Bootstrapped Control Limits for Score-Based Concept Drift Control Charts

Jiezhong Wu[1]

Daniel W. Apley[1]

[1]Department of Industrial Engineering and Management Sciences, Northwestern University

Abstract

Monitoring for changes in a predictive relationship represented by a fitted supervised learning model (aka concept drift detection) is a widespread problem, e.g., for retrospective analysis to determine whether the predictive relationship was stable over the training data, for prospective analysis to determine when it is time to update the predictive model, for quality control of processes whose behavior can be characterized by a predictive relationship, etc. A general and powerful Fisher score-based concept drift approach has recently been proposed, in which concept drift detection reduces to detecting changes in the mean of the model's score vector using a multivariate exponentially weighted moving average (MEWMA). To implement the approach, the initial data must be split into two subsets. The first subset serves as the training sample to which the model is fit, and the second subset serves as an out-of-sample test set from which the MEWMA control limit (CL) is determined. In this paper we develop a novel bootstrap procedure for computing the CL. Our bootstrap CL provides much more accurate control of false-alarm rate, especially when the sample size and/or false-alarm

rate is small. It also allows the entire initial sample to be used for training, resulting in a more accurate fitted supervised learning model. We show that a standard nested bootstrap (inner loop accounting for future data variability and outer loop accounting for training sample variability) substantially underestimates variability and develop a 632-like correction that appropriately accounts for this. We demonstrate the advantages with numerical examples.

Keywords: Control charts, Concept drift, Bootstrap, Predictive modeling, Machine learning.

1 Introduction

The increasing reliance on data-driven decision making has led to the widespread adoption of supervised learning models across various domains. These models aim to capture the predictive relationship $\mathbb{P}(Y|\mathbf{X})$ between a response variable Y and covariates $\mathbf{X}$. However, a fundamental challenge arises when the predictive relationship in new data deviates from that used to train the model, potentially rendering the model's predictions unreliable or obsolete [10, 18, 20], and/or reflecting a change in process behavior that should be detected. This challenge is particularly acute in domains like finance [16] and healthcare analytics [14], as most AI systems and algorithms require training data that may contain inherent biases or may not remain representative of the broader population over time [4].

Statistical process monitoring (SPM) and control charts have long served as foundational tools for detecting changes in process characteristics over time [11]. The field has evolved considerably, with recent advances incorporating machine learning methods like kernel methods [2] and neural networks [13] to enhance monitoring capabilities. As datasets have grown larger and organizations increasingly rely on predictive models, the predictive relationship between variables has emerged as a critical characteristic that requires monitoring for quality control purposes, alongside traditional process variables monitoring.

This evolution in the goals of monitoring reflects a fundamental shift in how data are

used in practice and/or representing a change in process behavior that should be detected. While traditional SPM methods are designed to detect shifts in the distribution of process variables, modern applications often require understanding changes in the predictive relationships themselves. This phenomenon, known as concept drift in the machine learning literature, occurs when the relationship between input features $\mathbf{X}$ and the target variable Y evolves over time [18], potentially degrading model performance [12, 18, 21] and/or representing a change in process behavior that should be detected. These changes can manifest gradually or abruptly, and may not affect predictive accuracy immediately [8], making them difficult to detect using conventional methods.

Most existing concept drift detection methods fall into two categories, with each having significant limitations. The first category consists of error-based approaches that rely primarily on monitoring classification error rates or prediction accuracy metrics [3, 15]. While straightforward to implement, these methods often fail to detect concept drift when changes in $\mathbb{P}(Y|\mathbf{X})$ do not manifest as increased error rates, such as when decision boundaries shift in ways that maintain similar overall accuracy despite fundamental changes in the underlying relationship. The second category comprises adaptive learning algorithms, which continually retrain or update models to adapt to incoming data streams [8, 9, 17]. While these approaches can improve responsiveness, the adaptive model retraining also requires significant computation and may overfit transient changes.

Related to, but distinctly different from concept drift detection, profile monitoring methods in statistical process control have been extensively developed to monitor functional relationships between response and predictor variables [19]. These methods typically involve monitoring regression parameters or fitted curves over time to detect changes in the relationship between a response variable and one or more explanatory variables [1, 6]. While this may appear similar to concept drift detection, profile monitoring fundamentally differs in its objectives and approaches and in the structure of the data to which it applies. In profile monitoring, data are grouped as batches of $(\mathbf{X}, Y)$ observations, where some feature

of $\mathbb{P}(Y|\mathbf{X})$ (typically $\mathbb{E}[Y|\mathbf{X}]$) for each batch represents a profile associated with the batch, and the objective is to monitor for changes in the nature of the profiles from batch to batch. This requires fitting predictive models separately to each batch of data. In contrast, in concept drift monitoring, data are individual ($\mathbf{X}$, Y) observations, and the objective is to detect whether the predictive relationship for new observations changes relative to what it was when some baseline model was fit to a prior set of training observations. Changes in the predictive relationship are detected by comparing a new stream of individual ($\mathbf{X}$, Y) observations to the baseline model, which does not involve fitting separate models to the new data.

Recently, [20] introduced a new Fisher score-based concept drift detection that uses well-established statistical theory to show that detecting changes in $\mathbb{P}(Y|\mathbf{X})$ is equivalent to detecting changes in the mean of the score vector (the gradient of the log-likelihood) of the supervised learning model, a more familiar problem in SPM for which a conventional multivariate exponentially weighted moving average (MEWMA) control charts can be used. Their approach demonstrates superior detection power compared to traditional error-based concept drift methods, while also providing valuable diagnostic information to help understand the nature of detected changes. It also avoids having to continuously retrain the model, as in adaptive learning algorithms. Computing the MEWMA control limit (CL) in [20] requires that a second large sample of ($\mathbf{X}$, Y) observations be collected, in addition to the training sample to which the baseline model is fit. This can be achieved by splitting the training sample into two sets (with the baseline model fit to the first set, and the second set used to compute the CL), but this reduces the size of the data to which the baseline model is fit and generally requires very large sample sizes to accurately control false-alarm rate.

Our main contribution is that we present a more data-efficient and reliable approach that uses bootstrapping to compute the CL from the same training sample to which the baseline model is fit, thereby allowing the entire sample to be used to train a more accurate baseline model, as well as providing much more accurate control of false-alarm rate. For

this we develop a nested bootstrap procedure with the inner loop accounting for future data variability and outer loop accounting for training sample variability. This is challenging because a naïve nested bootstrap procedure substantially underestimates variability of the MEWMA monitoring statistic for reasons that we discuss later. To account for this, we develop a 632-like correction that controls false-alarm rate much more accurately than the two-sample CL calculation of [20], especially when the training sample size and/or the desired false-alarm rate is small. Our implementation takes advantage of the inherent parallelism in the nested bootstrap structure, making the method computationally efficient and practical for modern applications involving complex models like deep neural networks.

The remainder of the paper is organized as follows. Section 2 reviews the score-based concept drift detection approach that is the basis of our approach. Section 3 develops a nested bootstrap approach and derives the associated variance-inflation correction that leads to appropriate CL. Section 4 demonstrates our method's effectiveness through a variety of examples ranging from linear mixture models to complex nonlinear dynamical systems. Section 5 concludes with a summary of the main findings in this paper.

2 Background on Score-Based Concept Drift Detection

This work considers a parametric supervised learning model $g(\boldsymbol{\theta}; \mathbf{X})$ to represent the conditional distribution $\mathbb{P}(Y \mid \mathbf{X}; \boldsymbol{\theta})$ of a target Y given inputs $\mathbf{X} \in \mathbb{R}^p$, where $\boldsymbol{\theta}$ denotes the model parameters. In classification settings, $g(\boldsymbol{\theta}; \mathbf{X})$ produces class probabilities, whereas in regression with Gaussian errors and squared-error loss, $g(\boldsymbol{\theta}; \mathbf{X})$ corresponds to the conditional mean $\mathbb{E}[Y \mid \mathbf{X}; \boldsymbol{\theta}]$ of the Gaussian distribution $\mathbb{P}(Y \mid \mathbf{X}; \boldsymbol{\theta})$. Following [20], we estimate $g(\boldsymbol{\theta}; \mathbf{X})$ via (possibly regularized) maximum likelihood using training data $\{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$, assumed to be i.i.d. from $\mathbb{P}(Y, \mathbf{X})$.

Denote the training data to which $g(\boldsymbol{\theta}; \mathbf{X})$ is fit by $\{(\mathbf{x}_i, y_i) : i = 1, 2, \cdots, n\}$, assumed

to be an i.i.d. sample drawn from the joint distribution $\mathbb{P}(Y, \mathbf{X})$. For each observation $(\mathbf{x}_i, y_i)$, the (Fisher) score vector is defined as $\mathbf{s}(\boldsymbol{\theta}; \mathbf{x}_i, y_i) = \nabla_{\boldsymbol{\theta}} \log \mathbb{P}(y_i|\mathbf{x}_i; \boldsymbol{\theta})$. Under certain regularity conditions, if the assumed parametric model is indeed the true data-generating mechanism with true parameters denoted by $\boldsymbol{\theta}^0$, a fundamental result in statistical theory [5][Proposition 3.4.4] states that the expected value of the score vector, when evaluated at $\boldsymbol{\theta}^0$, is equal to zero

$$\mathbb{E}_{\boldsymbol{\theta}^0}\left[\mathbf{s}\left(\boldsymbol{\theta}^0; \mathbf{X}, Y\right)\middle|\mathbf{X}\right] = \int \mathbf{s}\left(\boldsymbol{\theta}^0; \mathbf{X}, Y = y\right) \mathbb{P}\left(Y = y\middle|\mathbf{X}; \boldsymbol{\theta}^0\right) \mathrm{d}y = \mathbf{0}. \tag{2.1}$$

In the context of machine learning, the notion of concept drift refers to the phenomenon where the underlying relationship between $\mathbf{X}$ and Y evolves over time, which can be characterized as a shift in $\mathbb{P}(Y|\mathbf{X}; \boldsymbol{\theta})$. In the parametric setting of [20], concept drift translates to a change in $\boldsymbol{\theta}$. Under certain identifiability conditions, when the parameters change (to some $\boldsymbol{\theta} \neq \boldsymbol{\theta}^0$), the score vector mean $\mathbb{E}_{\theta}[\mathbf{s}(\boldsymbol{\theta}^0; \mathbf{X}, Y)|\mathbf{X}] = \int \mathbf{s}(\boldsymbol{\theta}^0; \mathbf{X}, Y = y)\mathbb{P}(Y = y|\mathbf{X}; \boldsymbol{\theta})\mathrm{d}y$ will differ from zero. In light of this, the approach of [20] converts concept drift monitoring to the equivalent problem of monitoring for a change in the mean of the score vector $\mathbf{s}(\hat{\boldsymbol{\theta}}; \mathbf{X}, Y)$, where $\hat{\boldsymbol{\theta}}$ denotes the MLE of $\boldsymbol{\theta}^0$ obtained by fitting the model $g(\boldsymbol{\theta}; \mathbf{X})$ to the training data. The MLE is given by

$$\hat{\boldsymbol{\theta}} := \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \frac{1}{n} \sum_{i=1}^{n} \log \mathbb{P}\left(y_i|\mathbf{x}_i; \boldsymbol{\theta}\right), \tag{2.2}$$

in which case

$$\nabla_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \log \mathbb{P}\left(y_i|\mathbf{x}_i; \boldsymbol{\theta}\right)|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{s}\left(\hat{\boldsymbol{\theta}}; \mathbf{x}_i, y_i\right) = \mathbf{0}, \tag{2.3}$$

which is the empirical counterpart of (2.1), since it represents the average score vector over the training data.

