Model Averaging via Penalized Regression for Tracking Concept Drift

Kyupil Yeon, Moon Sup Song, Yongdai Kim,* Hosik Choi†
Cheolwoo Park‡

January 20, 2010

Abstract

A supervised learning algorithm aims to build a prediction model using training examples. This paradigm typically has the assumptions that the underlying distribution and the true input-output dependency does not change. However, these assumptions often fail to hold, especially in data streams. This phenomenon is known as concept drift.

We propose a new model combining algorithm for tracking concept drift in data streams. The final predictive ensemble model has a form of a weighted average and ridge regression combiner. The coefficients of the combiner are determined by ridge regression with the constraints such that the coefficients are nonnegative and sum to one. The proposed algorithm is devised via a new measure of concept drift, the angle between the estimated weights from data and the optimal weight vector obtained under no concept drift. It is shown that the ridge tuning parameter plays a crucial role of forcing the proposed algorithm to adapt to concept drift. Our main findings include (i) the proposed algorithm can achieve the optimal weights in the case of no concept drift if the tuning parameter is sufficiently large, and (ii) the angle is monotonically increasing as the tuning parameter decreases. These imply that if the tuning parameter is well-controlled, the algorithm can produce weights which reflect the degree of concept drift measured by the angle. Using various numerical examples, it is shown that the proposed algorithm can track concept drift better than other existing ensemble methods. Supplemental materials computer code and R-package are available online.

Keywords: Concept drift, Data stream, Model averaging, Penalized regression

1 Introduction

A huge amount of modern data sets are being generated every month, every day, or even every second. For example, credit card transaction data, retail sales records, or Internet traffic measure-

*Department of Statistics, Seoul National University, Seoul, 151-742, South Korea
†Department of Informational Statistics, Hoseo University, Asan, 330-713, South Korea
‡Corresponding author. Department of Statistics, University of Georgia, Athens, GA 30602-1952, U.S.A. Email: cpark@uga.edu

ments to name a few. Such sequential data streams make statistical modeling a challenge due to large volumes. Another challenge in data stream analysis comes from possible changes over time of the environment under which a certain data set is obtained. This is known as concept drift. The fundamental problem we are interested in this paper is, given a sequential data stream, how should a predictive model be constructed in order to accurately account for time-evolving concepts in the streams.

Let $\{D_j, j = 1, 2, \ldots\}$ be a sequence of data batches, each consisting of input-output pairs. Suppose that observations in D_j are random samples from unknown distributions $F_j(\mathbf{x}, y)$, where $\mathbf{x} \in \mathcal{X} \subset R^p$ is an input vector and $y \in \mathcal{Y}$ is a response. The objective of learning is to construct a predictive model for future instances based on the data sets $D_1, \ldots, D_m$ at each time point m. One problematic issue in this setting of sequential data stream is concept drift which states that the underlying target concept changes over time abruptly or gradually. In a probabilistic sense, it can be characterized as the change of underlying distributions such that $F_i(\mathbf{x}, y) \neq F_j(\mathbf{x}, y)$ for some $i \neq j$.

The concept drift phenomenon, which emerges naturally in a data stream, makes the learning process complex because the predictive model constructed on the past data is no longer consistent with the new examples. Therefore, in order to cope with concept drift, a learning algorithm should be equipped with a mechanism to adapt to the concept drift. For example, a learning algorithm should quickly detect concept drift, discriminate real concept drift from noise, cope with recurring concepts, and be simple and fast to deal with sequential data streams.

We propose an ensemble algorithm that aims to achieve these goals. The final predictive model produced by the proposed algorithm has a form of a weighted average and ridge regression combiner. It is a weighted average of base models that are independently constructed from each data batch. The model averaging process is done via penalized regression, which can track concept drift effectively.

Topics on concept drift have mostly been addressed in the machine learning literature as explained in Section 2, but they have also been dealt with in statistics. For example, Wegman & Marchette (2003) proposed recursive algorithms and evolutionary graphics focusing on Internet traffic data. Also, Hall et al. (2006) introduced a nonparametric time-dynamic kernel type density estimate that is capable of capturing changing trends when multivariate distribution evolves with

time. The proposed method in this paper aims to solve the problem as it naturally emerged from the machine learning community, but its statistical properties are shown here as well. In this way we attempt to bridge the gap between the two communities.

The organization of the paper is as follows. The next section reviews previous work on concept drift tracking algorithms. Section 3 describes the proposed algorithm with its motivation and properties. Their proofs are delayed to Appendix. In Section 4, we show our numerical examples. Finally, we conclude in Section 5.

2 Learning under Concept Drift

Concept drift tracking algorithms can roughly be divided into two categories. One is a single learner based tracker that aims to select previous examples or data batches most relevant to learning the current concept. We refer to it as *data combining approach* in this paper. The other is an ensemble approach mainly related to formulating and restructuring an ensemble of several base learners. In this approach, combining these base models is critical for dealing with various types of concept drift. Therefore, we refer to it as *model combining or ensemble approach*.

In a data combining approach, a conventional way of coping with the concept drift problem is to use a time window of fixed size over data streams. In other words, the most recent M data batches are used to construct a predictive model. However, there is a dilemma in this approach. A large size of the time window is preferable when there is no concept drift, but cannot adapt quickly to concept drift. On the contrary, a small size of the time window can track a new concept quickly, but is not preferable when the concept is stable or recurrent. Hence the optimal size of the time window cannot be set in general unless the type and degree of concept drift are known in advance. To adaptively determine the size of the time window, Widmer & Kubat (1996)'s FLORA systems use the WAH (Window Adjustment Heuristic) approach. Klinkenberg & Joachims (2000) proposed an algorithm for tracking concept drift with SVM (Support Vector Machines) in classification problems. A time window can be generalized to contain inconsecutive data batches selectively. This scheme is more relevant especially in the case of recurrent concepts. Klinkenberg (2004) suggested a local batch selection scheme by utilizing $\xi\alpha$-estimates for SVM (Joachims, 2000).

While a data combining approach is intuitive in the sense that one selects a subset of the past

data which is related to a new observation, it is not trivial to define "related" data batches. Also, retaining all (or parts) of the previous data sets is not efficient since it will eventually exhaust limited storage space.

A model combining approach is an ensemble strategy for learning in changing environments. In a static setting, ensemble methods such as bagging (Breiman, 1996), boosting (Freund & Schapire, 1997), and stacking (Wolpert, 1992) are generally known to produce a better prediction model than one single best model. In the wake of success in a static setting, ensemble methods have effectively been applied to a concept drift setting. Compared with data combining approaches, the ensemble method may be more suitable to data streams in that it does not need to retain all the previous data sets, rather just the base models. We take the model combining approach in this paper.

