{R}^D \rightarrow \{0, 1\}$ at future time instances $t \in \{T + 1, T + 2, ...\}$ to precisely classify the future data.

3.2 Model

The non-parametric mixture model based on Dirichlet process (DP) [8] and its extensions form a well-known family of probabilistic models used for tasks such as topic modelling [11], regression [35], etc. The DP framework assumes that a countably infinite set of mixture components is needed to generate the available data, but, the DP prior (with a concentration parameter α) exhibits a clustering property [31]. Our model is among the DP-GLM family [12,29] and is explicitly developed for a non-stationary environment wherein $p(y|\boldsymbol{x})$ is changing. Each component within a mixture locally models the distribution of covariates as well as a linear (classifying) relationship between the response variable and covariates, separately at all time instances. Additionally, within a component, a time series model is used to capture the dynamics of changing classifier weights. Globally, a mixture of such components is expected to model a time-varying non-linear relationship between the response variable and covariates.

Figure 1 shows the graphical representation of our model and below we explain the generative process of a sample pair $(\mathbf{x}_{t,i}, y_{t,i})$. Our model first generates π_k independently for each component $k \in \{1, 2, \cdot \cdot \infty\}$ with the distribution $Beta(1, \alpha)$. The mixture weight given to each component, π'_k, is determined from $\pi = \{\pi_k\}_{k=1}^{\infty}$ using the stick breaking process representation of DP [28] as,

$$\pi'_k = \pi_k \Pi_{k'=1}^{k-1}(1 - \pi_{k'}) \tag{1}$$

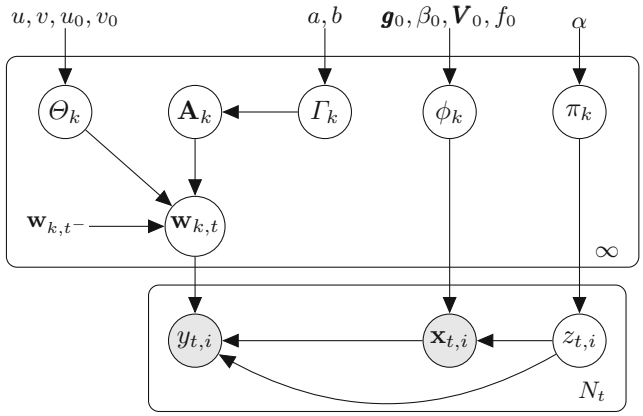

Fig. 1. Graphical model representation of the proposed model at time t. Here, $\mathbf{w}_{k,t^-} := \{\mathbf{w}_{k,t-m}\}_{m=1}^{M}$, $\Theta_k := \{(\theta_k, \theta_{0,k})\}$, $\Gamma_k := \{\gamma_{k,m}\}_{m=0}^{M}$. Nodes with gray and empty circles are observed and latent variables, respectively. Nodes with deterministic value are without circles. Note that $\mathbf{w}_{k,t^-}$ is already determined by the time t

The component assignment variable, $z_{t,i}$, for our sample is obtained from a multinomial distribution as,

$$z_{t,i}|(\pi'_1, \pi'_2, \ldots) \sim Mult(\pi'_1, \pi'_2, \ldots) \tag{2}$$

Here, $z_{t,i}$ is represented as an infinite length indicator vector with the entry corresponding to the selected component being 1, and 0 for all others. Given $z_{t,i}$, the covariate $\mathbf{x}_{t,i}$ is obtained from a multivariate normal distribution as,

$$\mathbf{x}_{t,i}|z_{t,i}, \{(\mathbf{m}_k, \mathbf{R}_k^{-1})\}_{k=1}^{\infty} \sim \mathit{\Pi}_{k=1}^{\infty} \left(\mathcal{N}(\mathbf{m}_k, \mathbf{R}_k^{-1})\right)^{z_{t,i,k}} \tag{3}$$

The mean $\mathbf{m}_k \in \mathbb{R}^D$ and precision matrix $\mathbf{R}_k \in \mathbb{R}^{D \times D}$ for each component are drawn from a Normal-Wishart distribution with mean vector $g_0 \in \mathbb{R}^D$, scale matrix $V_0 \in \mathbb{R}^{D \times D}$, scale parameter $\beta_0 \in \mathbb{R}$ and degree of freedom $f_0 \in \mathbb{R}$ as,

$$\phi_k = \{\mathbf{m}_k, \mathbf{R}_k\} \sim \mathcal{NW}(\boldsymbol{g}_0, \beta_0, \boldsymbol{V}_0, f_0) \tag{4}$$

Note that ϕ_k doesn't vary with time since we assume the covariate distribution $p(x)$ to be stationary. The binary label $y_{t,i}$ is drawn from a Bernoulli (Ber) distribution with the "success" parameter given by standard logistic regression of $\mathbf{x}_{t,i}$ as,

$$y_{t,i}|z_{t,i}, \mathbf{x}_{t,i}, \{\mathbf{w}_{k,t}\}_{k=1}^{\infty} \sim \mathit{\Pi}_{k=1}^{\infty} \left(Ber(\sigma(\mathbf{w}_{k,t}^T \mathbf{x}_{t,i}))\right)^{z_{t,i,k}} \tag{5}$$