The empirical counterpart (2.3) also provides some justification for why the score-based concept drift monitoring approach is effective even when the structure of $\mathbb{P}(Y = y|\mathbf{X}; \boldsymbol{\theta})$ induced by the parametric supervised learning model $g(\boldsymbol{\theta}; \mathbf{X})$ does not perfectly match the true structure of $\mathbb{P}(Y|\mathbf{X})$. Suppose the predictive relationship between Y and $\mathbf{X}$ for a new set

of data $\{(\mathbf{x}_{n+i}, y_{n+i}) : i = 1, 2, \cdots, m\}$ differs substantially from what it was over the training data. The value of $\boldsymbol{\theta}$ that minimizes the log-likelihood over the new data will generally differ from the MLE $\hat{\boldsymbol{\theta}}$ over the training data, in which case

$$\nabla_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^{m} \log \mathbb{P}\left(y_{n+i} | \mathbf{x}_{n+i}; \boldsymbol{\theta}\right) |_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{s}\left(\hat{\boldsymbol{\theta}}; \mathbf{x}_{n+i}, y_{n+i}\right). \tag{2.4}$$

will differ from zero. The more the predictive relationship for the new data changes relative to the training data, the more we would expect the mean of the new score vectors $\{\mathbf{s}(\hat{\boldsymbol{\theta}}; \mathbf{x}_{n+i}, y_{n+i}) : i = 1, 2, \cdots, m\}$ evaluated at the same training $\hat{\boldsymbol{\theta}}$ to differ from zero. The same arguments hold if one uses a regularized version of MLE with the score vectors replaced by the derivative of the regularized log-likelihood, which is common in practice: When the predictive relationship changes from the training data, the derivative of the regularized log-likelihood for the new data will generally differ from zero when evaluated at the parameter values that minimize the regularized training log-likelihood.

This was the basis for the concept drift monitoring approach of [20], who used a standard multivariate EWMA (MEWMA) to monitor for changes in the mean of the score vector as the new data are collected. To determine the CL for the MEWMA, they divide the training data $\mathcal{D} = \{(\mathbf{x}_i, y_i) : i = 1, 2, \cdots, n\}$ into two subsets: $\mathcal{D}_1 := \{(\mathbf{x}_i, y_i) : i = 1, 2, \cdots, n_1\}$ and $\mathcal{D}_2 := \{(\mathbf{x}_i, y_i) : i = n_1+1, n_1+2, \cdots, n\}$. The first subset, $\mathcal{D}_1$, is used to fit the supervised learning model $g(\hat{\boldsymbol{\theta}}; \mathbf{x})$, producing the MLE $\hat{\boldsymbol{\theta}}$. They then compute the score vectors $\mathbf{s}(\hat{\boldsymbol{\theta}}; \mathbf{x}_i, y_i)$ over the second subset, $\mathcal{D}_2$, and apply an MEWMA to these score vectors to empirically compute the CL. Specifically, the CL is taken to be the $1-\alpha$ (α is the desired false-alarm rate) sample quantile of the Hotelling T^2 statistics, $\{(\mathbf{z}_i - \overline{\mathbf{s}})^\top \widehat{\boldsymbol{\Sigma}}^{-1} (\mathbf{z}_i - \overline{\mathbf{s}}) : i = n_1 + 1, n_1 + 2, \cdots, n\}$, where $\overline{\mathbf{s}} = \sum_{i=n_1+1}^{n} \mathbf{s}(\hat{\boldsymbol{\theta}}; \mathbf{x}_i, y_i)/(n - n_1)$ and $\widehat{\boldsymbol{\Sigma}} = \sum_{i=1}^{n_1} (\mathbf{s}(\hat{\boldsymbol{\theta}}; \mathbf{x}_i, y_i) - \overline{\mathbf{s}})(\mathbf{s}(\hat{\boldsymbol{\theta}}; \mathbf{x}_i, y_i) - \overline{\mathbf{s}})^\top / n_1$ are the mean vector and covariance matrix of the score vectors for the second subset, and the MEWMA $\mathbf{z}_i$ is defined recursively for $i = n_1 + 1, n_1 + 2, \cdots$ via $\mathbf{z}_i = \lambda \mathbf{s}_i + (1 - \lambda)\mathbf{z}_{i-1}$.

The method of [20] was primarily intended for situations in which n is very large, since the

size of $\mathcal{D}_2$ must be quite large to accurately determine the T^2 CL using the above procedure. Our nested bootstrap procedure (described in Section 3) to compute the T^2 CL results in much more accurate false-alarm rate control. This is especially true when n is not sufficiently large and/or the desired false-alarm rate is small, because the method of [20] requires a very large $\mathcal{D}_2$ for small false-alarm rates (e.g., for a desired false-alarm rate of 0.001, the size of $\mathcal{D}_2$ must be much larger than 1,000). Moreover, our procedure allows the CL to be computed from the same data to which the supervised learning model is fit. This removes the need to divide the training sample into two subsets and allows the entire sample $\mathcal{D}$ to be used for model fitting, which results in a more accurate model.

3 Nested Bootstrap Procedure for Computing the CL

Suppose the training sample $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is drawn i.i.d. from some joint distribution $\mathbb{P}_0(Y, \mathbf{X})$ for which the conditional distribution $\mathbb{P}(Y|\mathbf{X}; \boldsymbol{\theta}^0)$ can be implicitly represented by the parametric family $g_\gamma(\boldsymbol{\theta}; \mathbf{x})$ of supervised learning models. Let $\mathbb{E}_0[\cdot]$ denote the expectation operator with respect to $\mathbb{P}_0(Y, \mathbf{X})$.

We apply the supervised learning model to the training dataset $\mathcal{D}$ and compute the score vectors. For notational simplicity, we denote these score vectors as $\mathcal{S} = \{\mathbf{s}_i : i = 1, 2, \cdots, n\}$ instead of using the functional notation $\mathbf{s}(\hat{\boldsymbol{\theta}}; \mathbf{x}_i, y_i)$ from previous sections. Recall, each $\mathbf{s}_i$ is the derivative of the component of the model fitting objective function (the negative log-likelihood or a regularized version, with no optimization constraints) associated with observation $(\mathbf{x}_i, y_i)$. Let $\overline{\mathbf{s}} = 1/n \sum_{i=1}^n \mathbf{s}_i$ and $\widehat{\boldsymbol{\Sigma}} = 1/n \sum_{i=1}^n (\mathbf{s}_i - \overline{\mathbf{s}})(\mathbf{s}_i - \overline{\mathbf{s}})^\top$ denote the sample mean vector and covariance matrix of $\mathcal{S}$, respectively. By construction of the estimator $\hat{\boldsymbol{\theta}}$, $\overline{\mathbf{s}} = 0$.

Now we consider a set of new observations $\mathcal{D}^{\text{new}} = \{(\mathbf{x}_{n+i}, y_{n+i}) : i = 1, 2, \cdots\}$ drawn i.i.d. from the same distribution as $\mathcal{D}$, to represent the situation that there is no shift in the predictive distribution. The score vector $\mathbf{s}_{n+i} = \mathbf{s}(\hat{\boldsymbol{\theta}}; \mathbf{x}_{n+i}, y_{n+i})$ for each new observation

depends on $(\mathbf{x}_{n+i}, y_{n+i}, \hat{\boldsymbol{\theta}})$. Let $\mathcal{S}^{\text{new}} = \{\mathbf{s}_{n+i} : i = 1, 2, \cdots\}$ denote the new score vectors, and note that $\mathcal{S}^{\text{new}}$ depends on $\mathcal{D}$ only via the fitted model parameters $\hat{\boldsymbol{\theta}}$. Although the regularization parameters γ are also estimated from $\mathcal{D}$, we treat them as fixed for tractability and use the same values when fitting all models within the bootstrapping procedure described below. Consequently, conditioned on $\hat{\boldsymbol{\theta}}$, $\mathcal{S}^{\text{new}}$ constitutes an i.i.d. sample with each $\mathbf{s}_{n+i}$, $i = 1, 2, \cdots$, having some common mean $\mu(\hat{\boldsymbol{\theta}})$ and covariance matrix $\mathbf{V}(\hat{\boldsymbol{\theta}})$ that are deterministic functions of $\hat{\boldsymbol{\theta}}$, which we denote by

$$\mathbf{s}_{n+i} | \hat{\boldsymbol{\theta}} \sim \text{i. i. d.}(\mu(\hat{\boldsymbol{\theta}}), \mathbf{V}(\hat{\boldsymbol{\theta}})), \quad i = 1, 2, \cdots . \tag{3.1}$$

Although $\bar{\mathbf{s}} = \mathbf{0}$, it is not the case that $\mu(\hat{\boldsymbol{\theta}}) = \mathbf{0}$, because of estimation error in $\hat{\boldsymbol{\theta}}$ and because $\mathbf{s}_{n+i}$ are computed for new observations that are independent of the training data $\mathcal{D}$ to which $g_\gamma(\hat{\boldsymbol{\theta}}; \mathbf{x})$ is fit. For similar reasons, $\widehat{\boldsymbol{\Sigma}}$ should not be viewed as an estimator of $\mathbf{V}(\hat{\boldsymbol{\theta}})$.