Street & Kim (2001) proposed the streaming ensemble algorithm that applies a simple unweighted voting to combining M base models. Therefore, the final ensemble model for predicting future instances is given by

$$\hat{f}_E(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} \hat{f}_j^{(m)}(\mathbf{x}),$$

where $E_m = \{\hat{f}_1^{(m)}(\cdot), \cdots, \hat{f}_M^{(m)}(\cdot)\}$ denotes M ensemble members formulated at the current time point m. Theorem 1 in Section 3.1 will show that this is the best model when there is no concept drift. It will be shown that our approach also possesses this property. Although the simple average can track a gradual concept drift by the ensemble updating procedure, it lacks fast adaptability when the concept drift occurs abruptly. In order to adapt quickly to abrupt concept changes, an algorithm should be minimally affected by obsolete base models.

Wang et al. (2003) suggested an accuracy-based weighted ensemble algorithm, that is, the ensemble prediction model is given by a weighted average of the ensemble members

$$\hat{f}_E(\mathbf{x}) = \sum_{j=1}^{M} w_j \hat{f}_j^{(m)}(\mathbf{x}),$$

with the constraints $\sum_{j=1}^{M} w_j = 1$ and $w_j \geq 0$ for all $j = 1, \ldots, M$. The weights $\{w_1, \ldots, w_M\}$ are determined so that they are proportional to each corresponding base model's accuracy on the most recent data batch D_m. The proposed model combining approach in this paper utilizes the same ensemble form but the weights are determined using penalized regression.

In contrast to the average type approach, an additional meta learning can be adopted as a combiner to aggregate the outputs of base models. Typically, classical regression methods are used as a combiner. Chu et al. (2004) used ordinary logistic regression as a combiner with an integrated outlier elimination procedure based on clustering of the likelihood values evaluated at each observation. By explicitly filtering out some noise-like observations in the most recent data batch and constructing an ensemble predictive model based on remaining examples, they developed an algorithm not only adaptive to concept drift but also robust to label noise.

Since the framework of a regression based ensemble method is relevant to our approach, we briefly introduce the procedure. First, one constructs base models $\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_m(\mathbf{x})$ independently from each $D_1, \ldots, D_m$. As a base learner one can use any type of learning algorithm. If one considers a classification problem, then each $\hat{f}_j(\mathbf{x})$ could be a confidence output such as class probability or class label itself. Second, one obtains a meta learning set from the outputs of the base models for the examples in the most recent data batch $D_m = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$. That is, for each $\mathbf{x}_i$ $(i = 1, \ldots, n)$, one obtains the outputs of base models denoted by

$$\hat{\mathbf{f}}(\mathbf{x}_i)^T = (\hat{f}_1(\mathbf{x}_i), \ldots, \hat{f}_{m-1}(\mathbf{x}_i), \hat{f}_m^{(-i)}(\mathbf{x}_i)),$$

where $f_m^{(-i)}(\mathbf{x}_i)$ represents a leave-one-out estimate of $f_m(\mathbf{x}_i)$. This is to avoid an over-fitting problem since observations in D_m are used for constructing both $\hat{f}_m$ and the meta learning set. Practically it would be better to use v-fold cross-validation (e.g. 5 or 10-fold CV) than leave-one-out. We will refer to this meta learning set as a *level-1 set* for reference to stacked generalization (Wolpert, 1992). In summary, the level-1 set for training a regression combiner is given as follows:

$$X = \begin{pmatrix} \hat{f}_1(\mathbf{x}_1) & \cdots & \hat{f}_{m-1}(\mathbf{x}_1) & \hat{f}_m^{(-1)}(\mathbf{x}_1) \\ \hat{f}_1(\mathbf{x}_2) & \cdots & \hat{f}_{m-1}(\mathbf{x}_2) & \hat{f}_m^{(-2)}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{f}_1(\mathbf{x}_n) & \cdots & \hat{f}_{m-1}(\mathbf{x}_n) & \hat{f}_m^{(-n)}(\mathbf{x}_n) \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} . \tag{1}$$

Finally, a final regression combiner model using (1) is given by

$$\hat{f}_E(\mathbf{x}) = \sum_{j=1}^{m} \hat{w}_j \hat{f}_j(\mathbf{x})$$

where $\hat{w}_j$'s are the estimated regression coefficients.

The regression-based combining described above regards base models as explanatory variables and fits a regression model to a level-1 set derived using examples in the most recent data set. This method does not require a time window over data batches or an ensemble size to be optimally set. The concept drift tracking is accomplished by a regression combiner's ability of allocating adaptive weights to base models. Moreover, it needs only to retain base models instead of entire data sets, which is a highly required desideratum of an algorithm for data stream mining.

In spite of many advantages, the regression-based combining method has several drawbacks. First, the so called multi-collinearity problem causes the estimation to be unstable especially in the case of no or gradual concept drift because base models are similar to each other. Second, the resulting weights may have negative values. The negative weights do not indicate the fact that the corresponding base models are negatively correlated with the target concept because the negative signs can occur if other highly correlated base models exist. Third, when there is no concept drift, the optimal weight should intuitively be the equal weight since each base model makes equal contributions. The regression-based combining method, however, cannot produce this optimal weight. We will return to this issue in the next section.

Another interesting ensemble method is based on the advice of several prediction strategies, called experts. Since different experts would perform best on different concepts, these expert prediction algorithms perform similarly as the best expert on each concept. Recently, Kolter & Maloof (2005) proposed Additive Expert which bounds its performance over changing concepts relative to the actual performance of an online learner trained on each concept individually. See Kolter & Maloof (2005) and the references therein for more details.

3 Model Averaging via Penalized Regression

In this section, we propose a new ensemble method which overcomes the shortcomings of average and regression-based models. Our method mimics the properties of the average type approach when a concept does not change or changes slowly, and adapts those of the regression-based approach when a concept changes quickly. Section 3.1 introduces the form of the proposed method, and its motivation and properties. In Section 3.2, we introduce the computational algorithm to implement the proposed method.

3.1 Motivation and Properties of the proposed method

The proposed method has the form of regression combiner with the constraints that weights are nonnegative and sum to one:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \quad \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m} w_j \hat{f}_j(\mathbf{x}_i) \right)^2 + \lambda \sum_{j=1}^{m} w_j^2, \tag{2}$$

$$\text{subject to} \qquad \sum_{j=1}^{m} w_j = 1, \; w_j \geq 0 \; .$$

It will be referred to as "MC.Ridge1+" indicating *M*odel *C*ombining via *Ridge* regression with constraints of sum-to-*1* and nonnegativity (+). The level-1 set for training penalized regression is given in (1).

At first glance, the proposed algorithm seems to simply mix the weighted average and ridge regression combiners in an attempt to solve the multi-collinearity problem and to enhance the interpretability. While this can be an intuitive motivation of the proposed method, there is another important motivation that is more directly related to concept drift tracking, which follows next.