Here, $\sigma(\cdot)$ is the sigmoid function and $\mathbf{w}_{k,t}$ are the classifier weights for the component k at time t. Same as [20], we use simple AR process [26] with lag order M to model the dynamics of classifier weights $\mathbf{w}_{t,k}$, and their values are

obtained as,

$$\mathbf{w}_{k,t}|\mathbf{A}_k, \theta_k \sim \mathcal{N}(\sum_{m=1}^{M} \mathbf{A}_{k,m}\mathbf{w}_{k,t-m} + \mathbf{A}_{k,0},\ \theta_k^{-1}\mathbf{I}_D) \quad t > M$$
$$\mathbf{w}_{k,t}|\theta_{0,k} \sim \mathcal{N}(\mathbf{0},\ \theta_{0,k}^{-1}\mathbf{I}_D) \quad t \leq M \tag{6}$$

where $\mathbf{A}_{k,1}, \mathbf{A}_{k,2}, \ldots \mathbf{A}_{k,M} \in \mathbb{R}^{D\times D}$ define the classifier dynamics, $\mathbf{A}_{k,0} \in \mathbb{R}^D$ is the bias term, and $\theta_{0,k}, \theta_k \in \mathbb{R}^+$ are precision parameters. In the above equation, the classifier weights at time t depends on the weights of the same component at past M time instances, through linear dynamics. Since the classifier weights $\mathbf{W_k} := \{\mathbf{w}_{k,t}\}_{t=1}^T$ and time series parameters $\mathbf{A_k} := \{\mathbf{A}_{k,m}\}_{m=0}^M$ are not shared across the components, our model has the flexibility to learn separate dynamics for each component. For the sake of simplicity, we restrict $\mathbf{A}_{k,m}$ to be a diagonal matrix and for convenience use the same notation for the vector of diagonal elements of $\mathbf{A}_{k,m}$. This implies the dth dimension of classifier weight $w_{k,t,d}$, where $d \in \{1, \ldots, D\}$, depends only on its own past values $w_{k,t-1,d}, \ldots w_{k,t-M,d}$. For $t \leq M$ we assume each $\mathbf{w}_{k,t}$ is generated from $\mathcal{N}(\mathbf{0}, \theta_{0,k}^{-1}\mathbf{I}_D)$. The time series parameters $\mathbf{A}_{k,m}$ for $m = 0, \ldots, M$ are generated from multivariate normal distribution as,

$$\mathbf{A}_{k,m}|\gamma_{k,m} \sim \mathcal{N}(\mathbf{0}, \gamma_{k,m}^{-1}\mathbf{I}_D) \tag{7}$$

The precision parameters $\theta_{0,k}, \theta_k, \gamma_{k,m}$ are sampled from the Gamma priors as,

$$\theta_{0,k} \sim \Gamma(u_0, v_0), \quad \theta_k \sim \Gamma(u, v), \quad \gamma_{k,m} \sim \Gamma(a, b) \tag{8}$$

where u_0, v_0, u, v, a, b are fixed hyperparameters.

Correspondingly, the joint distribution of the labeled data $\mathcal{D}$, location parameters $\mathbf{\Phi} := \{\phi_k\}_{k=1}^\infty$, classifier weights $\mathbf{W} := \{\mathbf{W}_k\}_{k=1}^\infty$ and their precision parameters $\mathbf{\Theta} := \{(\theta_k, \theta_{0,k})\}_{k=1}^\infty$, dynamics' parameters $\mathbf{A} := \{\mathbf{A_k}\}_{k=1}^\infty$ and their precision parameters $\Gamma := \{\{\gamma_{k,m}\}_{m=0}^M\}_{k=0}^\infty$, assignment variables $\mathbf{Z} := \{\{z_{t,i}\}_{i=1}^{N_t}\}_{t=1}^T$ and stick components π can be written as,

$$p(\mathcal{D}, \mathbf{W}, \mathbf{A}, \mathbf{Z}, \mathbf{\Phi}, \mathbf{\Theta}, \Gamma, \pi) = p(\mathcal{D}|\mathbf{\Phi}, \mathbf{W}, \mathbf{Z})p(\mathbf{W}|\mathbf{A}, \mathbf{\Theta})$$
$$\times\ p(\mathbf{A}|\Gamma)p(\mathbf{Z}|\pi)p(\Gamma)p(\mathbf{\Theta})p(\mathbf{\Phi})p(\pi) \tag{9}$$

We have omitted the hyperparameters from the notation.

4 Inference

In this section we develop a learning algorithm to compute the posterior distribution of hidden variables i.e. $p(\mathbf{W}, \mathbf{A}, \mathbf{Z}, \mathbf{\Phi}, \mathbf{\Theta}, \Gamma, \pi|\mathcal{D})$. Since exact posterior inference is computationally intractable because of evidence calculation, we take a variational inference-based approach and approximate the model posterior using the distribution $q(\mathbf{W}, \mathbf{A}, \mathbf{Z}, \Phi, \mathbf{\Theta}, \Gamma, \pi)$ (shortly referred as $q(\cdot)$).

4.1 Evidence Lower Bound (ELBO)

In a standard fashion, we aim to minimize the reverse KL divergence of the model posterior with respect to the variational distribution $q(\cdot)$ [4]. Since the model evidence $p(\mathcal{D})$ is constant with respect to $q(\cdot)$, the minimization task is equivalent to the maximization of the lower bound of log evidence $L(q)$ given as,

$$L(q) := \mathbb{E}_q\big[\log p(\mathcal{D}, \mathbf{W}, \mathbf{A}, \mathbf{Z}, \Phi, \boldsymbol{\Theta}, \Gamma, \pi)\big] + H(q) \tag{10}$$

with respect to free variational parameters of the distribution $q(\cdot)$ where $H(q)$ denotes the entropy of distribution $q(\cdot)$. The logistic regression introduces non-conjugacy into our model. To tackle this issue, we use the following Gaussian lower bound [16]:

$$p(y_{t,i}|\mathbf{x}_{t,i}, \mathbf{w}_{k,t}) \;\geq\; \sigma(\xi_{k,t,i}) \exp\left(\frac{2y_{t,i}b - b - \xi_{k,t,i}}{2} + h(\xi_{k,t,i})(b^2 - \xi_{k,t,i}^2)\right)$$

where $b := \mathbf{w}_{k,t}^T \mathbf{x}_{t,i}$ and $h(\xi_{k,t,i}) = \frac{2\sigma(\xi_{k,t,i})-1}{4\xi_{k,t,i}}$. This inequality introduces an extra set of parameters ξ for all samples in $\mathcal{D}$ to approximate the logistic expression in each component mixture. This gives us a new variational objective function $L(q, \xi)$, a lower bound to $L(q)$, with additional parameters ξ to estimate. Maximizing the newly obtained variational objective function $L(q, \xi)$ is equivalent to minimizing the KL divergence between $q(\cdot)$ and model posterior $p(\cdot)$, with accuracy of the approximation given by parameters ξ.