Let $\{\mathbf{z}_{n+i} : i = 1, 2, \cdots\}$ denote the MEWMAs of the score vectors in $\mathcal{S}^{\text{new}}$, defined recursively as $\mathbf{z}_{n+i} = \lambda \mathbf{s}_{n+i} + (1 - \lambda)\mathbf{z}_{n+i-1}$, where $\lambda \in (0, 1)$ is the smoothing parameter and $\mathbf{z}_n = \mathbf{0}$. Write $\mathbf{z}_{n+i} = \lambda\big[\mathbf{s}_{n+i} + (1-\lambda)\mathbf{s}_{n+i-1} + (1-\lambda)^2\mathbf{s}_{n+i-2} + \cdots + (1-\lambda)^{i-1}\mathbf{s}_{n+1}\big]$ for $i = 1, 2, \cdots$. From (3.1), conditioned on $\hat{\boldsymbol{\theta}}$, the conditional mean and covariance of $\mathbf{z}_{n+i}$ are

$$\mathbb{E}_0\Big[\mathbf{z}_{n+i} \,\Big|\, \hat{\boldsymbol{\theta}}\Big] = \big[1 - (1-\lambda)^i\big]\, \boldsymbol{\mu}(\hat{\boldsymbol{\theta}}), \qquad \text{Cov}_0\Big[\mathbf{z}_{n+i} \,\Big|\, \hat{\boldsymbol{\theta}}\Big] \;= \frac{\lambda}{2-\lambda}\big[1 - (1-\lambda)^{2i}\big]\, \mathbf{V}(\hat{\boldsymbol{\theta}}). \tag{3.2}$$

The derivations in the remainder of this section relate the distribution of $\mathbf{z}_{n+i}$ (for each $i = 1, 2, \cdots$) to the distribution of analogous MEWMA ($\mathbf{z}_i^{b,j}$ from Step IV (b) of Algorithm 1) on the bootstrapped score vectors, in order to determine the CL for the T^2 chart on $\{\mathbf{z}_{n+i} : i = 1, 2, \cdots\}$ as a function of i. Algorithm 1 provides an overview of our nested bootstrapping procedure to compute the CL. For each outer bootstrap replicate b $(= 1, 2, \cdots, B_O)$, let $\mathcal{D}^b = \{(\mathbf{x}_i^b, y_i^b) : i = 1, 2, \cdots, n\}$ denote the bootstrap sample of size n from $\mathcal{D}$, and let $\mathcal{D}_{OOB}^b = \{\big(\mathbf{x}_{\text{OOB},i}^b,\ y_{\text{OOB},i}^b\big) : i \in OOB^b\}$ denote the corresponding out-

of-bag (OOB) observations, where OOB^b represents the indices of observations in $\mathcal{D}$ that are not selected in bootstrap sample b. The score vectors computed from these samples are denoted as $\mathcal{S}^b = \{\mathbf{s}_i^b : i = 1, 2, \cdots, n\}$ and $\mathcal{S}_{OOB}^b = \{\mathbf{s}_{OOB,i}^b : i \in OOB^b\}$ respectively, where $\mathbf{s}_i^b = \mathbf{s}(\hat{\boldsymbol{\theta}}^b; \mathbf{x}_i^b, y_i^b)$, $\mathbf{s}_{\text{OOB},i}^b = \mathbf{s}(\hat{\boldsymbol{\theta}}^b; \mathbf{x}_{\text{OOB},i}^b, y_{\text{OOB},i}^b)$, and $\hat{\boldsymbol{\theta}}^b$ are the parameters of the model fit to $\mathcal{D}^b$ in Step 3-II of Algorithm 1. Let $\bar{\mathbf{s}}^b = 1/n \sum_{i=1}^n \mathbf{s}_i^b$ and $\widehat{\boldsymbol{\Sigma}}^b = 1/n \sum_{i=1}^n (\mathbf{s}_i^b - \bar{\mathbf{s}}^b)(\mathbf{s}_i^b - \bar{\mathbf{s}}^b)^\top$ denote the sample mean vector and covariance matrix of $\mathcal{S}^b$. For each inner bootstrap replicate j $(= 1, 2, \cdots, B_I)$ within outer replicate b, we denote the bootstrap sample of score vectors drawn (with replacement) from $\mathcal{S}_{OOB}^b$ as $\{\mathbf{s}_i^{b,j} : i = 1, 2, \cdots\}$ and their corresponding MEWMA as $\mathbf{z}_i^{b,j}$ from Step IV(b) of Algorithm 1.

Algorithm 1: Nested Bootstrap Algorithm for Control Chart Setup

Input: the full training sample $\mathcal{D}$; the desired false-alarm probability α; and the MEWMA parameter λ.

Result: the upper CL CL_i for $i = 1, 2, 3, \cdots$.

1) Fit and tune a supervised learning model $g_\gamma(\hat{\boldsymbol{\theta}}; \mathbf{x})$ to $\mathcal{D}$ using CV to select hyperparameters γ.

2) Apply the model to compute the score vectors $\mathcal{S}$, and their sample mean $\bar{\mathbf{s}}$ and covariance matrix $\widehat{\boldsymbol{\Sigma}}$.

3) Using the following nested bootstrapping procedure, determine CL $\{CL_i : i = 1, 2, 3, \cdots\}$ for the T^2 statistic.

For $b = 1, 2, \cdots, B_O$ (outer bootstrap loop):

I. Draw a bootstrap sample $\mathcal{D}^b$ of size n from $\mathcal{D}$, and identify the OOB observations $\mathcal{D}^b_{OOB}$.

II. Using the same tuning parameters γ from Step 1, fit a new model $g_\gamma(\hat{\boldsymbol{\theta}}^b; \mathbf{x})$ to $\mathcal{D}^b$.

III. Compute the score vectors $\mathcal{S}^b$ and $\mathcal{S}^b_{OOB}$, and the sample mean $\bar{\mathbf{s}}^b$ and covariance matrix $\widehat{\boldsymbol{\Sigma}}^b$ of $\mathcal{S}^b$.

IV. For $j = 1, 2, \cdots, B_I$ (inner bootstrap loop):

(a) Draw a bootstrap sample of score vectors $\{\mathbf{s}_i^{b,j} : i = 1, 2, 3, \cdots\}$ from $\mathcal{S}^b_{OOB}$. Initialize $\mathbf{z}_0^{b,j} = \mathbf{0}$.

(b) For $i = 1, 2, 3, \cdots$, compute the MEWMA $\mathbf{z}_i^{b,j} = \lambda \mathbf{s}_i^{b,j} + (1 - \lambda)\mathbf{z}_{i-1}^{b,j}$ and the T^2 statistic in (3.20).

For each $i = 1, 2, 3, \cdots$, set CL_i to be the upper α quantile of $\{T_i^{b,j} : b = 1, 2, \cdots, B_O;\ j = 1, 2, \cdots, B_I\}$.

Within the context of the nested bootstrap procedure, each outer bootstrap sample $\mathcal{D}^b$ in Step 3-I assumes the role of $\mathcal{D}$, and will account for variability in $\hat{\boldsymbol{\theta}}$ that results from

fitting the model to the training data, and its OOB sample $\mathcal{D}^b_{\text{OOB}}$ assumes the role of $\mathcal{D}^{\text{new}}$ and will account for variability in the future data. A single (non-nested) boostrap procedure would only account for the latter and underestimate the CL. Let n^b denote the number of unique observations in $\mathcal{D}$ that were sampled in $\mathcal{D}^b$, in which case $\left|\mathcal{D}^b_{\text{OOB}}\right| = n - n^b$. By the 0.632 rule, unless n is exceptionally small, we are guaranteed that $n^b \cong 0.632n$, in which case $|\mathcal{D}^b_{OOB}| \cong 0.368n$. For notational simplicity, in the following derivations we assume each $n^b = 0.632n$.

Remark 3.1. *For each $i = 1, 2, \cdots$, one might consider using the empirical distribution of a naïve version of the bootstrapped T^2 statistics $\{(\mathbf{z}_i^{b,j} - \bar{\mathbf{s}}^b)^\top (\widehat{\boldsymbol{\Sigma}}^b)^{-1}(\mathbf{z}_i^{b,j} - \bar{\mathbf{s}}^b) : b = 1, 2, \cdots, B_O; j = 1, 2, \cdots, B_I\}$ to approximate the distribution of the T^2 statistic $T_{n+i} = (\mathbf{z}_{n+i} - \bar{\mathbf{s}})^\top \hat{\Sigma}^{-1}(\mathbf{z}_{n+i} - \bar{\mathbf{s}})$ for $\mathcal{S}^{new}$ and compute its control limit. However, this naïve bootstrap distribution can be severely biased due to the finite sample size of $\mathcal{S}^b_{OOB}$, for reasons that become clear later (also see Remark 3.2 below). The following derivations account for this bias and result in control limits that accurately control false-alarm rate.*

To derive appropriate control limits, we make the following standard bootstrap assumption.

Assumption 3.1. *(The typical bootstrap assumption) If $\mathcal{D}^{new}$ is drawn from the same distribution as $\mathcal{D}$, the joint distribution of $(\mathcal{S}^{new}_{.368}, \bar{\mathbf{s}}, \widehat{\boldsymbol{\Sigma}}, \hat{\boldsymbol{\theta}})$ is the same as the joint distribution of the analogous quantities $\left(\mathcal{S}^b_{OOB}, \bar{\mathbf{s}}^b, \widehat{\boldsymbol{\Sigma}}^b, \hat{\boldsymbol{\theta}}^b\right)$ from the bootstrapping procedure, where $\mathcal{S}^{new}_{.368}$ is any randomly drawn subset of $\mathcal{S}^{new}$ having the same cardinality $0.368n$ as $\mathcal{S}^b_{OOB}$.*

The Assumption 3.1, together with (3.1), imply that

$$\mathbf{s}^b_{OOB,i} | \hat{\boldsymbol{\theta}}^b \sim iid \left(\mu\left(\hat{\boldsymbol{\theta}}^b\right), \mathbf{V}\left(\hat{\boldsymbol{\theta}}^b\right)\right), \quad i = 1, 2, 3, \cdots. \tag{3.3}$$

In the inner loop of the nested bootstrapping procedure of Algorithm 1, $\mathcal{S}^b_{OOB}$ serves as the "population" from which the bootstrapped score vectors $\{\mathbf{s}_i^{b,j} : i = 1, 2, 3, \cdots\}$ for each inner

replicate j are drawn with replacement, having population mean and covariance

$$\overline{\mathbf{s}}^b_{OOB} = \frac{1}{\lfloor 0.368n \rfloor} \sum_{i=1}^{\lfloor 0.368n \rfloor} \mathbf{s}^b_{OOB,i}, \tag{3.4}$$

and

$$\widehat{\boldsymbol{\Sigma}}^b_{OOB} = \frac{1}{\lfloor 0.368n \rfloor} \sum_{i=1}^{\lfloor 0.368n \rfloor} \left(\mathbf{s}^b_{OOB,i} - \overline{\mathbf{s}}^b_{OOB}\right)\left(\mathbf{s}^b_{OOB,i} - \overline{\mathbf{s}}^b_{OOB}\right)^\top, \tag{3.5}$$

where from (3.3),

$$\overline{\mathbf{s}}^b_{OOB} | \, \hat{\boldsymbol{\theta}}^b \sim \left(\mu\left(\hat{\boldsymbol{\theta}}^b\right), \frac{\mathbf{V}(\hat{\boldsymbol{\theta}}^b)}{\lfloor 0.368n \rfloor}\right), \tag{3.6}$$

and

$$\mathbb{E}_0\left[\widehat{\boldsymbol{\Sigma}}^b_{OOB} \middle| \hat{\boldsymbol{\theta}}^b\right] = \frac{\lfloor 0.368n \rfloor - 1}{\lfloor 0.368n \rfloor} \mathbf{V}\left(\hat{\boldsymbol{\theta}}^b\right) \cong \mathbf{V}\left(\hat{\boldsymbol{\theta}}^b\right). \tag{3.7}$$