Let us consider the case that there is no concept drift at all. That is, the target concept remains the same and the underlying distribution does not vary over time. In this case, the optimal aggregation in terms of the MSE criterion is just a simple average of base models if each base model is unbiased. Since this is an important property in our context, it is shown in Theorem 1 for the completeness of the paper. See Jacobs (1995) for the proof.

Theorem 1. *If each base model is unbiased and independent, then the optimal ensemble weight vector under no concept drift is given by* $\mathbf{w}^* = (1/m, \ldots, 1/m)$.

Theorem 1 indicates that all base models are equally important in the case of no concept drift. It motivates us to derive a new measure of concept drift. If an ensemble algorithm adapts to concept drift, then its estimated weights $\hat{\mathbf{w}}$ in the case of no concept drift would be close to the optimal $\mathbf{w}^*$. When a considerable concept drift occurs, the derived weights $\hat{\mathbf{w}}$ would be far from $\mathbf{w}^*$. Since we consider the space confined only on the hyperplane $\mathbf{w}^T \cdot \mathbf{1} = 1$ with $\mathbf{w} \geq 0$, the degree of departure of $\hat{\mathbf{w}}$ from $\mathbf{w}^*$ can be captured through the angle between the two m-dimensional vectors. Therefore, we propose this angle as a measure of concept drift.

Definition 1 (A measure of concept drift)**.** *For* $\mathbf{w}^* = (1/m, \ldots, 1/m)$ *and* $\hat{\mathbf{w}}$ *obtained by a combiner, we define the degree of concept drift as the angle between the two vectors given by*

$$\eta(\mathbf{w}^*, \hat{\mathbf{w}}) = \cos^{-1}\left(\frac{|\langle \mathbf{w}^*, \hat{\mathbf{w}} \rangle|}{||\mathbf{w}^*||\,||\hat{\mathbf{w}}||}\right), \tag{3}$$

where $\langle \cdot\,, \cdot \rangle$ *denotes the inner product of two vectors in* R^m *and* $||\cdot||$ *is the length of a vector defined as* $||\boldsymbol{a}|| = \langle \boldsymbol{a}, \boldsymbol{a} \rangle^{1/2}$ *for any* $\boldsymbol{a} \in R^m$*.*

Definition 1 is useful for designing a combiner for tracking concept drift. When there is no or quite gradual concept drift, we need an algorithm that is able to produce a weight vector close to $\mathbf{w}^*$. On the other hand, in the case of more considerable concept drift, a combiner should provide a weight $\hat{\mathbf{w}}$ which makes the angle $\eta(\mathbf{w}^*, \hat{\mathbf{w}})$ large. Note that the algorithms reviewed in Section 2 such as simple average, weighted average based on base models' accuracy, and regression combiners do not guarantee this property. In the following two theorems it will be shown that the proposed algorithm has such a property, which is its key advantage. These two theorems illustrate the reason why the proposed algorithm can be a reasonable concept drift tracker regardless of the types of changes.

Theorem 2. *In the case of no concept drift, the proposed algorithm* MC.Ridge1+ *in (2) produces a weight vector which converges to the optimal weight* $\mathbf{w}^*$ *as* $\lambda \to \infty$*.*

Theorem 3. *For the* $\hat{\mathbf{w}}$ *obtained by* MC.Ridge1+ *in (2),* $\eta(\mathbf{w}^*, \hat{\mathbf{w}})$ *in (3) is monotonically decreasing as* λ *increases.*

The proofs of Theorems 2 and 3 are delayed to the end of the paper. Theorem 2 indicates that the proposed algorithm can produce a simple average model in the case of no or gradual concept drift for sufficiently large λ. The estimation of λ is usually done by cross-validation which will turn out to be satisfactory in numerical analysis in Section 4.

The implication of Theorem 3 is more attractive. The importance of the ridge parameter λ in tracking concept drift is clearly conceived since λ controls the adaptivity of the algorithm to concept drift. The algorithm with larger λ generates a weight vector $\hat{\mathbf{w}}$ for which $\eta(\mathbf{w}^*, \hat{\mathbf{w}})$ gets smaller, which corresponds to no or gradual concept drift. On the other hand, smaller λ induces the algorithm to produce $\hat{\mathbf{w}}$ such that $\eta(\mathbf{w}^*, \hat{\mathbf{w}})$ gets larger, which corresponds to abrupt concept drift. In other words, the proposed algorithm can output a weight vector $\hat{\mathbf{w}}$ on which the degree

of concept drift is properly reflected, and this is controlled by λ. This property is simulated in Section 4.1.

3.2 Algorithm

We solve the problem described in (2) using the coordinatewise gradient descent (CGD) algorithm. Kim et al. (2008) proposed the gradient LASSO algorithm by employing the CGD method to provide an approximated solution for generalized LASSO models. The CGD algorithm is shown to be computationally much simpler and stabler than the Quadratic Programming (QP) algorithm, and easily applicable to high dimensional data.

Let us denote the objective function and the constraints as follows:

$$\begin{aligned} C(\mathbf{w}) &= \sum_{i=1}^{n} l(y_i, \mathbf{x}_i^T \mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}, \\ S &= \{\mathbf{w} : \mathbf{w}^T \mathbf{1} = 1,\ \mathbf{w} \geq 0\}, \end{aligned}$$

where $l(y, \mathbf{x}^T \mathbf{w})$ is a loss function that can be either L_2 or cross-entropy loss.

The main idea of the CGD algorithm is to find $\hat{\mathbf{w}}$ sequentially. For a given $\mathbf{v} \in S$ and $\alpha \in [0, 1]$, let $\mathbf{w}[\alpha, \mathbf{v}] = \mathbf{w} + \alpha(\mathbf{v} - \mathbf{w})$ and suppose that $\mathbf{w}$ is the current solution. Then, the CGD algorithm searches a direction vector $\mathbf{v}$ such that $C(\mathbf{w}[\alpha, \mathbf{v}])$ decreases most rapidly and updates $\mathbf{w}$ to $\mathbf{w}[\alpha, \mathbf{v}]$. Note that $\mathbf{w}[\alpha, \mathbf{v}]$ is still in S. The Taylor expansion implies

$$C(\mathbf{w}[\alpha, \mathbf{v}]) \approx C(\mathbf{w}) + \alpha \langle \nabla C(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle,$$

where $\nabla C(\mathbf{w}) = \left(\frac{\partial C(\mathbf{w})}{\partial w_1}, \cdots, \frac{\partial C(\mathbf{w})}{\partial w_m} \right)^T$. It can be easily shown that

$$\min_{\mathbf{v} \in S} \langle \nabla C(\mathbf{w}), \mathbf{v} \rangle = \min \left\{ \frac{\partial C(\mathbf{w})}{\partial w_j}, \cdots, \frac{\partial C(\mathbf{w})}{\partial w_m} \right\} .$$

Hence the desired direction is a j^*-th coordinate unit vector $\mathbf{u}_{j^*}$ such that

$$\begin{aligned} j^* &= \arg\min_{j \in \{1, \ldots, m\}} \left\{ \frac{\partial C(\mathbf{w})}{\partial w_j} \right\}, \\ \mathbf{u}_{j^*} &= (0, \ldots, 0, \underset{\substack{\uparrow \\ j^*-th}}{1}, 0, \ldots, 0) \ . \end{aligned}$$

The optimization procedure is summarized in Fig. 1. The convergence of the algorithm is guaranteed by Theorem 1 in Kim et al. (2008).

