Data Mining and Knowledge Discovery (2021) 35:726–747
https://doi.org/10.1007/s10618-021-00739-7

Detecting virtual concept drift of regressors without ground truth values

Emilia Oikarinen[1] · **Henri Tiittanen**[1] · **Andreas Henelius**[1,2] · **Kai Puolamäki**[1,3]

Received: 15 May 2020 / Accepted: 15 January 2021 / Published online: 4 February 2021

Abstract
Regression analysis is a standard supervised machine learning method used to model an outcome variable in terms of a set of predictor variables. In most real-world applications the true value of the outcome variable we want to predict is unknown outside the training data, i.e., the ground truth is unknown. Phenomena such as overfitting and concept drift make it difficult to directly observe when the estimate from a model potentially is wrong. In this paper we present an efficient framework for estimating the generalization error of regression functions, applicable to any family of regression functions when the ground truth is unknown. We present a theoretical derivation of the framework and empirically evaluate its strengths and limitations. We find that it performs robustly and is useful for detecting concept drift in datasets in several real-world domains.

Keywords Concept drift · Generalization error · Unknown ground truth

1 Introduction

Regression models are one of the most used and studied machine learning primitives. They are used to model a dependent variable (denoted by $y \in \mathbb{R}$) given an m-dimensional vector of covariates (here we assume real valued attributes $x \in \mathbb{R}^m$). The regression model is trained using *training data* in such a way that it gives good

Responsible editor: Ira Assent, Carlotta Domeniconi, Aristides Gionis, Eyke Hüllermeier.

✉ Emilia Oikarinen
emilia.oikarinen@helsinki.fi

1 Department of Computer Science, University of Helsinki, Helsinki, Finland

2 OP Financial Group, Helsinki, Finland

3 Institute for Atmospheric and Earth System Research (INAR), University of Helsinki, Helsinki, Finland

estimates of the dependent variable y on *testing data* unseen in the training phase. In addition to estimating the value of the dependent variable, it is in practice important to know the *reliability of the estimate* on testing data. In this paper, we use the expected root mean square error (RMSE) between the dependent variable and its estimate to quantify the uncertainty, but some other error measure could be used as well. In textbooks, one finds a plethora of ways to train various regression models and to estimate uncertainties, see, e.g., Hastie et al. (2009). For example, for a Bayesian regression model the reliability of the estimate can be expressed in terms of the posterior distribution or, more simply, as a confidence interval around the estimate. Another alternative to assess the error of a regression estimate on unseen data is to use (cross-)validation. All of these approaches give some measure of the error on testing data, even when the dependent variable is unknown.

Textbook approaches are, however, *valid only when the training and testing data obey the same distribution*. In many practical applications this assumption does not hold: a phenomenon known as *concept drift* (Gama et al. 2014) occurs. Concept drift means that the distribution of the data changes over time, in which case the assumptions made by the regression model break down, resulting in regression estimates with *unknown and possibly large errors*. For example, in sensor calibration a regression model trained to model the sensor response may fail when the environmental conditions change from those used in training (Kadlec et al. 2011; Vergara et al. 2012; Rudnitskaya 2018; Maag et al. 2018; Huggard et al. 2018). Other examples arise, e.g., in online streaming data applications such as sentiment classification (Bifet and Frank 2010) and spam detection (Lindstrom et al. 2010).

In the simplest case, if the ground truth (the dependent variable y) is known, concept drift may be detected by observing the magnitude of the error, i.e., the difference between the regression estimate and the dependent variable. However, in practice, this is seldom possible. Indeed, a typical motive for using a regression model is that the value of the dependent variable is not readily available. In this paper, we address the problem of *assessing the regression error when the ground truth is unknown*, which, despite its significance, has not really been adequately addressed in the literature, see Sect. 2 for a discussion of related work. We do not focus on any particular application domain in this paper. Instead, our goal is to introduce a generic computational methodology that can be applied in a wide range of domains where regression is used.