For each $i = 1, 2, 3, \cdots$, the preceding provides a means to relate the distribution of each $\mathbf{z}^{b,j}_i$ computed in the inner bootstrap loop to the distribution of $\mathbf{z}_{n+i}$ for the new data. Since $\{\mathbf{s}^{b,j}_i : i = 1, 2, 3, \cdots\}$ is drawn randomly with replacement from $\mathcal{S}^b_{OOB}$, it follows from (3.4) - (3.7) that the conditional mean and covariance of $\mathbf{z}^{b,j}_i = \lambda\big[\mathbf{s}^{b,j}_i + (1-\lambda)\mathbf{s}^{b,j}_{i-1} + (1-\lambda)^2\mathbf{s}^{b,j}_{i-2} + \cdots + (1-\lambda)^{i-1}\mathbf{s}^{b,j}_1\big]$ are

$$\begin{aligned}
\mathbb{E}_0\Big[\mathbf{z}^{b,j}_i \;\Big|\; \hat{\boldsymbol{\theta}}^b\Big] &= \mathbb{E}_0\Big[\mathbb{E}_0\Big\{\lambda\big[\mathbf{s}^{b,j}_i + (1-\lambda)\mathbf{s}^{b,j}_{i-1} + (1-\lambda)^2\mathbf{s}^{b,j}_{i-2} + \cdots + (1-\lambda)^{i-1}\mathbf{s}^{b,j}_1\big] \;\Big|\; \hat{\boldsymbol{\theta}}^b, \mathcal{D}^b_{\text{OOB}}\Big\} \;\Big|\; \hat{\boldsymbol{\theta}}^b\Big] \\
&= \mathbb{E}_0\Big[\lambda\mathbb{E}_0\Big[\mathbf{s}^{b,j}_i \;\Big|\; \hat{\boldsymbol{\theta}}^b, \mathcal{D}^b_{\text{OOB}}\Big] + \lambda(1-\lambda)\mathbb{E}_0\Big[\mathbf{s}^{b,j}_{i-1} \;\Big|\; \hat{\boldsymbol{\theta}}^b, \mathcal{D}^b_{\text{OOB}}\Big] \\
&\qquad + \cdots + \lambda(1-\lambda)^{i-1}\mathbb{E}_0\Big[\mathbf{s}^{b,j}_1 \;\Big|\; \hat{\boldsymbol{\theta}}^b, \mathcal{D}^b_{\text{OOB}}\Big] \;\Big|\; \hat{\boldsymbol{\theta}}^b\Big] \\
&= \mathbb{E}_0\Big[\lambda\sum_{k=0}^{i-1}(1-\lambda)^k\,\overline{\mathbf{s}}^b_{OOB} \;\Big|\; \hat{\boldsymbol{\theta}}^b\Big] = \lambda\sum_{k=0}^{i-1}(1-\lambda)^k\,\mathbb{E}_0\Big[\overline{\mathbf{s}}^b_{OOB} \;\Big|\; \hat{\boldsymbol{\theta}}^b\Big] = \big[1-(1-\lambda)^i\big]\,\boldsymbol{\mu}\big(\hat{\boldsymbol{\theta}}^b\big).
\end{aligned} \tag{3.8}$$

and

$$
\begin{aligned}
\mathrm{Cov}_0\Big[\mathbf{z}_i^{b,j} \;\Big|\; \hat{\boldsymbol{\theta}}^{\,b}\Big] &= \mathbb{E}_0\Big[\mathrm{Cov}\Big(\lambda\big[\mathbf{s}_i^{b,j} + (1-\lambda)\mathbf{s}_{i-1}^{b,j} + (1-\lambda)^2\mathbf{s}_{i-2}^{b,j} + \cdots + (1-\lambda)^{i-1}\mathbf{s}_1^{b,j}\big] \;\Big|\; \hat{\boldsymbol{\theta}}^{\,b}, \mathcal{D}_{\mathrm{OOB}}^b\Big)\,\Big|\,\hat{\boldsymbol{\theta}}^{\,b}\Big] \\
&\quad + \mathrm{Cov}_0\Big[\mathbb{E}_0\Big(\lambda\big[\mathbf{s}_i^{b,j} + (1-\lambda)\mathbf{s}_{i-1}^{b,j} + (1-\lambda)^2\mathbf{s}_{i-2}^{b,j} + \cdots + (1-\lambda)^{i-1}\mathbf{s}_1^{b,j}\big] \;\Big|\; \hat{\boldsymbol{\theta}}^{\,b}, \mathcal{D}_{\mathrm{OOB}}^b\Big)\,\Big|\,\hat{\boldsymbol{\theta}}^{\,b}\Big] \\
&= \mathbb{E}_0\Big[\lambda^2\sum_{k=0}^{i-1}(1-\lambda)^{2k}\,\mathrm{Cov}_0\Big[\mathbf{s}_{i-k}^{b,j} \;\Big|\; \hat{\boldsymbol{\theta}}^{\,b}, \mathcal{D}_{\mathrm{OOB}}^b\Big]\,\Big|\,\hat{\boldsymbol{\theta}}^{\,b}\Big] \\
&\qquad + \mathrm{Cov}_0\Big\{\lambda\sum_{k=0}^{i-1}(1-\lambda)^k\mathbb{E}_0\Big[\mathbf{s}_{i-k}^{b,j} \;\Big|\; \hat{\boldsymbol{\theta}}^{\,b}, \mathcal{D}_{\mathrm{OOB}}^b\Big]\,\Big|\,\hat{\boldsymbol{\theta}}^{\,b}\Big\} \\
&= \mathbb{E}_0\Big[\lambda^2\sum_{k=0}^{i-1}(1-\lambda)^{2k}\widehat{\boldsymbol{\Sigma}}_{OOB}^b\,\Big|\,\hat{\boldsymbol{\theta}}^{\,b}\Big] + \mathrm{Cov}_0\Big[\lambda\sum_{k=0}^{i-1}(1-\lambda)^k\bar{\mathbf{s}}_{OOB}^b\,\Big|\,\hat{\boldsymbol{\theta}}^{\,b}\Big] \\
&= \mathbb{E}_0\Big[\tfrac{\lambda}{2-\lambda}\,\hat{\boldsymbol{\Sigma}}_{\mathrm{OOB}}^{\,b}\big[1-(1-\lambda)^{2i}\big]\,\Big|\,\hat{\boldsymbol{\theta}}^{\,b}\Big] + \mathrm{Cov}_0\Big\{\big[1-(1-\lambda)^i\big]\,\bar{\mathbf{s}}_{\mathrm{OOB}}^b\,\Big|\,\hat{\boldsymbol{\theta}}^{\,b}\Big\} \\
&\cong \tfrac{\lambda}{2-\lambda}\big[1-(1-\lambda)^{2i}\big]\,\mathbf{V}\big(\hat{\boldsymbol{\theta}}^{\,b}\big) + \tfrac{1}{0.368\,n}\big[1-(1-\lambda)^i\big]^2\,\mathbf{V}\big(\hat{\boldsymbol{\theta}}^{\,b}\big) \\
&= \Big\{\tfrac{\lambda}{2-\lambda}\big[1-(1-\lambda)^{2i}\big] + \tfrac{1}{0.368\,n}\big[1-(1-\lambda)^i\big]^2\Big\}\mathbf{V}\big(\hat{\boldsymbol{\theta}}^{\,b}\big).
\end{aligned}
\tag{3.9}
$$

The additive term $[1-(1-\lambda)^i]^2/0.368n$ quantifies the extra variability introduced by estimating the mean score from an out-of-bag sample that contains on average $(1-0.632)n = 0.368n$ points rather than the full training size. Including this 632-style correction inflates the conditional variance to the appropriate level and is essential for the resulting CL to achieve the target false-alarm rate (also see Remark 3.2 below).

From (3.8) and (3.9), the unconditional mean and covariance of $\mathbf{z}_i^{b,j}$ are

$$
\mathbb{E}_0\left[\mathbf{z}_i^{b,j}\right] = \mathbb{E}_0\left[\mathbb{E}_0\left[\mathbf{z}_i^{b,j}\middle|\hat{\boldsymbol{\theta}}^{b}\right]\right] = \left[1-(1-\lambda)^i\right]\mathbb{E}_0\left[\mu\left(\hat{\boldsymbol{\theta}}^{b}\right)\right], \tag{3.10}
$$

and

$$
\begin{aligned}
\mathrm{Cov}_0\left[\mathbf{z}_i^{b,j}\right] &= \mathbb{E}_0\left[\mathrm{Cov}_0\left[\mathbf{z}_i^{b,j}\middle|\hat{\boldsymbol{\theta}}^{b}\right]\right] + \mathrm{Cov}_0\left[\mathbb{E}_0\left[\mathbf{z}_i^{b,j}\middle|\hat{\boldsymbol{\theta}}^{b}\right]\right] \\
&= \left\{\frac{\lambda}{2-\lambda}[1-(1-\lambda)^{2i}] + \frac{1}{0.368n}[1-(1-\lambda)^i]^2\right\}\mathbb{E}_0\left[\mathbf{V}\left(\hat{\boldsymbol{\theta}}^{b}\right)\right] + \left[1-(1-\lambda)^i\right]^2\mathrm{Cov}_0[\boldsymbol{\mu}(\hat{\boldsymbol{\theta}}^{\,b})].
\end{aligned}
\tag{3.11}
$$

To approximate the terms on the right hand sides of (3.10) and (3.11) and relate them to (3.2), we denote by $(\mathbf{X}, Y)$ a predictor-response pair randomly drawn from the population $\mathbb{P}_0(Y, \mathbf{X})$, or equivalently from $\mathcal{D}^b_{OOB}$, and we consider the first-order Taylor approximations of $\mu\left(\hat{\boldsymbol{\theta}}^b\right)$ and $\mathbf{V}\left(\hat{\boldsymbol{\theta}}^b\right)$ about $\hat{\boldsymbol{\theta}}^b \cong \boldsymbol{\theta}^0$, where $\boldsymbol{\theta}^0$ denotes the true values of the parameters. Assume that the functions $\mu(\boldsymbol{\theta})$ and $\mathbf{V}(\boldsymbol{\theta})$ are continuously differentiable with respect to $\boldsymbol{\theta}$ in a neighborhood of $\boldsymbol{\theta}^0$. Then, it holds that

$$\begin{aligned}
\mu\left(\hat{\boldsymbol{\theta}}^b\right) &\cong \mu\left(\boldsymbol{\theta}^0\right) + \nabla_{\boldsymbol{\theta}^\top}\mu(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0}\left(\hat{\boldsymbol{\theta}}^b - \boldsymbol{\theta}^0\right) \\
&= \mathbb{E}_0\left[\nabla_{\boldsymbol{\theta}} \log \mathbb{P}\left(Y|\mathbf{X};\boldsymbol{\theta}\right)|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0}\right] + \nabla_{\boldsymbol{\theta}^\top}\mathbb{E}_0\left[\nabla_{\boldsymbol{\theta}} \log \mathbb{P}\left(Y|\mathbf{X};\boldsymbol{\theta}\right)\right]|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0}\left(\hat{\boldsymbol{\theta}}^b - \boldsymbol{\theta}^0\right) \qquad (3.12) \\
&= \mathbf{0} + \mathbb{E}_0\left[\nabla^2_{\boldsymbol{\theta}} \log \mathbb{P}\left(Y|\mathbf{X};\boldsymbol{\theta}\right)\right]\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0}\left(\hat{\boldsymbol{\theta}}^b - \boldsymbol{\theta}^0\right) = -\mathbf{I}\left(\boldsymbol{\theta}^0\right)\left(\hat{\boldsymbol{\theta}}^b - \boldsymbol{\theta}^0\right),
\end{aligned}$$