Let $C(\mathbf{w}) = (\mathbf{y} - X\mathbf{w})^T(\mathbf{y} - X\mathbf{w}) + \lambda \mathbf{w}^T\mathbf{w}$.

For $k = 1, 2, 3, \ldots,$

1. $\nabla C(\mathbf{w}^k) = \left(\frac{\partial C(\mathbf{w}^k)}{\partial w_1}, \cdots, \frac{\partial C(\mathbf{w}^k)}{\partial w_m}\right)^T$, $\mathbf{w}^1 = \mathbf{w}^*$: initial value.
2. $j^* = \underset{j \in \{1, \cdots, m\}}{\arg\min} \langle \nabla C(\mathbf{w}^k), \mathbf{u}_j \rangle$, $\mathbf{u}_j$: j-th unit vector.
3. $\hat{\alpha} = \underset{0 \le \alpha \le 1}{\arg\min}\ C\left((1-\alpha)\mathbf{w}^k + \alpha \mathbf{u}_{j^*}\right)$.
4. $\mathbf{w}^{k+1} = (1 - \hat{\alpha})\mathbf{w}^k + \hat{\alpha}\mathbf{u}_{j^*}$.

Iterate 1–4 until the solution converges.

Figure 1: Coordinatewise gradient descent method for MC.Ridge1+

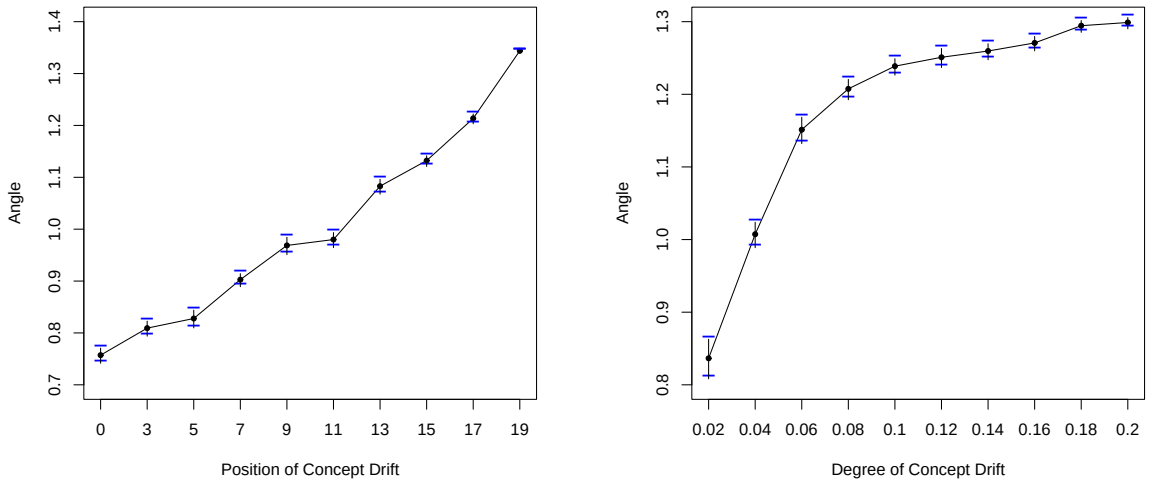
4 Numerical study

4.1 Angle Simulation

Theorem 3 in Section 3.1 implies that the proposed algorithm, with the appropriately estimated penalty parameter, produces weights so that the angle, defined in Definition 1, increases as concept drift occurs. In this subsection, this property is verified by estimating λ via the 10-fold cross validation (CV). We formulate two simulation settings.

Setting 1

We construct $m = 20$ data batches $D_1, \cdots, D_{20}$. Each D_j contains 200 observations generated independently from $y = \sum_{i=1}^{10} \alpha_i X_i + \epsilon$ where y is a response variable and $(X_1, \cdots, X_{10})$ are independent predictor variables from Uniform(0,1) distribution. The ϵ is a noise variable from $N(0, 1)$ independent of predictor variables. Concept drift is incorporated in the simulation by changing the value of $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10})$ before and after a certain time point Δ. For example, if $\Delta = 10$, each D_j for $j = 1, \cdots, 9$ is generated with $\boldsymbol{\alpha} = (5, 5, 5, 5, 5, 0, 0, 0, 0, 0)$ and the remaining data batches with $\boldsymbol{\alpha} = (0, 0, 0, 0, 0, 5, 5, 5, 5, 5)$. We consider ten time points $\Delta = 0, 3, 5, 7, 9, 11, 13, 15, 17, 19$. For each Δ, we measure the angle between the weight $\hat{\mathbf{w}}$ obtained by



(a) Angle vs. Position of concept drift (b) Angle vs. Degree of concept drift

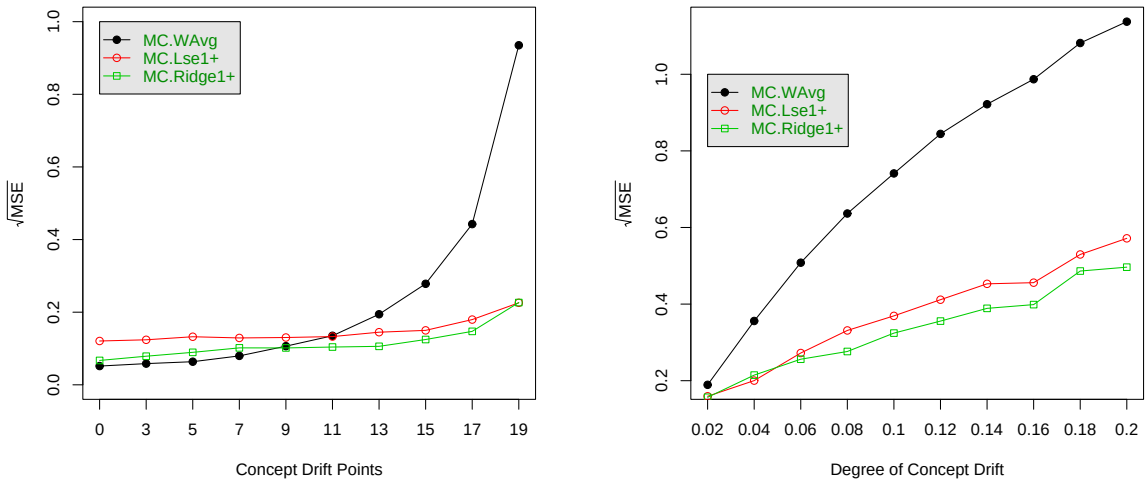
Figure 2: Change of angles according to concept drift.

the proposed algorithm and $\mathbf{w}^* = (1/20, \cdots, 1/20)$. This process is repeated 30 times and the mean of the angles is displayed in Figure 2(a) with the mark of one standard error.