4.2 Variational Distributions and Update Equations

For analytical purposes, we take a mean-field approximation and restrict the distribution $q(\cdot)$ to a fully factorized form. We utilize truncated representation of DP's stick-breaking process [32] in our variational distribution and fix a truncation level K by enforcing $q(\pi_K = 1) = 1$. That is, the component $k = K$ takes all of the stick left after $K - 1$ splits and the remaining (infinite) components are assigned zero probability of being chosen by the assignment variable. The value of K can be chosen before the start of the learning process. Putting it all together, $q(\cdot)$ can be represented as,

$$q(\cdot) := \prod_{k=1}^{K}\prod_{t=1}^{T} q(\mathbf{w}_{k,t}) \prod_{k=1}^{K}\prod_{m=0}^{M} q(A_{k,m}) q(\gamma_{k,m}) \times \prod_{t=1}^{T}\prod_{i=1}^{N_t} q(z_{t,i}) \prod_{k=1}^{K} q(\phi_k) q(\theta_k) q(\theta_{0,k}) q(\pi_k) \tag{11}$$

The log of the optimal solution for each factor in $q(\cdot)$ in the above equation is obtained by taking the expectation of log of the joint distribution given in (9) with respect to all other factors [4]. Using this method, the optimal forms are

derived as,

$$
\begin{aligned}
&q(\pi'_k) = Beta(a^{\pi}_k, b^{\pi}_k), && q(z_{t,i}) = Mult(\nu_{t,i,1} \dots \nu_{t,i,K}) \\
&q(\phi_k) = \mathcal{NW}(\boldsymbol{g}_k, \beta_k, \boldsymbol{V}_k, f_k), && q(\mathbf{w}_{k,t}) = \prod_{d=1}^{D} \mathcal{N}(\eta_{k,t,d}, \lambda^{-1}_{k,t,d}) \\
&q(\theta_{0,k}) = \Gamma(u^{\theta_0}_k, v^{\theta_0}_k), && q(\mathbf{A}_{k,m}) = \mathcal{N}(\boldsymbol{\mu}_{k,m}, \boldsymbol{S}^{-1}_{k,m}) \\
&q(\theta_k) = \Gamma(u^{\theta}_k, v^{\theta}_k), && q(\gamma_{k,m}) = \Gamma(a^{\gamma}_{k,m}, b^{\gamma}_{k,m})
\end{aligned} \tag{12}
$$

where $\eta_{k,t,d} \in \mathbb{R}$, $\boldsymbol{g}_k, \boldsymbol{\mu}_{k,m} \in \mathbb{R}^D$ and $\boldsymbol{V}_k, \boldsymbol{S}_{k,m} \in \mathbb{R}^{D \times D}$ and the remaining parameters belong to $\mathbb{R}$. Corresponding update equations for the parameters in (12) are provided below. The variational posterior distribution $q(\cdot)$ is obtained by iterating through these equations in a cyclic fashion until a convergence criterion is satisfied.

$$
a^{\pi}_k = 1 + \sum_{t=1}^{T} \sum_{i=1}^{N_t} \nu_{k,t,i}, \quad b^{\pi}_k = \alpha + \sum_{k'=k+1}^{K} \sum_{t=1}^{T} \sum_{i=1}^{N_t} \nu_{k,t,i}, \quad \boldsymbol{g}_k = \frac{1}{\beta_k}(\beta_0 \boldsymbol{g}_0 + \bar{\nu}_k \bar{\mathbf{x}}_k)
$$

$$
\beta_k = \beta_0 + \bar{\nu}_k, \quad f_k = f_0 + \bar{\nu}_k, \quad \mathbf{V}^{-1}_k = \boldsymbol{V}^{-1}_0 + \bar{\nu}_k \bar{\sigma}_k + \frac{\beta_0 \bar{\nu}_k}{\beta_0 + \bar{\nu}_k}(\bar{\mathbf{x}}_k - \boldsymbol{g}_0)(\bar{\mathbf{x}}_k - \boldsymbol{g}_0)^T
$$

$$
u^{\theta}_k = u + \frac{1}{2}(T - M)D, \quad u^{\theta_0}_k = u_0 + \frac{1}{2}MD, \quad v^{\theta_0}_k = v_0 + \frac{1}{2}\sum_{t=1}^{T} ||\boldsymbol{\eta}_{k,t}||^2 + Tr(\boldsymbol{\Lambda}^{-1}_{k,t})
$$

$$
\begin{aligned}
v^{\theta}_k = v + \frac{1}{2} \sum_{t=M+1}^{T} \Bigg(& Tr(\boldsymbol{S}^{-1}_{0,k} + \boldsymbol{\Lambda}^{-1}_{k,t}) + ||\boldsymbol{\eta}_{k,t} - \sum_{l=1}^{M} dg(\boldsymbol{\mu}_{k,l}) \boldsymbol{\eta}_{k,t-l} - \boldsymbol{\mu}_{k,0}||^2 \\
& + \sum_{l=1}^{M} \boldsymbol{\eta}^T_{k,t-l} \boldsymbol{S}^{-1}_{k,l} \boldsymbol{\eta}_{k,t-l} + \sum_{l=1}^{M} \Big\{ Tr(dg(\boldsymbol{\mu}_{k,l})^2) \boldsymbol{\Lambda}^{-1}_{k,t-l} + Tr(\boldsymbol{S}^{-1}_{k,l} \boldsymbol{\Lambda}^{-1}_{k,t-l}) \Big\} \Bigg) \\
& \qquad \forall k = 1 \dots K
\end{aligned}
$$