where we assume sufficient regularity conditions on the log-likelihood to allow the expectation and gradient operators to be exchanged, $\mathbf{I}(\boldsymbol{\theta}^0)$ denotes the expected Fisher information matrix, and we have used the standard result that, at the true parameters $\boldsymbol{\theta}^0$, the score function is zero-mean. Similarly,

$$\mathbf{V}\left(\hat{\boldsymbol{\theta}}^b\right) \cong \mathbf{V}\left(\boldsymbol{\theta}^0\right) + \nabla_{\boldsymbol{\theta}^\top}\mathbf{V}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0}\left(\hat{\boldsymbol{\theta}}^b - \boldsymbol{\theta}^0\right) = \mathbf{I}\left(\boldsymbol{\theta}^0\right) + \nabla_{\boldsymbol{\theta}^\top}\mathbf{V}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0}\left(\hat{\boldsymbol{\theta}}^b - \boldsymbol{\theta}^0\right), \quad (3.13)$$

where we have used the standard result that the covariance matrix of the score vector at $\boldsymbol{\theta}^0$ is $\mathbf{I}(\boldsymbol{\theta}^0)$. Using the standard asymptotic result $\hat{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{\theta}^0, \mathbf{I}^{-1}(\boldsymbol{\theta}^0)/n)$ (which by Assumption 3.1 is also the asymptotic distribution of $\hat{\boldsymbol{\theta}}^b$) with (3.12) and (3.13) gives

$$\mathbb{E}_0\left[\mu\left(\hat{\boldsymbol{\theta}}^b\right)\right] \cong \mathbb{E}_0\left[-\mathbf{I}\left(\boldsymbol{\theta}^0\right)\left(\hat{\boldsymbol{\theta}}^b - \boldsymbol{\theta}^0\right)\right] \cong \mathbf{0}, \quad (3.14)$$

$$\mathrm{Cov}_0\left[\mu\left(\hat{\boldsymbol{\theta}}^b\right)\right] \cong \mathrm{Cov}_0\left[-\mathbf{I}\left(\boldsymbol{\theta}^0\right)\left(\hat{\boldsymbol{\theta}}^b - \boldsymbol{\theta}^0\right)\right] \cong \mathbf{I}\left(\boldsymbol{\theta}^0\right)\frac{\mathbf{I}^{-1}(\boldsymbol{\theta}^0)}{n}\mathbf{I}\left(\boldsymbol{\theta}^0\right) = \frac{\mathbf{I}(\boldsymbol{\theta}^0)}{n}, \quad (3.15)$$

and

$$\mathbb{E}_0\left[\mathbf{V}\left(\hat{\boldsymbol{\theta}}^b\right)\right] \cong \mathbb{E}_0\left[\mathbf{I}\left(\boldsymbol{\theta}^0\right) + \frac{\partial \mathbf{V}(\boldsymbol{\theta}^0)}{\partial \boldsymbol{\theta}^\top}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0}\left(\hat{\boldsymbol{\theta}}^b - \boldsymbol{\theta}^0\right)\right] = \mathbf{I}\left(\boldsymbol{\theta}^0\right). \quad (3.16)$$

Substituting (3.14)-(3.16) into (3.10) and (3.11) gives

$$\mathbb{E}_0\left[\mathbf{z}_i^{b,j}\right] \cong \mathbf{0}, \quad \mathrm{Cov}_0\left[\mathbf{z}_i^{b,j}\right] = \left\{\frac{\lambda}{2-\lambda}[1-(1-\lambda)^{2i}] + \frac{3.72}{n}[1-(1-\lambda)^{i}]^2\right\}\mathbf{I}\left(\boldsymbol{\theta}^0\right). \quad (3.17)$$

Using equations (3.2) and repeating the derivations of (3.12)-(3.16) for $\hat{\boldsymbol{\theta}}$ instead of for $\hat{\boldsymbol{\theta}}^b$ gives the analogous results

$$\mathbb{E}_0\left[\mathbf{z}_i\right] \cong \mathbf{0}, \quad \mathrm{Cov}_0\left[\mathbf{z}_i\right] \cong \left\{\frac{\lambda}{2-\lambda}[1-(1-\lambda)^{2i}] + \frac{1}{n}[1-(1-\lambda)^{i}]^2\right\}\mathbf{I}\left(\boldsymbol{\theta}^0\right). \quad (3.18)$$

Comparing (3.17) with (3.18), we see that $\mathbf{z}_i^{b,j}$ and $\mathbf{z}_i$ have the same mean $\mathbf{0}$ and covariance matrices that are scalar multiples of $\mathbf{I}(\boldsymbol{\theta}^0)$. Relative to $\mathbf{z}_i$, the covariance matrix of $\mathbf{z}_i^{b,j}$ is inflated by the factor

$$k\left(\lambda, i, n\right) := \frac{\frac{\lambda}{2-\lambda}[1-(1-\lambda)^{2i}] + \frac{3.72}{n}[1-(1-\lambda)^{i}]^2}{\frac{\lambda}{2-\lambda}[1-(1-\lambda)^{2i}] + \frac{1}{n}[1-(1-\lambda)^{i}]^2}. \quad (3.19)$$

This provides the basis for taking the CL function CL_i in Algorithm 1 to be the upper α quantile of the quantities $\{T_i^{b,j} : b = 1, 2, \cdots, B_0; j = 1, 2, \cdots, B_I\}$, which serves as an approximation to the upper α quantile of the distribution of T_{n+i}, where

$$T_i^{b,j} \equiv \left(\frac{\mathbf{z}_i^{b,j}}{\sqrt{k(\lambda,i,n)}} - \overline{\mathbf{s}}^b\right)^\top \left(\widehat{\boldsymbol{\Sigma}}^b\right)^{-1} \left(\frac{\mathbf{z}_i^{b,j}}{\sqrt{k(\lambda,i,n)}} - \overline{\mathbf{s}}^b\right), \quad (3.20)$$

and

$$T_{n+i} \equiv \left(\mathbf{z}_{n+i} - \overline{\mathbf{s}}\right)^\top \widehat{\boldsymbol{\Sigma}}^{-1} \left(\mathbf{z}_{n+i} - \overline{\mathbf{s}}\right). \quad (3.21)$$

Remark 3.2. *Comparing* (3.17) *and* (3.18)*, we see that* $\mathrm{Cov}_0[\mathbf{z}_i^{b,j}] = \mathrm{Cov}_0[\mathbf{z}_i] + [1-(1-\lambda)^i]^2\mathbf{I}(\boldsymbol{\theta}^0)/0.368n$*, where, from equations* (3.9) *and* (3.11)*, the term* $I(\boldsymbol{\theta}^0)/(0.368\,n)$ *in the difference is precisely* $\mathbb{E}_0[\mathrm{Cov}_0[\overline{\mathbf{s}}_{\mathrm{OOB}}^b \mid \hat{\boldsymbol{\theta}}^b]]$*. Thus, this difference accounts for the variability in the mean* $\overline{\mathbf{s}}_{\mathrm{OOB}}^b$ *of the "population"* $\mathcal{S}_{\mathrm{OOB}}^b$ *(whose cardinality is* $0.368\,n$*) from which the inner bootstrap samples are drawn in Algorithm 1. This constitutes a nuanced instantiation*

of the 0.632 *bootstrap rule that is crucial for obtaining a correct CL. When this correction is omitted (in which case the scaling factor* $k(\lambda, i, n)$ *in* (3.20) *is replaced by* 1*), we have observed empirically that the bootstrapped CL is typically far too large and the detection power is unnecessarily compromised.*

Several implementation aspects of the algorithm warrant further discussion. When fitting models to bootstrap samples in Step 3-II, it is important to use the same tuning parameters γ that were selected in Step 1 rather than performing new CV, as this maintains consistency in the model structure across replicates. If the covariance matrices $\widehat{\mathbf{\Sigma}}$ in Step 2 and $\widehat{\mathbf{\Sigma}}^b$ in Step 3-III are poorly conditioned (e.g., when $\hat{\boldsymbol{\theta}}^b$ is high-dimensional and/or has highly correlated components), then $\epsilon\mathbf{I}$ for some small scalar ϵ should be added to them before they are inverted in (3.20) and (3.21). If this is done, it should be done consistently in Steps 2 and 3-III using the same value of ϵ. Note that $\widehat{\mathbf{\Sigma}}^b$ will typically be more poorly conditioned than $\widehat{\mathbf{\Sigma}}$, since there are fewer distinct score vectors in $\mathbf{S}_{\mathcal{D}}^b$ than in $\mathbf{S}_{\mathcal{D}}$. Thus, whether $\widehat{\mathbf{\Sigma}}$ is replaced by $\widehat{\mathbf{\Sigma}} + \epsilon\mathbf{I}$ in Step 2 should depend on how poorly conditioned the $\widehat{\mathbf{\Sigma}}^b$ are in step 3-III. The value for the desired false-alarm probability α (e.g., 0.01, 0.001, etc) should be chosen to balance detection power (which decreases as α decreases) with false-alarm control (which decreases as α decreases). The number of bootstrap replicates B_O and B_I should be chosen with computational considerations in mind. Because the dominant cost comes from refitting the model to each outer bootstrap sample $\mathcal{D}^b$, we use a smaller B_O together with a larger B_I to obtain accurate quantile estimates at reasonable runtime. In all of our numerical simulations in Section 4 we used $B_O = 100$, $B_I = 200$, and set the false-alarm probability to $\alpha = 0.001$.

After using Algorithm 1 to establish the CL, monitoring for concept drift in new observations is straightforward. Given each new observation $(\mathbf{x}_{n+i}, y_{n+i})$, we compute its score vector $\mathbf{s}_i$ using the model $g_\gamma(\hat{\boldsymbol{\theta}}; \mathbf{x})$, update the MEWMA statistic via $\mathbf{z}_{n+i} = \lambda\mathbf{s}_{n+i} + (1-\lambda)\mathbf{z}_{n+i-1}$, and compute the T^2 monitoring statistics (3.21).

Concept drift is detected when T_{n+i} exceeds its corresponding CL CL_i. The choice of the

MEWMA smoothing parameter λ also affects the monitoring sensitivity, and the tradeoff is the same as in any MEWMA monitoring procedure: Smaller values of λ provide more powerful eventual detection of large shifts but can delay the detection of large shifts, whereas larger values provide quicker detection of large shifts but may fail to detect smaller shifts. Finally, it is worth mentioning that our Algorithm 1 exhibits a high degree of parallelism, in that the outer procedures for $i = 1, 2, \cdots, B_O$ are independent, allowing for straightforward parallelization. This is especially important, since the primary computational expense of Algorithm 1 is in fitting the model to each outer bootstrap sample $\mathcal{D}^b$.