It can be seen that the angle increases as concept drift occurs at a closer position to the current time point. It fits well with our expectation that the closer the change point is, the more abruptly the concept drift is regarded to occur to the current time point. For example, a change at $\Delta = 17$ has a more severe effect on the current time point than the change at $\Delta = 3$, and thus the angle measured at $\Delta = 17$ is much larger than the one at $\Delta = 3$. This result confirms automatic adaptiveness of the proposed algorithm to concept drift and the relevance of the CV procedure implemented for estimating the ridge penalty λ.

Setting 2

The second simulation setting is designed to identify adaptiveness of the proposed algorithm in the presence of gradual concept drift. We examine the magnitude of the angle according to the degree of concept drift. The general setting of the data generation is the same as the setting 1 except the incorporation of concept drift. We generate the first data batch D_1 with $\boldsymbol{\alpha}_1 = (5, 5, 5, 5, 5, 5, 5, 5, 5, 5)$ and then D_j with $\boldsymbol{\alpha}_j = \boldsymbol{\alpha}_{j-1} + \delta \cdot \mathbf{1}^*$ for $j = 2, \ldots, 20$, where $\mathbf{1}^* = (1, 1, 1, 1, 1, -1, -1, -1, -1, -1)$ and δ is a constant denoting an increment. Ten different



(a) RMSE vs. Position of concept drift (b) RMSE vs. Degree of concept drift

Figure 3: RMSE ($\sqrt{MSE}$) according to concept drift.

values of $\delta = 0.02, 0.04, \ldots, 0.2$ are used in the simulation. For each δ, we measure the angle between the weight $\hat{\mathbf{w}}$ produced by the proposed algorithm and $\mathbf{w}^* = (1/20, \cdots, 1/20)$. As δ increases, the degree of concept drift also increases. This procedure is repeated 30 times and the mean of the angles is displayed in Figure 2(b) with the mark of one standard error.

In Figure 2(b), the angle increases as the degree of concept drift becomes large. This result indicates that the estimated λ by 10-fold CV successfully reflects the amount of concept drift and hence the weights are well estimated in terms of the angle adaptively to concept drift.

For reference, we compare three algorithms, MC.WAvg (weighted average model), MC.Lse1+ (least squares), and MC.Ridge1+ (proposed). For the purpose of assessing the effect of the ridge penalty, we consider MC.Lse1+, which is formulated with constraints of sum-to-1 and nonnegativity (+) but no ridge penalty.

The average performances of the three algorithms for the two simulation settings are illustrated in Figure 3. We use a test data set consisting of 1000 examples that are generated from the same model as the last data batch. Figure 3(a) shows that if the concept drift occurs before the first half of the whole data batches, the weighted average works the best, but after the half it dramatically becomes the worst. This indicates that if the location of the concept drift is close

to the prediction point, the weighted average does not predict a new instance well. The two regression-based methods perform stably and the proposed method does slightly better. It is seen that the location of concept drift does not have a serious effect on the performance of regression combiners. Figure 3(b) shows that the degree of concept drift affects the weighted average method more severely compared to the regression combiners, and, again, the proposed method performs the best.

4.2 Simulated data

With two well-known two-class classification problems, we compare five model combining algorithms: model combining via simple average (MC.Avg), model combining via weighted average (MC.WAvg), model combining via least squares estimation with constraints (MC.Lse1+), model combining via ridge regression with constraints (MC.Ridge1+), and Dynamic selection (DS). The DS is a dynamic integration technique proposed by Tsymbal et al. (2008), which helps to better handle concept drift at an instance level. It selects the best base classifier in terms of a local predictive performance with regard to the instance tested. For reference, we add one more model, denoted as MC.1, which is constructed using only the most recent data batch.

4.2.1 Moving hyperplane data

We analyze a synthetic streaming data in which concept drift is simulated with a moving hyperplane. Hyperplanes have been used to simulate time-changing concepts by many authors (Hulten et al., 2001; Wang et al., 2003; Yang et al., 2005).

We consider a 10-dimensional hyperplane such as $\sum_{i=1}^{10} \alpha_i X_i = \alpha_0$. Examples are generated independently from Uniform $[0,1]^{10}$ and they are assigned a class label y as follows: $y = 1$ if $\sum_{i=1}^{10} \alpha_i X_i > \alpha_0$, and $y = 0$ otherwise. By setting $\alpha_0 = \frac{1}{2}\sum_{i=1}^{10} \alpha_i$, we make the hyperplane divide the unit hypercube into two parts of the same volume. Thus, the number of examples in each class is made to be balanced. We generate 30 data batches and each batch contains 100 examples. Concept drift between data batches is simulated by changing the magnitude of coefficients of the hyperplane. We initialize $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{10})$ with $(1, \ldots, 1)$. For each $\delta = 0.01,\ 0.02,\ 0.03$, gradual concept drift is incorporated by randomly adding or subtracting δ from each component of the one step previous $\boldsymbol{\alpha}$. Abrupt concept drift is also substantiated with $\delta = 1$ at a certain time

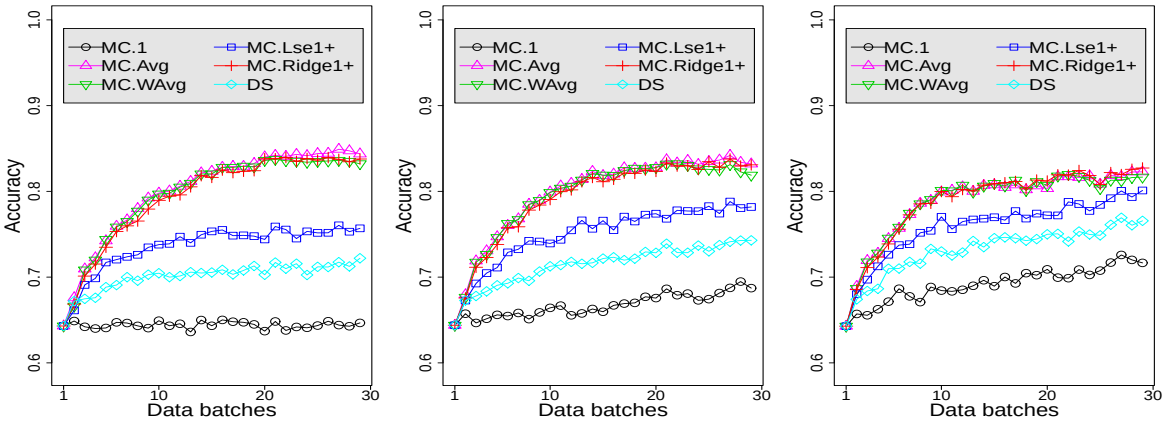
point. Furthermore, we observe the effect of noise by inserting 10% or 20% class label noise at each data batch. To compare the prediction accuracy of the algorithms we also construct test data sets. Each test set contains 1000 examples generated without noise from the same concept as the corresponding training data batch.