Concept drift can be divided into two main categories: *real* and *virtual concept drift* (Gama et al. 2014). The former refers to the change in the conditional probability $p(y \mid x)$ and the latter to the change in the distribution of the covariates $p(x)$. If only the covariates x are known but the ground truth y is not, then it is not possible even in theory to detect changes occurring only in $p(y \mid x)$ and not in $p(x)$. However, it is possible to detect changes in $p(x)$ even when the values of y have not been observed; hence, we focus on the *detection of virtual concept drift* in this paper. Note, that one possible interpretation for a situation where $p(y \mid x)$ changes but $p(x)$ remains unchanged is that we are missing some covariates from x which would parametrize the changes in $p(y \mid x)$. Thus, an occurrence of real concept drift without virtual concept drift can indicate that not all necessary attributes are at our disposal. An obvious solution is then to include more attributes into the set of covariates. One should further observe, that when studying concept drift, we are not interested in detecting merely

any changes in the distribution of x. Rather, we are only interested in changes that likely increase the error of the regression estimates. This property is satisfied by our proposed method.

Contributions and organization In this paper we (i) define the problem of detecting the concept drift affecting the regression error when the ground truth is unknown, (ii) present an efficient algorithm to solve the problem for arbitrary (black-box) regression models, (iii) show theoretical properties of our solution, and (iv) present an empirical evaluation of our approach.

The rest of this paper is structured as follows. In Sect. 2, we review the related work. In Sect. 3, we introduce the idea behind our proposed method for detecting virtual concept drift, which is then formalized in the algorithm discussed in Sect. 4. We demonstrate different aspects of our method in the experimental evaluation in Sect. 5. Finally, we conclude with a discussion in Sect. 6.

2 Related work

The term *concept drift* was coined by Schlimmer and Granger (1986) to describe the phenomenon where the data distribution changes over time in dynamic and non-stationary environments. The research related to concept drift has become popular over the last decades with many real world applications, see, e.g., the recent surveys (Gama et al. 2014; Žliobaite et al. 2016; Lu et al. 2019). Concept drift detection methods can be divided into *supervised* (requiring ground truth values) and *unsupervised* approaches. Our approach falls into the latter category, and we focus on reviewing the unsupervised approaches. Furthermore, most concept drift literature focuses on classification and concept drift adaptation problems, while our focus is on concept drift in *regression problems*.

One of the few concept drift detection methods for regression is proposed by Wang et al. (2017), where an ensemble of multiple regression models trained on subsets of the data is used to find the best weighting for combining their predictions, and concept drift is defined as the angle between the estimated weight and mean weight vectors. While there are similarities to our method, i.e., subsets of data are used to train several regressors, the fundamental difference is that ground truth values are required in the method by Wang et al. (2017).

The unsupervised approaches can be roughly divided into two categories: those detecting purely distributional changes, e.g., Dasu et al. (2006), Shao et al. (2014) and Qahtan et al. (2015), and those taking the model into account in some way, e.g., Sethi and Kantardzic (2017), Lindstrom et al. (2013) and Sobolewski and Wozniak (2013). Approaches directly monitoring the covariate distribution $p(x)$ detect all changes in $p(x)$ regardless of their effect on the performance of the model. However, when detecting concept drift that degrades the performance of the model, these approaches suffer from a high false alarm rate (Sethi and Kantardzic 2017).

The approaches taking the model into account are typically not generic, but, e.g., require a classifier with a meaningful notion of margin (Sethi and Kantardzic 2017) or a score interpretable as an estimate of the confidence of the correctness of the prediction (Lindstrom et al. 2013). The MD3 method (Sethi and Kantardzic 2017)

uses classifier margin densities for concept drift detection, hence, requiring a classifier with some meaningful notion of margin, e.g., a probabilistic classifier or a Support Vector Machine (SVM). The method works by dividing the input data into segments, and for each segment the proportion of samples in the margin ρ is computed. The minimum and maximum values of ρ are monitored, and if their difference exceeds a given threshold, concept drift is declared. Lindstrom et al. (2013) calculate a stream of indicator values using the Kullback–Leibler divergence to compare the histogram of classifier output confidence scores on a test window to a reference window. If a certain proportion of previous indicator values are above a threshold, concept drift is declared. The method is not generic, however, since it requires a classifier producing a score that can be interpreted as an estimate of the confidence associated with the correctness of the prediction.