4 Numerical Examples

To illustrate our approach and demonstrate its effectiveness at controlling the false-alarm rate, we present two numerical examples. The first is a more transparent example in which the predictive relationship $\mathbb{P}(Y \mid \mathbf{X})$ is a mixture of two simple linear relationships, and the second involves a more complex nonlinear predictive relationship.

4.1 Mixed Linear Population

The data-generating process for this example involves the two linear models

$$y_i = 16x_i + 5 + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{NID}(0, 16), \tag{4.1}$$

and

$$y_i = 12x_i + 3 + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{NID}(0, 16), \tag{4.2}$$

where predictor values x_i are uniformly sampled from $[-\sqrt{3}, \sqrt{3}]$. The training data of size $n = 2,000$ are generated exclusively from model (4.1), yielding an pre-shift sample governed by a single linear relationship. The future data to be monitored comprise 1,000 observations: the first 200 points (pre-shift) follow the same single linear model (4.1), while the remaining

800 points (post-shift) are generated from a mixture that draws y_i from either (4.1) or (4.2) with equal probability 0.5, representing a shift in $\mathbb{P}(Y|\mathbf{X})$.

We fit a linear predictive model $g_\gamma(\hat{\boldsymbol{\theta}}; x) = \hat{\theta}_0 + \hat{\theta}_1 x$ via ridge regression with L_2 regularization parameter $\gamma = 0.1$ (i.e. with the loss function $\sum_{i=1}^{n}[(y_i - \mathbf{x}_i^\top \hat{\boldsymbol{\theta}})^2 + \gamma\|\hat{\boldsymbol{\theta}}\|^2/n]$, for which the penalized log-likelihood for observation i is

$$\ell_{\text{pen}}\left(\hat{\boldsymbol{\theta}}; \mathbf{x}_i, y_i\right) = \log \mathbb{P}\big(y_i \mid \mathbf{x}_i; \hat{\boldsymbol{\theta}}\big) - \frac{\gamma}{2n}\|\hat{\boldsymbol{\theta}}\|^2 = -\frac{1}{2}\big(y_i - \mathbf{x}_i^\top \hat{\boldsymbol{\theta}}\big)^2 - \frac{1}{2}\log\big(2\pi\sigma^2\big) - \frac{\gamma}{2n}\|\hat{\boldsymbol{\theta}}\|^2.$$

The score vectors are therefore

$$\mathbf{s}(\hat{\boldsymbol{\theta}}; \mathbf{x}_i, y_i) = \big(y_i - \mathbf{x}_i^\top \hat{\boldsymbol{\theta}}\big)\mathbf{x}_i - \frac{\gamma}{n}\,\hat{\boldsymbol{\theta}}. \tag{4.3}$$

We used MEWMA parameter $\lambda = 0.01$, which we chose via 5-fold CV, and desired pointwise false-alarm probability $\alpha = 0.001$. In Algorithm 1, we used $B_O = 100$ and $B_I = 200$.

In Figure 1, we visualize the transition from the pre-shift predictive relationship (4.1) to the post-shift relationship that is a mixture of (4.1) and (4.2). Although the shift appears quite small in Figure 1, Figure 2 demonstrates that it is still detectable. Figure 2 shows a typical evolution of our T^2 monitoring statistic over 1,000 observations with the first 200 following the pre-shift distribution and the last 800 the post-shift distribution. The T^2 statistic remains below the CL prior to the shift and then begins to increase immediately following the shift at time 201 until it first exceeds the CL at time 258, demonstrating the method's ability to detect the shift. This behaviour aligns with our derivation in Section 3, where changes in $\mathbb{P}(Y|\mathbf{X})$ were shown to induce non-zero means in the score vectors.

To better assess false-alarm control, the blue curve in Figure 3 shows the pointwise false-alarm rate over the first 1,000 observations when no shift occurs, estimated as the average pointwise false-alarm rate over 50 replicates. On each replicate, a different training data set was generated, the ridge regression model was fit to these data, Algorithm 1 was used to compute the CL as a function of time, a new set of 1,000 observations was generated from

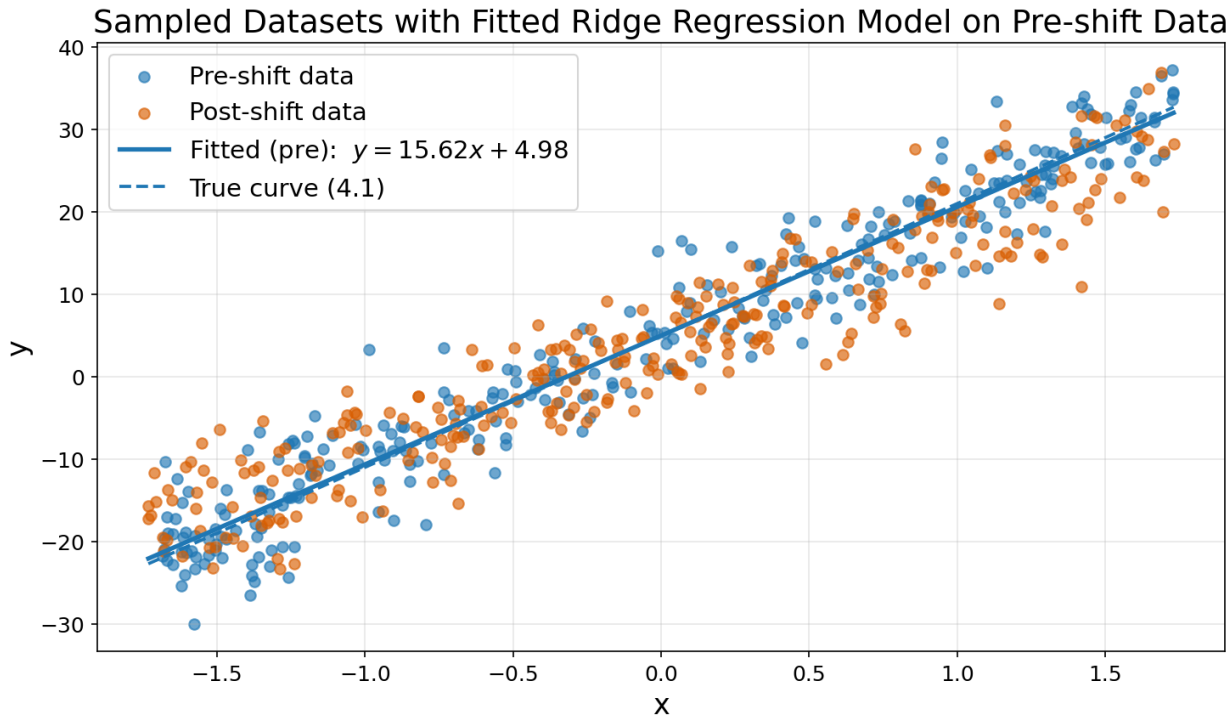


Figure 1: Visualization of predictive relationships for the linear example before and after the shift. The blue points are a scatter plot of y vs. x for the pre-shift single linear model (4.1), and the orange points correspond to the post-shift mixture model (4.1) and (4.2). The shift from a single component to a mixture alters $\mathbb{P}(Y|\mathbf{X})$, representing a structural shift in the predictive relationship that is relatively small but can still be detected by our approach.

the same distribution (representing no shift), and then the T^2 statistic was computed for each new observation and compared to its CL to determine if there was a signal at that time. Our method consistently maintains the false-alarm rate at close to the desired value $\alpha = 0.001$. The orange curve shows the corresponding results from the baseline method of [20], which exhibits substantially inflated false-alarm rates exceeding 0.06, highlighting the more accurate false-alarm rate provided by our bootstrap CL.

The discrepancy arises for two reasons. First, [20] used a constant CL, because without bootstrapping there is no clear way to compute a time-varying CL. Second, [20] fit their model on only the first half of the training data and compute the constant CL from the second half. The resulting CL is based on a much smaller effective sample size than in our nested bootstrapping procedure and cannot reliably provide correct Type I error control

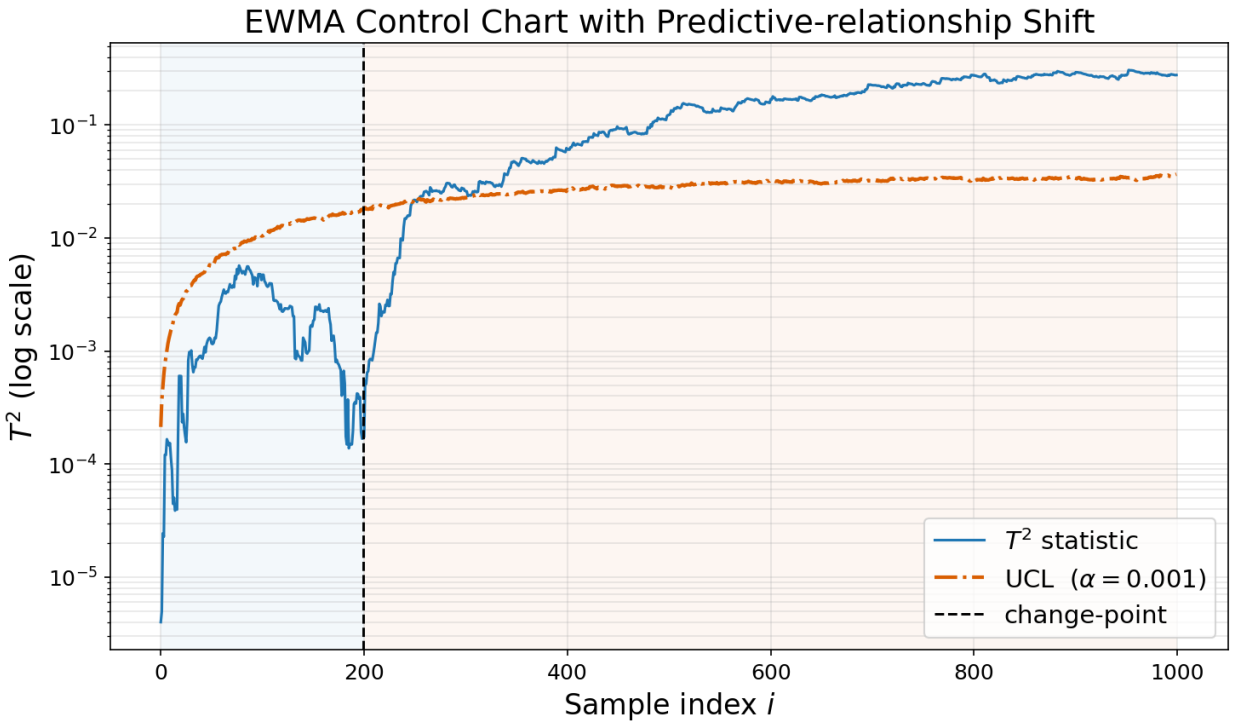


Figure 2: Typical monitoring results using our bootstrap CL approach for the linear mixture example showing the detection of concept drift at new observation number 258 (the shift first occurred at observation number 201). The T^2 statistics (T_i from Eq. (3.21), represented by the blue line) remain below the control limit (dashed red line) prior to the shift at observation 201, and sharply exceed it shortly after the shift.

unless the training sample size is quite large. In contrast, our approach provides accurate false-alarm rate control even with the relatively small training sample size of 2,000 in this example.