As base learners we use decision trees, support vector machines, and linear discriminant analysis. To save space we report the result only from decision trees, but other results are available in *Supplemental material*. We observe that the main lesson stays the same regardless of the type of base learners. Decision trees are constructed independently from each data batch utilizing a pruning procedure with 10-fold CV. Moreover, the probabilistic outputs of base models are combined by the combiners.

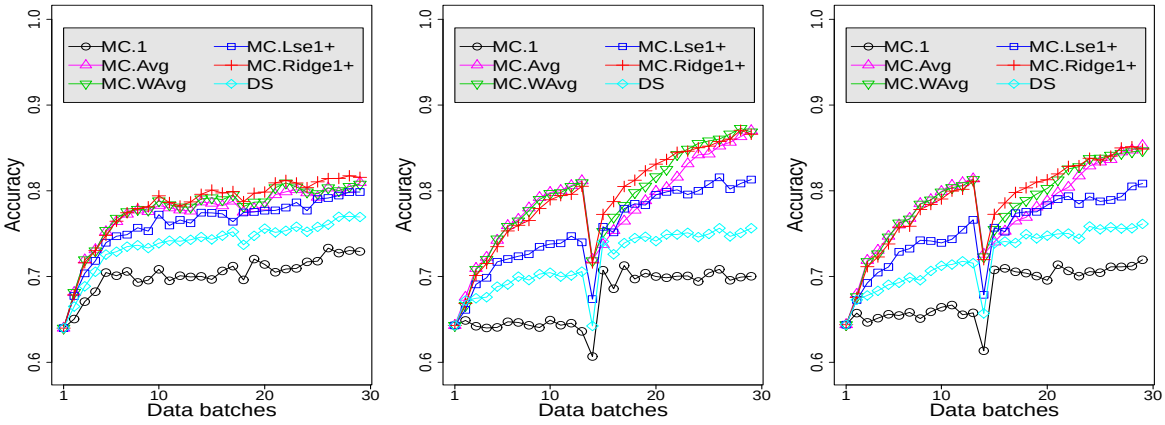
Figure 4 represents the results of the simulation when the ensemble size is 20 batches. In the plots, accuracy denotes the averaged value of 20 test accuracies obtained through 20 repeated simulations. In the case of no concept drift, the simple average is optimal as stated in Theorem 1 and this is demonstrated in Figure 4(a). The MC.Avg performs the best, and note that the proposed method MC.Ridge1+ shows almost identical accuracy. The MC.WAvg is also close to the best and the MC.Lse1+ follows next. Since the DS and MC.1 consistently perform poorly throughout the simulation study, we do not mention these two methods in the future discussion. The plot suggests that the average methods work well in the case of no concept drift and so does the proposed method since it achieves the optimal weight as stated in Theorem 2.

Figures 4(b)-4(d) show the case of gradual concept drift with increasing degrees from $\delta = 0.1$ to 0.3. One interesting observation is that if the degree is small (Figures 4(b) and 4(c)), the average-type slightly works better than the regression-based, but if the degree gets larger (Figure 4(d)), the regression-based starts to be more accurate. Note that the proposed method remains the best for all the three cases. When abrupt concept drift occurs (Figure 4(e)), the two regression methods (MC.Lse1+ and MC.Ridge1+) perform better than the average-type in the sense that they recover the accuracy more quickly after the drift occurs. The proposed method performs the best before (no drift) and after the drift. When we add a gradual drift to the abrupt change (Figure 4(f)), the result remains similar. Since the proposed method inherits good properties from average- and regression-type models, it can adapt well to various situations.

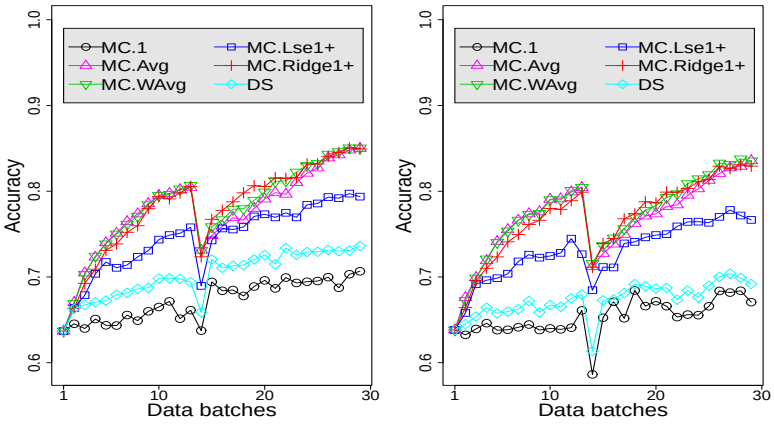
Figures 4(g) and 4(h) show the performance of the concept drift tracking methods when noise



(a) No concept drift (b) Gradual drift ($\delta = 0.1$) (c) Gradual drift ($\delta = 0.2$)



(d) Gradual drift ($\delta = 0.3$) (e) Abrupt drift (f) Abrupt & Gradual drift



(g) (f) & Noise 10% (h) (f) & Noise 20%

Figure 4: Concept drift tracking in Moving-hyperplane data.

is present. Since all the methods show little changes in terms of accuracy, they are robust to the class noise.