$$
\log \nu_{t,i,k} \propto \sum_{k'=1}^{k-1} \psi(b^{\pi}_{k'}) - \psi(a^{\pi}_{k'} + b^{\pi}_{k'}) + \psi(a^{\pi}_k) - \psi(a^{\pi}_k + b^{\pi}_k) + \frac{1}{2}\Psi(f_k/2)
$$

$$
+ \frac{1}{2}\left(D \log 2 + \log |\boldsymbol{V}_k|\right) + (y_{t,i} - 0.5)\boldsymbol{\eta}^T_{k,t} \mathbf{x}_{t,i} - \mathbf{x}^T_{t,i}[\boldsymbol{\eta}_{k,t} \boldsymbol{\eta}^T_{k,t} + \boldsymbol{\Lambda}^{-1}_{k,t}] \mathbf{x}_{t,i} h(\xi_{t,i,k})
$$

$$
- \frac{1}{2}\left(\frac{D}{\beta^{-1}_k} + f_k (\mathbf{x}_{t,i} - \boldsymbol{g}_k)^T \boldsymbol{V}_k (\mathbf{x}_{t,i} - \boldsymbol{g}_k) \right) \forall k = 1 \dots K, i = 1 \dots N_t, t = 1 \dots T
$$

$$
\eta_{k,t,d} = \lambda^{-1}_{k,t,d} \Bigg[\sum_{i=1}^{N_t} \nu_{k,t,i} \Big((y_{t,i} - 0.5) x_{t,i,d} - 2h(\xi_{t,i,k}) \sum_{l \neq d} \eta_{k,t,l} \mathbf{x}_{t,i,l} \mathbf{x}_{t,i,d} \Big)
$$

$$
+ \frac{u^{\theta}_k}{v^{\theta}_k} \sum_{l=1}^{K^t_1} \Big((\eta_{k,m+l,d} - \mu_{k,0,d}) \mu_{k,m+l-t,i} - \sum_{l' \neq m+l-t}^{m} \eta_{k,m+l-l',d} \cdot \mu_{k,l',d} \cdot \mu_{k,m+l-t,d} \Big) \Bigg]
$$

$$\boldsymbol{\Lambda}_{k,t} = \frac{u_k^\theta}{v_k^\theta}\sum_{l=1}^{K_1^t}\Big(dg(\boldsymbol{\mu}_{k,m-t+l})^2 + \boldsymbol{S}_{k,m-t+l}^{-1}\Big) + \frac{u_k^{\theta_0}}{v_k^{\theta_0}}\mathbf{I}_d + 2\sum_{i=1}^{N_t} h(\xi_{t,i,k} dg(\mathbf{x}_{t,i})^2)$$
$$\forall k = 1,\dots K, \quad t = 1,\dots M, \quad d = 1,\dots D$$

$$\eta_{k,t,d} = \lambda_{k,t,d}^{-1}\Bigg(\frac{u_k^\theta}{v_k^\theta}\Big[\sum_{l=1}^{K_2^t}\Big((\eta_{k,t+l,d} - \mu_{k,0,d})\mu_{k,l,d} - \sum_{l'\neq l}^{m}\eta_{k,t+l-l',d}\cdot\mu_{k,l',d}\cdot\mu_{k,l,d}\Big)$$
$$+\sum_{l=1}^{m}\eta_{k,t-l,d}\mu_{k,l,d} + \mu_{k,0,d}\Big] + \sum_{i=1}^{N_t}\nu_{k,t,d}\mathbf{x}_{t,i,d}\Big\{y_{t,i} - \frac{1}{2} - 2h(\xi_{t,i,k})\sum_{l\neq d}\eta_{k,t,l}\mathbf{x}_{t,i,l}\Big\}\Bigg)$$

$$\boldsymbol{\Lambda}_{k,t} = \frac{u_k^\theta}{v_k^\theta}\sum_{l=1}^{K_2^t}(dg(\boldsymbol{\mu}_{k,l})^2 + \boldsymbol{S}_{k,l}^{-1}) + \frac{u_k^{\theta_0}}{v_k^{\theta_0}}\mathbf{I}_d + 2\sum_{i=1}^{N_t} h(\xi_{t,i,k}) dg(\mathbf{x}_{t,i})^2$$
$$k = 1,\dots K, \quad t = M+1,\dots T, \quad d = 1,\dots D$$

$$(\xi_{t,i,k})^2 = \mathbf{x}_{t,i}^T(\boldsymbol{\Lambda}_{t,k}^{-1} + \boldsymbol{\eta}_{t,k}\boldsymbol{\eta}_{t,k}^T)\mathbf{x}_{t,i} \qquad k = 1,\dots K, t = 1,\dots T, i = 1,\dots N_t$$

$$\boldsymbol{S}_{k,m} = \frac{a_{k,m}^\gamma}{b_{k,m}^\gamma}\mathbf{I}_d + \frac{u_k^\theta}{v_k^\theta}\sum_{t=m+1}^{T}(dg(\boldsymbol{\eta}_{k,t-m})^2 + \boldsymbol{\Lambda}_{k,t-m}^{-1}), \boldsymbol{S}_{k,0} = (\frac{a_{k,0}^\gamma}{b_{k,0}^\gamma} + (T-M)\frac{u_k^\theta}{v_k^\theta})\mathbf{I}_D$$

$$a_{k,m}^\gamma = a + \frac{1}{2}D, \quad \boldsymbol{\mu}_{k,m} = \boldsymbol{S}_{k,m}^{-1}\frac{u_k^\theta}{v_k^\theta}\sum_{t=m+1}^{T} dg\Big(\boldsymbol{\eta}_{k,t} - \sum_{l\neq m} dg(\boldsymbol{\mu}_{k,l})\boldsymbol{\eta}_{k,t-l} - \boldsymbol{\mu}_0\Big)\boldsymbol{\eta}_{k,t-m}$$