It should be noted that the intended application scenarios in [20] involved much larger training sample sizes and/or larger values for the desired α. For $\alpha = 0.01$ (for example), the method of [20] provides much more accurate false-alarm control than in this example with $\alpha = 0.001$. With $\alpha = 0.001$, the method of [20] generally requires that the size of the training subset used to compute the CL is at least 10,000 to accurately estimate the upper 0.001 quantile of the empirical distribution of the T^2 statistic. In this regard, our bootstrap CL approach can be viewed as an extension of the approach of [20] to smaller training sample sizes and/or larger desired α values.

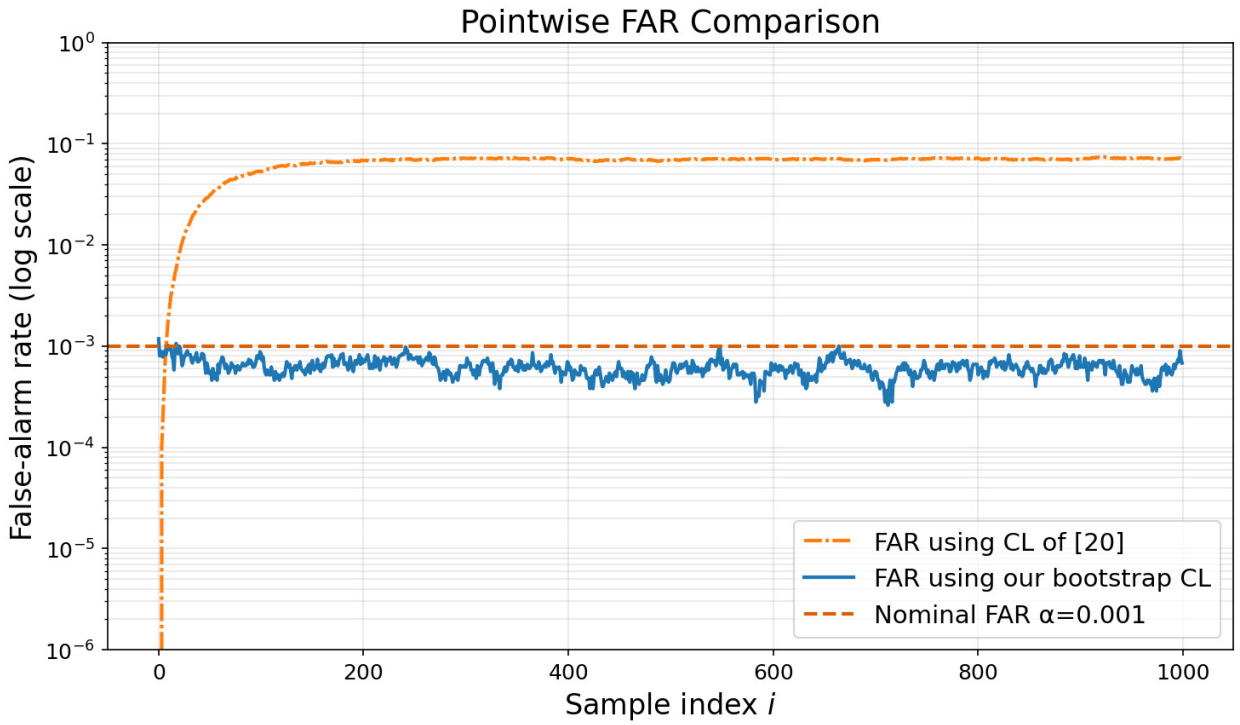


Figure 3: Comparison of the pointwise false-alarm rate for our bootstrap CL (blue curve) versus the CL of [20] (orange curve) averaged over 50 database replicates under the pre-shift predictive relationship (4.1). Our bootstrap CL much more accurately maintains the empirical false-alarm rate close to the desired $\alpha = 0.001$.

4.2 Nonlinear Oscillator System

To demonstrate our method's effectiveness in detecting changes in complex nonlinear predictive relationships, we consider a physics-based system that represents a variant of the two-degree-of-freedom nonlinear oscillator with a finite-extensibility coupling spring proposed by [7]. The system consists of two masses moving in one dimension, connected by nonlinear springs. Let $p_1(t)$ and $p_2(t)$ denote the positions at (continuous) time t of the two masses having masses m_1 and m_2, respectively, and let $v_1(t) = \dot{p}_1(t)$ and $v_2(t) = \dot{p}_2(t)$ denote their velocities. The system evolves according to the following differential equations

$$\begin{aligned} \ddot{p}_1(t) &= \frac{1}{m_1}(-k_1 p_1(t) - c_1 \dot{p}_1(t) + k_3 \phi(p_1(t), p_2(t))), \\ \ddot{p}_2(t) &= \frac{1}{m_2}(-k_2 p_2(t) - c_2 \dot{p}_2(t) - k_3 \phi(p_1(t), p_2(t))), \end{aligned} \tag{4.4}$$

where $\phi(p_1, p_2) = (p_1 - p_2)/(1 + |p_1 - p_2|)$ represents the nonlinear coupling force. We sample the oscillator uniformly on $[0, 30]$ to build a training set $\{(\mathbf{X}_i, y_i)\}_{i=1}^n$ with observations of $n = 3{,}000$. The feature vector $\mathbf{X}_i \in \mathbb{R}^4$ from the state of the system is

$$\mathbf{X}_i = [p_1(t_i), v_1(t_i), p_2(t_i), v_2(t_i)]^\top, \tag{4.5}$$

where the sampling interval is $\Delta t = t_i - t_{i-1} = 30/2\,999 \approx 0.01$s. The response variable y_i represents total mechanical energy (kinetic + potential) of the system at time t_i, given by

$$y_i = \frac{1}{2}(m_1 v_1^2(t_i) + m_2 v_2^2(t_i)) + \frac{1}{2}(k_1 p_1^2(t_i) + k_2 p_2^2(t_i)) + k_3 \phi\left(p_1(t_i), p_2(t_i)\right) + \epsilon_i, \tag{4.6}$$

where $\varepsilon_i \sim \mathcal{NID}(0, \sigma^2)$ represents measurement noise. For the training data, we used the system parameters $m_1 = 1.0$, $m_2 = 2.0$, $k_1 = 1.0$, $k_2 = 2.0$, $k_3 = 1.5$, $c_1 = 0.1$, and $c_2 = 0.2$. This setting presents a more challenging monitoring problem than the previous example because the predictive relationship is inherently nonlinear, and the predictive model that we fit to capture this is a neural network (described below).

To simulate concept drift, we generate new observations from the same system but with modified physical parameters

$$m_1' = 1.1 m_1, \quad m_2' = 1.2 m_2, \quad k_1' = 1.3 k_1, \tag{4.7}$$

while holding the remaining parameters fixed. These changes represent interpretable real-world perturbations, such as increased mass (e.g., due to material accumulation) or stiffened springs (e.g., due to thermal effects). They alter both the system's natural frequencies and the energy landscape, inducing subtle but systematic shifts in the predictive relationship that our monitoring framework aims to detect.

We fit a feed-forward neural network with one hidden layer of 4 ReLU units to model the predictive relationship. Let $\boldsymbol{\Theta} = (\mathbf{W}_1, \boldsymbol{b}_1, \boldsymbol{w}, b)$, where $\mathbf{W}_1 \in \mathbb{R}^{4\times 4}$ and $\boldsymbol{b}_1 \in \mathbb{R}^4$ denote

the weights and biases for the hidden layer, and let $\boldsymbol{w} \in \mathbb{R}^4$ and $b \in \mathbb{R}$ denote the weights and biases of the output layer. The resulting predictor is

$$g_\gamma(\boldsymbol{\Theta}; \mathbf{x}) = \boldsymbol{w}^\top \sigma\big(\mathbf{W}_1 \mathbf{x} + \boldsymbol{b}_1\big) + b, \qquad \sigma(\mathbf{z}) = \max\{\mathbf{0}, \mathbf{z}\},$$

where the vector max operation is element-by-element and given training data, the parameter vector monitored for drift is $\boldsymbol{\theta} = (\boldsymbol{w}, b)$. For large neural network models, [20] recommends defining $\boldsymbol{\theta}$ as only the parameters of the last layer to keep its dimension reasonable and since changes in the predictive relationship should be reflected as changes in the parameters of the last layer when the parameters of the earlier layers are fixed, and we adopt this convention here. When fitting the model to the training data the complete set of parameters are estimated, but for the purpose of monitoring the reduced dimension $\boldsymbol{\theta}$ with $\mathbf{W}_1$ and $\boldsymbol{b}_1$ viewed as fixed, the penalized log likelihood is

$$\ell_{\text{pen}}(\boldsymbol{\theta}; \mathbf{x}_i, y_i) = -\frac{1}{2\sigma^2}\big(y_i - g_\gamma(\boldsymbol{\theta}; \mathbf{x}_i)\big)^2 - \frac{1}{2}\log\big(2\pi\sigma^2\big) - \frac{\gamma}{2n}\big(\|\boldsymbol{w}\|^2 + b^2\big),$$

with regularization parameter $\gamma = 10^{-1}$. For an observation $(\mathbf{x}_i, y_i)$, the corresponding score vector is the gradient of this log-likelihood with respect to $\boldsymbol{\theta}$, evaluated at the fitted parameters $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{w}}, \hat{b})$, is

$$\mathbf{s}(\hat{\boldsymbol{\theta}}; \mathbf{x}_i, y_i) = \frac{y_i - g_\gamma(\hat{\boldsymbol{\theta}}; \mathbf{x}_i)}{\sigma^2} \begin{bmatrix} \sigma\big(\hat{\mathbf{W}}_1 \mathbf{x}_i + \hat{\boldsymbol{b}}_1\big) \\ 1 \end{bmatrix} - \frac{\gamma}{n} \begin{bmatrix} \hat{\boldsymbol{w}} \\ \hat{b} \end{bmatrix} \in \mathbb{R}^5.$$

The explicit form is shown for clarity and the implementation does not hand-code this gradient. Instead, for each observation we call `loss.backward()` in PyTorch, letting automatic differentiation compute the score vector directly from the penalized loss. This keeps the monitoring code concise and readily adaptable to more complex network architectures.