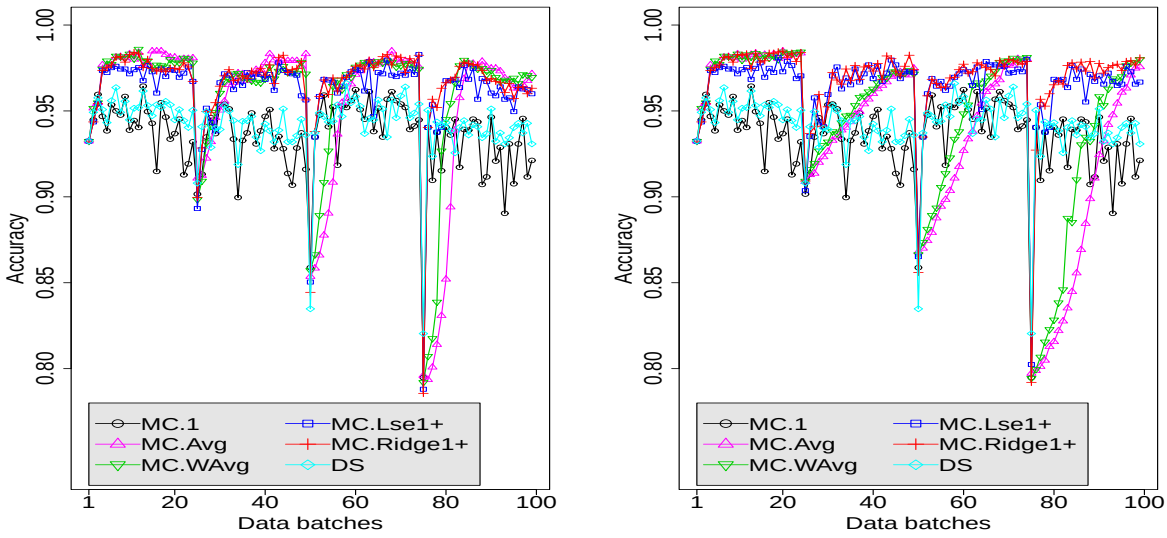
4.2.2 SEA data

This simulated data was analyzed in Street & Kim (2001). There are three predictor variables X_1, X_2, X_3 generated from Uniform [0,10] independently. The target variable y is a class label determined by the first two predictor variables X_1, X_2 such that $y = 1$ if $X_1 + X_2 \leq \theta$, and $y = 0$ otherwise. The θ can be regarded as an identifier of the target concept. To simulate concept drift we consider four concepts specified by $\theta = 8,\ 9,\ 7,\ 9.5$, respectively. We construct 25 data batches from each concept. That is, $D_1, \ldots, D_{25}$ are generated from concept 1 ($\theta = 8$), $D_{26}, \ldots, D_{50}$ from concept 2 ($\theta = 9$), $D_{51}, \ldots, D_{75}$ from concept 3 ($\theta = 7$), and $D_{76}, \ldots, D_{100}$ from concept 4 ($\theta = 9.5$). Each data batch consists of 500 examples. We insert approximately 10% class noise into each data batch. There are three concept changes occurring at $t = 25$, 50, and 75. At each time point $t = 1, \ldots, 99$, we construct a prediction model using $D_1, \ldots, D_t$ and then predict examples in a test set which is generated from the same concept as D_{t+1} without class noise. Each test data set contains 2500 examples. All the algorithms use the decision trees constructed independently using only each data batch and the probabilistic outputs of these trees are used by all the combiners. We use two ensemble sizes 10 and 25 to see the effect of different sizes on the accuracy.

Figure 5 confirms that the proposed algorithm MC.Ridge1+ recovers the prediction accuracy very quickly after sudden drops at the concept drift points and keeps its accuracy higher than those of the other algorithms, especially compared to the average type methods. It is worth mentioning that MC.Ridge1+ shows fast adaptivity regardless of the ensemble size (Figures 5(a) and (b)), but the average-type combiners are ineffective especially in the case of large ensemble size since the accuracy recovers slowly in Figure 5(b).

4.3 Real data

In this section we analyze three real examples using the same six models used in Section 4.2. We consider decision trees as the base learner because the examples contain many categorical variables.



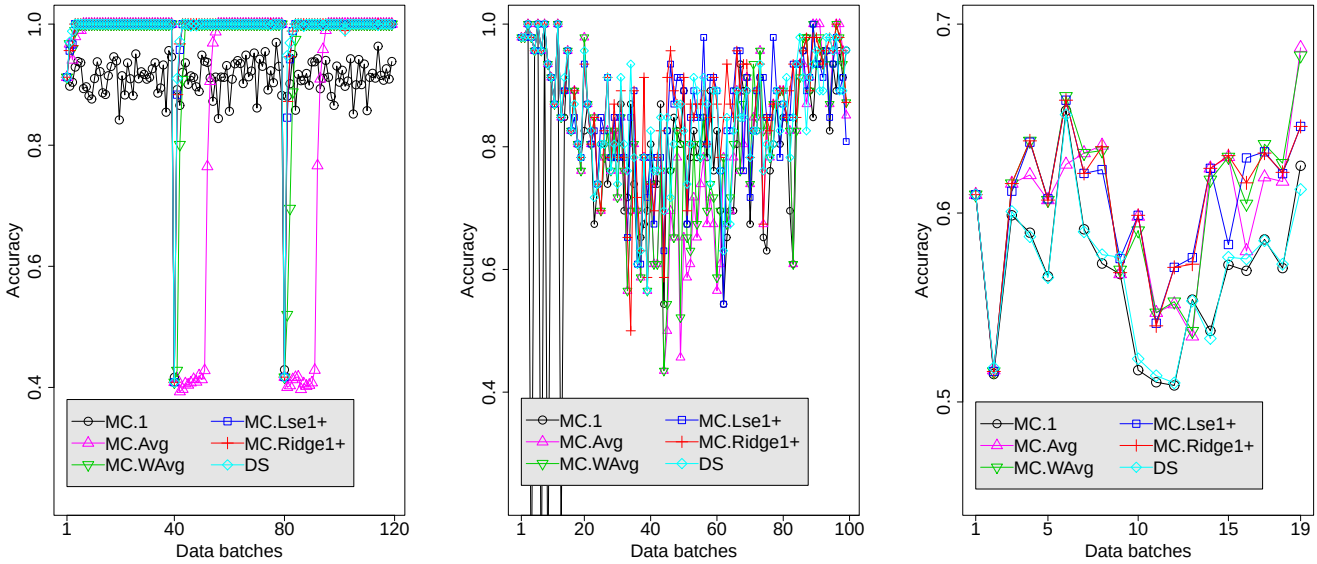
(a) Ensemble size = 10

(b) Ensemble size = 25

Figure 5: Concept drift tracking in SEA data.

STAGGER data set: The STAGGER is an incremental learning system developed by Schlimmer & Granger (1986). Since they tested the performance of STAGGER on the objects represented with three symbolic attributes, the stagger data set has been a standard benchmark data set for concept drift problems. The conceptual attributes are size (small, medium, large), color (red, blue, green), and shape (circle, square, triangle). We make up 120 sequential data batches with three different target concepts. The first 40 batches are based on the first target concept - (small $\wedge$ red). For the next 40 batches, the target concept is (green $\wedge$ circle). The last 40 data batches are from the third target concept - (medium $\vee$ large). Each data set consists of 100 examples.