$$b_{k,m}^\gamma = b + \frac{1}{2}(||\boldsymbol{\mu}_{k,m}||^2 + Tr(\boldsymbol{S}_{k,m}^{-1})), \ \boldsymbol{\mu}_{k,0} = \boldsymbol{S}_{k,0}^{-1}\frac{u_k^\theta}{v_k^\theta}\sum_{t=1}^{T}(\boldsymbol{\eta}_{k,t} - \sum_{m=1}^{M} dg(\boldsymbol{\mu}_{k,m})\boldsymbol{\eta}_{k,t-m})$$
$$k = 1....K, \quad m = 1...M$$

where $\bar{\sigma}_k = \frac{1}{\bar{\nu}_k}\sum_t^T\sum_i^{N_t}\nu_{t,i,k}(\mathbf{x}_{t,i} - \bar{\mathbf{x}}_k)(\mathbf{x}_{t,i} - \bar{\mathbf{x}}_k)^T$, $\bar{\nu}_k = \sum_t^T\sum_i^{N_t}\nu_{t,i,k}$, $\bar{\mathbf{x}}_k = \frac{1}{\bar{\nu}_k}\sum_t^T\sum_i^{N_t}\nu_{t,i,k}\mathbf{x}_{t,i}$, $\boldsymbol{\Lambda}_{k,t} \in \mathcal{R}^{D\times D}$ is a diagonal matrix with diagonal elements as $(\lambda_{k,t,1},\dots\lambda_{k,t,D})$, $\psi(\cdot)$ and $\Psi(\cdot)$ are digamma and polygamma functions, $K_1^t := min(t, T-m)$, $K_2^t := min(m, T-t)$, $dg(\mathbf{x})$ is a diagonal matrix whose diagonal elements are $\mathbf{x} = (x_1,\dots x_d)$, $h(x) = \frac{1}{2x}(\sigma(x) - \frac{1}{2})$, $||\cdot||$ is l_2-norm, and Tr represents the trace.

4.3 Prediction

For unlabeled time-stamped data sample $\mathbf{x}_{T',i}$ arrived at time $T' > T$, the probability of its label being 1 can be obtained as,

$$Pr(y_{T',i} = 1|\mathbf{x}_{T',i}) = \sum_{k=1}^{K}\omega_{k,T',i}c_{k,T',i} \tag{13}$$

where $c_{k,T',i}$ is the prediction by classifier at time T' of component k obtained using AR process and $\omega_{k,T',i} = P(z_{T',i,k} = 1|\mathbf{x}_{T',i})$ is the probabilistic weight of

the component in the mixture. Here,

$$\begin{aligned}\omega_{k,T',i} &\propto Pr(z_{T',i,k}=1)p(\mathbf{x}_{T',i}|\mathbf{\Phi}, z_{T',i,k}=1)\\ &= Pr(z_{T',i,k}=1)\mathcal{N}(\mathbf{x}_{T',i}|\mathbf{m}_k, \mathbf{R}_k^{-1})\end{aligned} \tag{14}$$

Note that, $\sum_{k=1}^{K}\omega_{k,T',i}=1$. The prior weight of a component k in the mixture $Pr(z_{T',i,k}=1)$ can be approximated as $\frac{\sum_{t=1}^{T}\sum_{i=1}^{N_t}\nu_{k,t,i}}{N}$, where N denotes total number of samples in the training data $\mathcal{D}$.

5 Experiments

In this section, we first describe the methods used for comparison and the experimental settings that were used to evaluate the performance of the proposed method. Next, we describe the synthetic and real-world data used for experiments and present the obtained results.

Comparison Methodology. We compare our method's performance against AAAI16 [20], AAAI17 [21], KDD18 [22], RF-Batch and RF-Present. AAAI17 and AAAI16 are the baseline approaches for predicting future linear classifiers. KDD18 is an extension to AAAI17 which utilizes the unlabeled data at test time instances for the training purpose. RF is a random forest classifier trained using the labeled data available at training time instances. RF-Batch uses all of the available training data $\mathcal{D}_{1:T}$ to learn the classifier while RF-Present uses the data available at the latest time instance i.e. $\mathcal{D}_T$. We use the area under the ROC curve (AUC) as an evaluation metric, a well-used measure for classification tasks, which was also used in the previous studies [21,22]. The performance results presented are obtained by aggregating the AUC values at test time instances and are averaged over five independent runs for each experiment.

Settings. For the proposed method, we set the value of hyperparameters as follows. We choose same values as [20] for Gamma prior hyperparameters: $u_0 = u = a = 1, v_0 = v = b = 0.1$. Since we normalize the data, the mean vector $\mathbf{g}_0$ and scale matrix $\mathbf{V}_0$ of Normal-Wishart distribution are set as $\mathbf{0}$ and $\mathbf{I}_D$, respectively. The scale parameter β_0 and degree of freedom f_0 should be greater than 0 and $D-1$, respectively. α, the concentration parameter of the Dirichlet Process, should be a positive real. As $\alpha \to 0$, the concentration property of the Dirichlet Process increases, and the number of inferred components decreases. K is the truncation parameter which determines the maximum number of inferred components and is user-specified. M is the lag order of the AR process and should be less than T in our model. We set their values as $\beta_0 = 2, f_0 = D+2, \alpha = 2, K = 6, M = 5$ from preliminary experiments. In AAAI16, AAAI17, and KDD18, we set the hyperparameters as suggested by authors in [21] and vary the AR lag order for AAAI16 from $1, \ldots 9$. All of these methods are variational inference-based and we run each of them till convergence.

5.1 Datasets

Synthetic Data. We consider two non-stationary settings (datasets) to test our method's competence in the prediction of future non-linear classifiers. Each setting is described briefly followed by the generative process for a sample pair $(\mathbf{x}_{t,i}, y_{t,i}) \in \mathcal{D}_t$.

Multimodal: In this experiment, covariate vector is sampled from a two-dimensional bimodal Gaussian mixture distribution. Decision boundary in one mode is rotating while in the second mode it is horizontal and moving vertically. We expect our method to infer two components one for each mode and learn the two different dynamics separately.