After fitting the network on this baseline data, we run Algorithm 1 with $B_O = 100$,

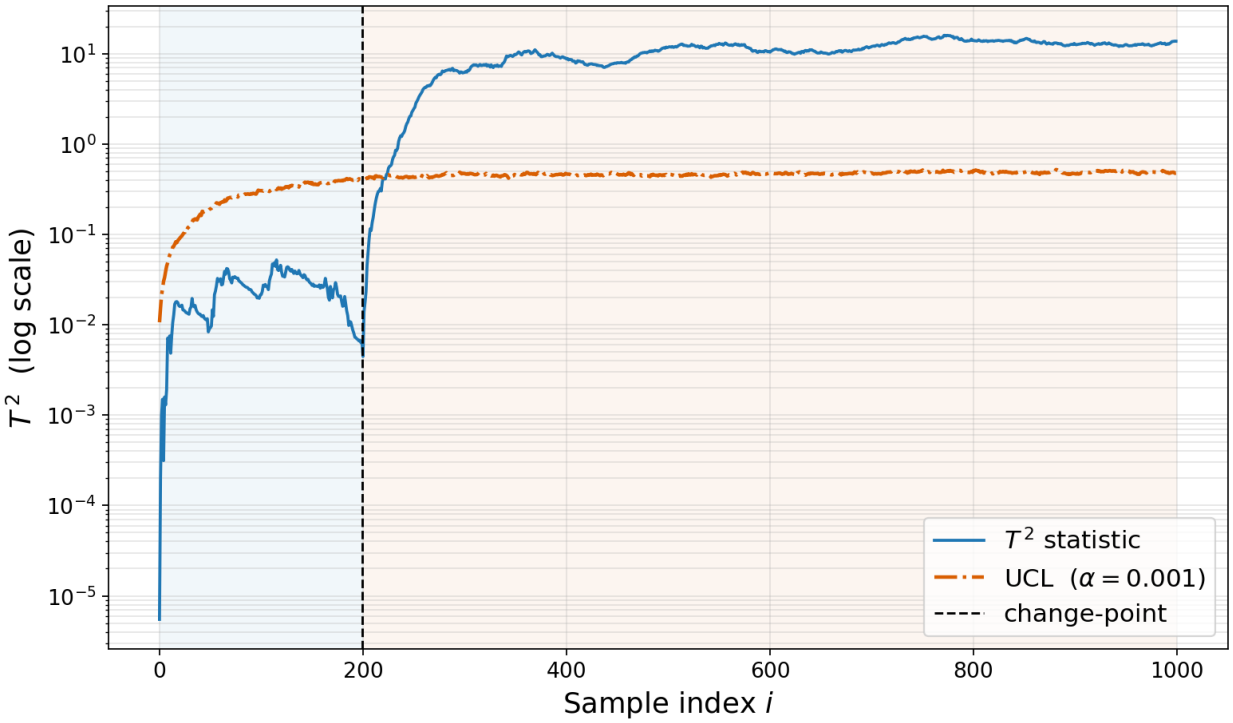
$B_I = 200$, $\alpha = 0.001$, and $\lambda = 0.01$. The subsequent monitoring stream contains $M = 1{,}000$ observations, of which the final 800 incorporate the prescribed parameter shift (4.7).

We consider two scenarios with different noise levels, the results for which are shown in Figure 4 for a typical replicate. In the low-noise scenario (Figure 4a; $\sigma = 0.03$), the fitted model has a CV R^2 of 0.928, and the T^2 statistic increases above the CL shortly after the distribution shifts at new observation 201. The T^2 statistic remains consistently above the CL after the shift, reflecting our method's ability to reliably detect subtle changes in the system dynamics. The high-noise scenario ($\sigma = 0.3$) represents the situation that the noise is so high that the model's predictive power becomes very poor (CV $R^2 = 0.108$). However, as seen in Fig. 4b, our approach still successfully detects the concept drift, although there are post-shift periods over which the T^2 statistic temporarily drops below the CL.

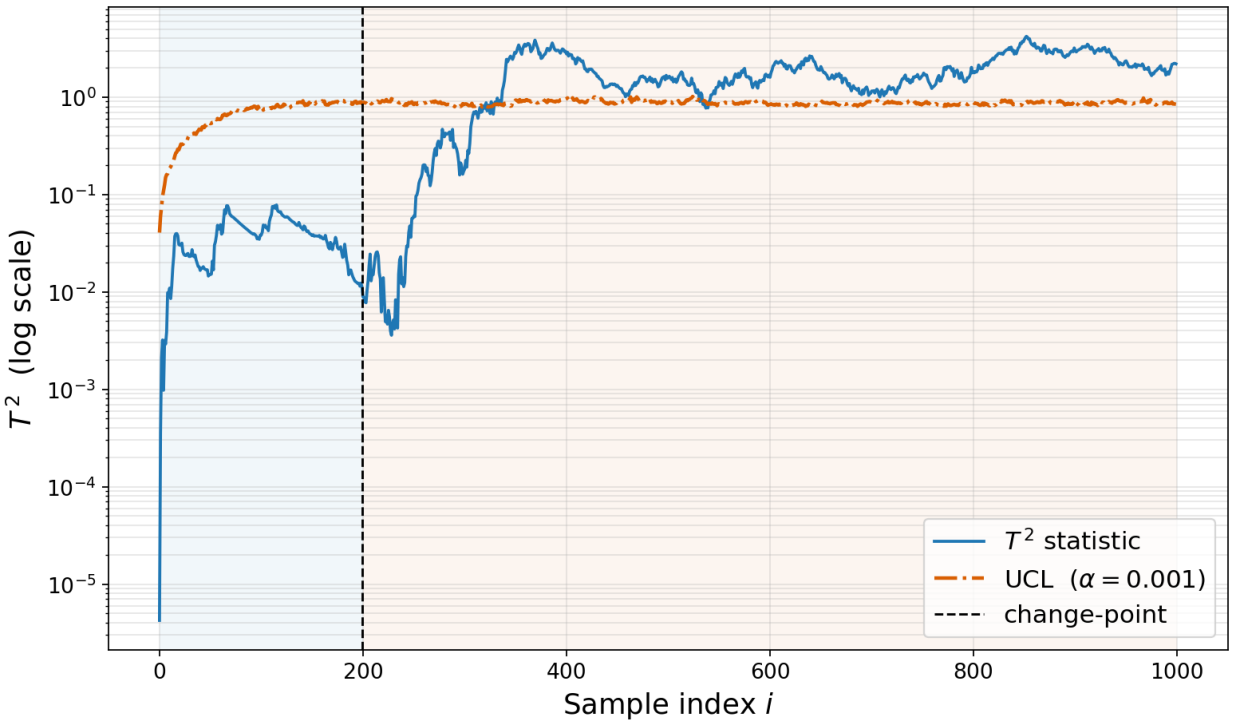
Although the neural network is only an approximation of the underlying dynamics of Eq. (4.4) (i.e., there is some level of model misspecification), the approach is still able to detect shifts in the predictive relationship. These results align with the arguments surrounding Eq. (2.4) and in [20] that the score vector mean can capture changes in the underlying predictive relationship $\mathbb{P}(Y|\mathbf{X})$ even with model misspecification. This feature is important in practice, since fitted machine learning models are always approximations of predictive relationships.

5 Concluding Remarks

In this paper, we present a nested bootstrapping procedure to compute a time-varying CL for the score-based MEWMA control chart of [20], for detecting changes in the predictive distribution $\mathbb{P}(Y|\mathbf{X})$ represented by a supervised learning model. Our approach provides much more accurate false-alarm rate control than the approach of [20], especially with smaller training sample size n and/or larger desired false-alarm rates, and also produces a time-varying CL to maintain accurate false-alarm rate control prior to the MEWMA reaching steady state. Our derivations involve a non-obvious 632-type bootstrap correction that is



(a) Low noise ($\sigma = 0.03$, 5-fold CV $R^2 \approx 0.928$)



(b) High noise ($\sigma = 0.3$, 5-fold CV $R^2 \approx 0.108$)

Figure 4: Typical monitoring results using our bootstrap CL approach for the coupled nonlinear oscillator system under (a) low-noise and (b) high-noise conditions. The shifts were introduced at $i = 201$ and are first detected at observations $i = 225$ and $i = 307$, respectively.

essential for controlling the false-alarm rate.

Our numerical illustrations, in which the predictive relationships are a linear mixture model and a more complex nonlinear dynamical system, demonstrate the method's ability to detect shifts while controlling pointwise false-alarm rates more accurately than the approach of [20]. These results highlight the practical utility of our approach in scenarios where detecting subtle changes in $\mathbb{P}(Y \mid \mathbf{X})$ is important.

Disclosure statement

The authors report there are no competing interests to declare.

References

[1] S. A. Abbasi, A. Yeganeh, and S. C. Shongwe. Monitoring non-parametric profiles using adaptive EWMA control chart. *Scientific Reports*, 12(14336), 2022.

[2] A. Apsemidis, S. Psarakis, and J. M. Moguerza. A review of machine learning kernel methods in statistical process monitoring. *Computers & Industrial Engineering*, 149:106776, 2020.

[3] M. Baena-García, J. del Campo-Avila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno. Early drift detection method. In *Proceedings of the 4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams*, pages 77–86. Springer, 2006.

[4] S. Barocas, M. Hardt, and A. Narayanan. *Fairness and Machine Learning: Limitations and Opportunities*. MIT Press, 2023.

[5] P. J. Bickel and K. A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics*, volume 1 of *Chapman & Hall/CRC Texts in Statistical Science*. CRC Press, Boca Raton, FL, 2015.

[6] S. I. Chang and S. Yadama. Statistical process control for monitoring non-linear profiles using wavelet filtering and B-spline approximation. *International Journal of Production Research*, 48(4):1049–1068, 2010.

[7] M. Febbo and S. P. Machado. Nonlinear dynamic vibration absorbers with a saturation. *Journal of Sound and Vibration*, 332(6):1465–1483, 2013.

[8] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):701–720, 2014.

[9] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Wozniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

[10] A. Malinovskaya, P. Mozharovskyi, and P. Otto. Statistical process monitoring of artificial neural networks. *Technometrics*, 66(1):104–117, 2024.

[11] D. C. Montgomery. *Introduction to Statistical Quality Control*. John Wiley & Sons, 8th edition, 2020.

[12] J. G. Moreno-Torres, T. Raeder, R. Alaíz-Rodríguez, N. Chawla, and F. Herrera. A unifying view on dataset shift in classification. *Pattern Recognit.*, 45:521–530, 2012.

[13] S. Psarakis. The use of neural networks in statistical process control charts. *Quality and Reliability Engineering International*, 27(5):641–650, 2011.

[14] A. M. S. Razak, C. R. Nirmala, B. R. Sreenivasa, H. Lahza, and H. F. M. Lahza. A survey on detecting healthcare concept drift in AI/ML models from a finance perspective. *Frontiers in Artificial Intelligence*, 5:955314, 2023.

[15] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33:191–198, 2012.

[16] J. Sun, H. Fujita, P. Chen, and H. Li. Dynamic financial distress prediction with concept drift based on time weighting combined with adaboost support vector machine ensemble. *Knowledge-Based Systems*, 120:4–14, 2017.

[17] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *KDD*, pages 226–235, 2003.

[18] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016.

[19] W. H. Woodall, D. J. Spitzner, D. C. Montgomery, and S. Gupta. Using control charts to monitor process and product quality profiles. *Journal of Quality Technology*, 36(3):309–320, 2004.

[20] K. Zhang, A. T. Bui, and D. W. Apley. Concept drift monitoring and diagnostics of supervised learning models via score vectors. *Technometrics*, 65(2):137–149, 2023.

[21] I. Žliobaitė, M. Pechenizkiy, and J. Gama. An overview of concept drift applications. *Big data analysis: new algorithms for a new society*, pages 91–114, 2016.