Spam data set: The Spam data set from the UCI machine learning repository contains 4601 observations with 57 attributes (http://archive.ics.uci.edu/ml/). The objective of the learning with the data is classifying emails as Spam or Non-Spam. As Koychev (2004) conducted the experiment with the data in concept drift setting, we also simulate the changing context by sorting the data according to the *"capital_run_length_total"*, which is the total number of capital letters in the email. Then, the sorted data set is divided into 100 sequential data batches.



(a) STAGGER (b) Spam (c) Elec2

Figure 6: Real data analysis.

Elec2 data set: Elec2 data set was first described and analyzed in detail by Harries (1999). Gama et al. (2004) also used the data to experiment with their concept drift tracking algorithm. The data is based on the electricity market in the Australian state of New South Wales (NSW). It consists of 45,312 examples collected from 7 May 1996 to 5 December 1998. One example is obtained at every 30 minute. Therefore, 48 examples are observed in a time sequence every day. The target class label (UP/DOWN) is formulated depending on the current NSW electricity price is higher or lower than a moving average over the last 24 hours (or 48 instances). The predictor variables are *Day of week, Time stamp based on half hour periods, NSW electricity demand, Victorian electricity demand,* and *Scheduled electricity transfer between states.* In our experimental setting, we make up 20 data batches. Each data batch corresponds to one month and hence contains about 1,440 examples. In addition, only the examples since after May 1997 are used in the experiment because the previous examples have missing values in some variables. As a result, we have 20 batches in total.

In the STAGGER data (Figure 6(a)) the two regression methods, MC.Ridge1+ and MC.Lse1+, perform the best due to the abrupt drifts. The DS method works as well as the regression method in this example. The MC.WAvg closely follows next with a slight delay of the accuracy recovery

right after the abrupt changes. The MC.Avg slowly recovers the accuracy after the two sudden changes as expected. For the Spam data (Figure 6(b)) the methods show gradual changes in terms of their accuracy, which reflects gradual drifts rather than abrupt changes in the streaming data. For this reason it is not easy to pick the dominant performer, but the overall performance of the proposed method is superior especially in the middle batches. The Elec2 data (Figure 6(c)) shows gradual drifts again, and the regression and weighted average methods perform similarly well. It is demonstrated that the proposed method shows the highest accuracy in most batches.

5 Conclusion

In this paper, we proposed a new model combining method for tracking concept drift in data streams. The motivation of the proposed algorithm stems from a newly defined measure of concept drift. We showed that the new measure of concept drift, defined as an angle between two weight vectors, can be good guidance for designing an ensemble tracker of concept drift. The proposed algorithm addresses the concept drift problem by constructing an ensemble prediction model, which can be viewed as both a weighted average and regression of base models. The core of the algorithm is how to determine the combining weights, which are estimated by ridge regression with the constraints such that weights are nonnegative and sum to one. It was shown with various numerical examples that the proposed algorithm can track concept drift better than other ensemble methods under both abrupt and gradual changes. The proposed method does not need to tune an optimal ensemble size, and does not depend on an intuitive restructuring of the current ensemble in order to adapt to concept drift. In addition, it has only to retain all base models instead of all the previous data batches within the maximal ensemble size.

One may try different penalties, for example L_q with $q > 0$. This penalty has been considered in various contexts, for example see Liu et al. (2007). Advantages of this approach is that it has the same properties as the proposed method when $q = 2$, and it can automatically select (i.e., exactly 0 weights for some base models) relevant base models when $q \leq 1$. We suggest this as future work.

Acknowledgement

The third author's work was supported by the Korea Research Foundation Grant funded by the Korean Government (KRF-2008-314-C00046).

Supplemental Materials

Appendix: We provide proofs of Theorems 2 and 3. We also report the results for support vector machines (SVM) and linear discriminant analysis (LDA). (appendix.pdf)

R-package for the proposed algorithm: R-package contains the code to perform the methods described in the article. (drift_0.1.zip)

References

Breiman, L. (1996). Bagging predictors. *Machine Learning* **24**, 123–140.

Chu, F., Wang, Y. & Zaniolo, C. (2004). Mining noisy data streams via a discriminative model. In *Discovery Science*, pages 47–59.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Science* **55**, 119–139.

Gama, J., Medas, P., Castillo, G. & Rodrigues, P. (2004). Learning with drift detection. In *Advances in Artificial Intelligence-SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence*.

Hall, P., Müller, H.-G. & Wu, P.-S. (2006). Real-time density and mode estimation with application to time-dynamic mode tracking. *Journal of Computational and Graphical Statistics* **15**, 82–100.

Harries, M. (1999). Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales.

Hulten, G., Spencer, L. & Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106, San Francisco, CA. ACM Press.

Jacobs, R. A. (1995). Methods for combining experts' probability assessments. *Neural Computation* **7**, 867–888.

Joachims, T. (2000). Estimating the generalization performance of a SVM efficiently. In Langley, P., editor, *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 431–438, Stanford, US. Morgan Kaufmann Publishers, San Francisco, US.

Kim, J., Kim, Y. & Kim, Y. (2008). A gradient-based optimization algorithm for lasso. *Journal of Computational and Graphical Statistics* **17**, 994–1009.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis (IDA), Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift* **8**.

Klinkenberg, R. & Joachims, T. (2000). Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 487–494. Morgan Kaufmann.

Kolter, J. Z. & Maloof, M. A. (2005). Using additive expert ensembles to cope with concept drift. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 449–456, New York, NY. ACM Press.

Koychev, I. (2004). Experiments with two approaches for tracking drifting concepts. *Serdica Journal of Computing* **1**, 27–44.

Liu, Y., Zhang, H., Park, C. & Ahn, J. (2007). Support vector machines with adaptive l_q penalty. *Computational Statistics and Data Analysis* **51**, 6380–6394.

Schlimmer, J. C. & Granger, R. H. (1986). Incremental learning from noisy data. *Machine Learning* **1**, 317–354.

Street, W. N. & Kim, Y. S. (2001). A streaming ensemble algorithm (sea) for large-scale classification. In *7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382. ACM Press.

Tsymbal, A., Pechenizkiy, M., Cunningham, P. & Puuronen, S. (2008). Dynamic integration of classifiers for handling concept drift. *Information Fusion* **9**, 56–68.

Wang, H., Fan, W., Yu, P. S. & Han, J. (2003). Mining concept drifting data streams using ensemble classifiers. In *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–235. ACM Press.

Wegman, E. J. & Marchette, D. J. (2003). On some techniques for streaming data: a case study of internet packet headers. *Journal of Computational and Graphical Statistics* **12**, 893–914.

Widmer, G. & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning* **23**, 69–101.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks* **5**, 241–259.

Yang, Y., Wu, X. & Zhu, X. (2005). Proactive-reactive prediction for data streams. Technical Report CS-05-03, Computer Science, University of Vermont.