1. $z_{t,i} \sim Bernoulli(0.5)$
2. $\mathbf{x}_{t,i}|z_{t,i} \sim z_{t,i}\mathcal{N}(\mathbf{m}_1, 0.2\mathbf{I}_2) + (1 - z_{t,i})\mathcal{N}(\mathbf{m}_2, 0.2\mathbf{I}_2)$ where $\mathbf{x}_{t,i} \in \mathbb{R}^2, \mathbf{m}_1 = (0, 0), \mathbf{m}_2 = (2, 0)$
3. If $z_{t,i} = 1$, $y_{t,i} = 1$ if $\mathbf{x}_{t,i}^T(\cos\psi, \sin\psi) \geq 0$ else 0; $\psi = \pi(t-1)/4$ If $z_{t,i} = 0$, $y_{t,i} = 1$ if $x_{t,i,2} + (t-15)/150 \geq 0$ else 0

XOR: We conduct this experiment to emulate the classic example of XOR classification where covariates in the opposite quadrants have same class labels. At any given time instance, we can approximate the underlying decision boundary using just two linear classifiers. We take uniform stationary distribution for the covariates, and gradually rotate the decision boundary.

1. $\mathbf{x}_{t,i,d} \sim Uniform(-2, 2)$ where $d = \{1, 2\}$
2. $z'_{t,i} = sgn(x_{t,i,1}\cos\psi + x_{t,i,2}\sin\psi)$; $\psi = \pi(t-1)/8$
   $z''_{t,i} = sgn(-x_{t,i,1}\sin\psi + x_{t,i,2}\cos\psi)$; $\psi = \pi(t-1)/8$
3. $y_{t,i} = 1$ if $z'_{t,i}z''_{t,i} > 0$ else 0

Real Data. We conduct experiments on two real-world datasets that have been used as benchmarks in previous studies on concept drift.

NOAA[1]: This dataset contains approximately 50 years of meteorological data containing measurements of temperature, pressure, etc. on a daily basis and the data is shown to be cyclical in nature [7]. It has 18,159 samples and 8 features. The task is to predict whether it rained or not.

ONP[2]: This is a dataset about the news articles published on a digital media website for a period of 2 years. It consists of 39,797 samples and 61 features. The regression task is converted into a classification problem by setting a threshold on the number of shares of an article [9].

Preprocessing. For the study of temporal variation of the classification performance in our experiments, we utilize the data until a certain time unit for

[1] https://www.ncdc.noaa.gov/cdo-web/datasets.
[2] https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity.

Table 1. Average and standard deviation of the mean of AUC at all test time instances for five independent runs

Method	Multimodal	XOR	ONP2	ONP4	NOAA
RF-Batch	0.648 ± 0.003	0.464 ± 0.008	$\mathbf{0.615 \pm 0.008}$	$\mathbf{0.633 \pm 0.002}$	0.712 ± 0.003
RF-Present	0.722 ± 0.001	0.541 ± 0.004	0.579 ± 0.010	0.607 ± 0.006	0.690 ± 0.006
AAAI16	0.717 ± 0.004	0.516 ± 0.002	0.503 ± 0.011	0.524 ± 0.061	0.634 ± 0.049
AAAI17	0.710 ± 0.009	0.508 ± 0.009	0.569 ± 0.025	0.584 ± 0.006	0.772 ± 0.086
KDD18	0.732 ± 0.010	0.514 ± 0.010	0.568 ± 0.025	0.584 ± 0.009	–
Proposed	$\mathbf{0.990 \pm 0.003}$	$\mathbf{0.976 \pm 0.005}$	$\mathbf{0.616 \pm 0.025}$	$\mathbf{0.630 \pm 0.014}$	$\mathbf{0.828 \pm 0.034}$

training, and the remaining for testing. For Multimodal and XOR, we obtain 300 samples randomly per time instance, i.e. $N_t = 300$ for $t \in \{1, \ldots 30\}$. We split the data into train and test data by taking $T = 20$. Since the data is generated in a discretized manner, it can be directly used in the proposed and the other three competing methods (i.e. KDD18, AAAI17, and AAAI16). The real datasets that we use have sequential data. As a result, data within an interval of length ΔT is binned together and considered to have arrived at the same timestamp. The value of ΔT may have a significant effect on the performance of the four methods and a grid search approach should be used to determine its value. However, its optimum value may be different for each of the three methods. Thus, we do not tune this parameter to compare the performance over the same sequential data. Moreover, we randomly sample 80% of the data at each time instance and train the models using it. For NOAA, we discard the last 159 samples and set $\Delta T = 600$ days which splits the data into 30 units and keep $T = 20$. The remaining 10 units are used as testing data. For ONP, we use the same setting as [22] i.e. one month as ΔT and $T = 15$ while the remaining 10 units are used as testing data. ONP2 and ONP4 datasets are generated by further subsampling the data at training time instances by 20% and 40%, respectively.

Complexity of Datasets. We perform a short study on the classification complexity of the prepared datasets by training a linear Support Vector Machine (SVM) (as described in [25]). The mean m of the distances of incorrectly classified samples from the decision boundary is used to find a measure of linearity of the dataset as follows:

$$L = 1 - \frac{1}{1 + m}$$

As a result, we found the values of L for ONP2, ONP4, NOAA are respectively 0.37, 0.37, and 0.44, which shows the non-linearity of decision boundary in these datasets. Note that low values for L (bounded in $[0, 1)$) indicate that the problem is close to being linearly separable.

5.2 Results

Table 1 shows the average and standard deviation of the mean of AUCs over test time instances and Fig. 2 shows the average and standard deviation of AUCs for

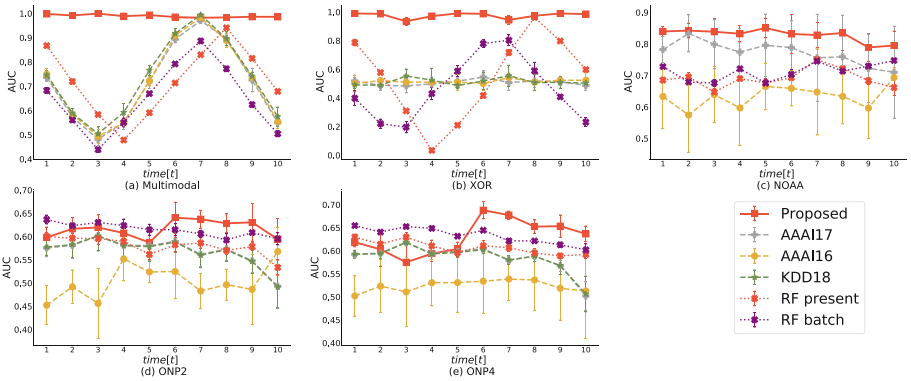

Fig. 2. Average and standard deviations of AUCs for all test time units over five runs

all test time instances. In synthetic data experiments, the proposed method consistently outperformed all other approaches by a significant margin. Competing approaches (i.e. KDD18, AAAI17, and AAAI16) that predict linear classifiers in the future performed poorly since the aggregated decision boundary was non-linear in the dataset. RF-Batch and RF-Present showed poor performances since the classifiers are not updated at test time instances in these approaches.

In Multimodal, the proposed method was able to identify the two modes and learn the dynamics separately for each mode to predict future classifiers, as validated by the AUC scores. KDD18, AAAI17, and AAAI16 did not predict classifiers based on the dynamics of the rotating boundary (corresponding to the left mode) and thus showed an oscillating AUC performance (see Fig. 2). RF-Batch and RF-Present showed a similar trend with RF-Present achieving the best AUC value at $t_{Test} = 8$, which corresponds to the period of rotation.

The nature of the XOR dataset exposes the limitations of linear approaches since any orientation of a linear classifier will separate the input space into two regions with (almost) an equal number of positive and negative samples. As a result, all three competing approaches had average AUC values near 0.5 i.e. as poor as a random guess (see Table 1). The proposed method showed robust performance even if the underlying covariate distribution is not a Gaussian mixture, and the decision boundary is non-linear. Figure 3 shows the predicted future classifiers by the proposed method for the XOR dataset in a single run. It can be observed from the figure that the proposed method identified the location of the mixture components in the input space such that the evolving non-linear decision boundary can be approximated well with the mixture of linear classifiers at all time-instances.

On real datasets, the proposed method achieved better average AUC values (see Table 1) than the competing linear approaches. In ONP2, the proposed method consistently showcased better AUC values at all the test time instances. In ONP4, the proposed method showcased similar or better performance than competing approaches at all time instances except at $t_{Test} = 3$. In ONP experiments, the results of AAAI16 were poor compared to AAAI17. One of the reasons for this drop is the difficulty in modeling the dynamics of the decision

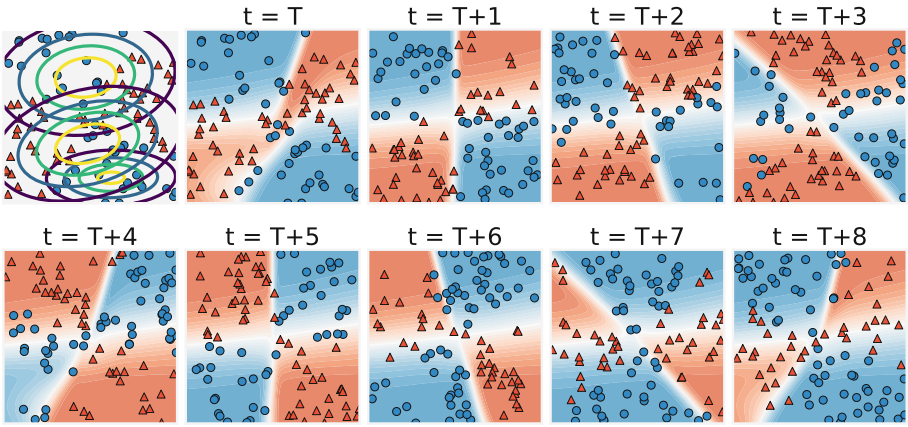

Fig. 3. For XOR dataset, contour plot of covariate distribution of the 3 inferred components followed by filled contour plots of (global) class probabilities at last training instance $t = T$ and future predictions for $t = T + 1, \ldots T + 8$. Red to blue scale represents class probabilities varying from 0 to 1. Ground truth positive (blue dots) and negative samples (red triangles) are shown too (Color figure online)

boundary using the AR process compared to GP. The proposed method uses the AR process as well for modeling the dynamics within each component. However, we believe it is not a limitation in our method since the mixture approach could help capture the complex dynamics. However, a more rigorous study on the hypothesis space of such evolutionary models is required, which is out of the scope of current work. For ONP experiments, the proposed method showed a similar result to the RF-Batch method and RF is considered as one of the best non-linear classification algorithms. At last, the proposed method was able to outperform all other approaches on the NOAA dataset. The result of KDD on the NOAA dataset is not presented since we faced convergence issues in the maximization step for mean and variances of classifier weights for the test time instances (refer [22]). The results validate that the proposed method can predict future classifiers when the decision boundary, possibly non-linear, is evolving and maintain it for some time.

6 Conclusion

We proposed a probabilistic model for predicting future non-linear classifiers given time-stamped labeled data collected until the current time. At each time instance, our model approximates the non-linear decision boundary using a mixture of linear classifiers. We used the Dirichlet process as prior in our model to automatically determine the number of required linear classifiers and developed a variational inference-based learning algorithm. Through synthetic and real data experiments we confirmed the effectiveness of the proposed method.

References

1. Abdallah, Z.S., Gaber, M.M., Srinivasan, B., Krishnaswamy, S.: Adaptive mobile activity recognition system with evolving data streams. Neurocomputing **150**, 304–317 (2015)
2. Alippi, C., Liu, D., Zhao, D., Bu, L.: Detecting and reacting to changes in sensing units: the active classifier case. IEEE Trans. Syst. Man Cybern. Syst. **44**(3), 353–362 (2013)
3. Angelov, P., Zhou, X.: Evolving fuzzy systems from data streams in real-time. In: 2006 International Symposium on Evolving Fuzzy Systems, pp. 29–35. IEEE (2006)
4. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Boston (2006). https://doi.org/10.1007/978-1-4615-7566-5
5. Brzezinski, D., Stefanowski, J.: Reacting to different types of concept drift: the accuracy updated ensemble algorithm. IEEE Trans. Neural Netw. Learn. Syst. **25**(1), 81–94 (2014)
6. Crammer, K., Even-Dar, E., Mansour, Y., Vaughan, J.W.: Regret minimization with concept drift. In: Proceedings of the 23rd Annual Conference on Learning Theory (COLT) (2010)
7. Ditzler, G.: Incremental learning of concept drift from imbalanced data (2011)
8. Ferguson, T.S.: A bayesian analysis of some nonparametric problems. The annals of statistics pp. 209–230 (1973)
9. Fernandes, K., Vinagre, P., Cortez, P.: A proactive intelligent decision support system for predicting the popularity of online news. In: Pereira, F., Machado, P., Costa, E., Cardoso, A. (eds.) EPIA 2015. LNCS (LNAI), vol. 9273, pp. 535–546. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23485-4_53
10. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. (CSUR) **46**(4), 44 (2014)
11. Griffiths, T.L., Jordan, M.I., Tenenbaum, J.B., Blei, D.M.: Hierarchical topic models and the nested Chinese restaurant process. In: Advances in Neural Information Processing Systems, pp. 17–24 (2004)
12. Hannah, L.A., Blei, D.M., Powell, W.B.: Dirichlet process mixtures of generalized linear models. J. Mach. Learn. Res. **12**, 1923–1953 (2011)
13. Haque, A., Khan, L., Baron, M.: Sand: semi-supervised adaptive novel class detection and classification over data stream. In: THIRTIETH AAAI Conference on Artificial Intelligence (2016)
14. Haque, A., Khan, L., Baron, M., Thuraisingham, B., Aggarwal, C.: Efficient handling of concept drift and concept evolution over stream data. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 481–492. IEEE (2016)
15. Hofer, V.: Adapting a classification rule to local and global shift when only unlabelled data are available. Eur. J. Oper. Res. **243**(1), 177–189 (2015)
16. Jaakkola, T.S., Jordan, M.I.: Bayesian parameter estimation via variational methods. Stat. Comput. **10**(1), 25–37 (2000)
17. Klinkenberg, R.: Learning drifting concepts: example selection vs. example weighting. Intell. Data Anal. **8**(3), 281–300 (2004)
18. Kolter, J.Z., Maloof, M.A.: Using additive expert ensembles to cope with concept drift. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 449–456. ACM (2005)
19. Koychev, I.: Gradual forgetting for adaptation to concept drift. In: Proceedings of ECAI 2000 Workshop on Current Issues in Spatio-Temporal ... (2000)

20. Kumagai, A., Iwata, T.: Learning future classifiers without additional data. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
21. Kumagai, A., Iwata, T.: Learning non-linear dynamics of decision boundaries for maintaining classification performance. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
22. Kumagai, A., Iwata, T.: Learning dynamics of decision boundaries without additional labeled data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1627–1636. ACM (2018)
23. Lee, K.H., Verma, N.: A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals. IEEE J. Solid-State Circuits **48**(7), 1625–1637 (2013)
24. Lindstrom, P., Delany, S.J., Mac Namee, B.: Handling concept drift in a text data stream constrained by high labelling cost. In: Twenty-Third International FLAIRS Conference (2010)
25. Lorena, A.C., Garcia, L.P., Lehmann, J., Souto, M.C., Ho, T.K.: How complex is your classification problem? A survey on measuring classification complexity. ACM Comput. Surv. (CSUR) **52**(5), 1–34 (2019)
26. Lütkepohl, H.: Vector Autoregressive Models. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-04898-2_609
27. Martínez-Rego, D., Pérez-Sánchez, B., Fontenla-Romero, O., Alonso-Betanzos, A.: A robust incremental learning method for non-stationary environments. Neurocomputing **74**(11), 1800–1808 (2011)
28. Sethuraman, J.: A constructive definition of Dirichlet priors. Statistica Sinica, 639–650 (1994)
29. Shahbaba, B., Neal, R.: Nonlinear models using Dirichlet process mixtures. J. Mach. Learn. Res. **10**, 1829–1850 (2009)
30. Škrjanc, I., Iglesias, J.A., Sanchis, A., Leite, D., Lughofer, E., Gomide, F.: Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey. Inf. Sci. **490**, 344–368 (2019)
31. Teh, Y.W., Jordan, M.I., Beal, M.J., Blei, D.M.: Sharing clusters among related groups: Hierarchical Dirichlet processes. In: Advances in Neural Information Processing Systems, pp. 1385–1392 (2005)
32. Wang, C., Paisley, J., Blei, D.: Online variational inference for the hierarchical Dirichlet process. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 752–760 (2011)
33. Wang, J., Zhao, P., Hoi, S.C.H.: Exact soft confidence-weighted learning. In: Proceedings of the 29th International Conference on Machine Learning, pp. 121–128 (2012)
34. Yang, X., Dinh, A., Chen, L.: Implementation of a wearerable real-time system for physical activity recognition based on Naive Bayes classifier. In: 2010 International Conference on Bioinformatics and Biomedical Technology, pp. 101–105. IEEE (2010)
35. Yuan, C., Neubauer, C.: Variational mixture of gaussian process experts. In: Advances in Neural Information Processing Systems, pp. 1897–1904 (2009)
36. Zhu, X., Zhang, P., Lin, X., Shi, Y.: Active learning from stream data using optimal weight classifier ensemble. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **40**(6), 1607–1621 (2010)
37. Žliobaitė, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drifting streaming data. IEEE Trans. Neural Netw. Learn. Syst. **25**(1), 27–39 (2013)