Since probabilistic regression models provide direct information of the model behavior in the form of uncertainty estimates, it is straightforward to implement a concept drift detection measure by thresholding the uncertainty estimate, e.g., Chandola and Vatsavai (2011) present a method based on Gaussian processes for time series change detection. Sobolewski and Wozniak (2013) develop a concept drift detection method especially for data containing recurring concepts. Hence, they require prior knowledge about properties of concepts present in the data (namely the samples residing in the centers or at the borders of the class clusters). Then, a distinct classification model is trained for each concept, and for each test data segment the closest concept in the training data is selected using a non-parametric statistical test.

Generalization error is a central concept in statistical learning theory and in the study of empirical processes. One of the key insights relevant to our work are the symmetrization lemmas; see, e.g., Mohri and Medina (2012) and Kuznetsov and Mohri (2017). Our Theorem 1 follows these ideas, where we estimate generalization loss, i.e., the difference between the regression estimate and the unknown ground truth, by using the differences of estimates given by regressors trained on separate samples of data. Our work differs from these earlier more theoretical approaches by the fact that we wish to provide a practical method by which concept drift can be detected for off-the-self regression functions. We can therefore give a theoretical intuition that applies in a special case and show that our method works in practice with experiments.

In time series regression the objective is to estimate a variable of interest given a set of covariates and/or lagged (past) measurements; see, e.g., Hyndman and Athanasopoulos (2018). In this paper, we focus on the straightforward task of building regression functions of covariates on time series data. We do not attempt to predict future values using past values, even though in principle we could append the most recent data values as *additional covariates* to the data, nor do we explicitly control autocorrelation, seasonality, or trends; instead, we assume that this is taken care of by the used (black-box) regression model. Our objective is solely to find whether the regression error on the test data is likely to exceed a given threshold. We use the time series nature of the data only when we assume that the samples within a temporal segment are more likely to be from the same distribution.

3 Methods

Let the *training data* D_{tr} consist of n_{tr} triplets: $D_{\text{tr}} = \{(i, x_i, y_i)\}_{i=1}^{n_{\text{tr}}}$, where $i \in [n_{\text{tr}}] = \{1, \ldots, n_{\text{tr}}\}$ is the time index, $x_i \in \mathbb{R}^m$ are the covariates and $y_i \in \mathbb{R}$ is the dependent variable. Also, let the *testing data* similarly be given by $D_{\text{te}} = \{(i, x'_i, y'_i)\}_{i=1}^{n_{\text{te}}}$ where $i \in [n_{\text{te}}]$, and the covariates and the dependent variable are given by $x'_i \in \mathbb{R}^m$ and $y'_i \in \mathbb{R}$, respectively. Furthermore, let the *reduced testing data* be the testing data without the dependent variable, i.e., $D'_{\text{te}} = \{(i, x'_i)\}_{i=1}^{n_{\text{te}}}$. *Segments* of the data are defined by tuples $s = (a, b)$ where a and b are the endpoints of the segment such that $a \leq b$. We write $D_{|s}$ to denote the triplets in $D = \{(i, x_i, y_i)\}_{i=1}^{n}$ such that the time index i belongs to the segment s, i.e., $D_{|s} = \{(i, x_i, y_i) \mid a \leq i \leq b\}$.

Assume that we are given a *regression function* $f : \mathbb{R}^m \to \mathbb{R}$ trained using D_{tr}. The function f estimates the value of the dependent variable at time i given the covariates, i.e., $y'_i \approx \hat{y}'_i = f(x'_i)$. The *generalization error* of f on the data set $D = \{(i, x'_i, y'_i)\}_{i=1}^{n}$ is defined as

$$\text{RMSE}(f, D) = \left(\sum_{i=1}^{n} \left[f(x'_i) - y'_i \right]^2 / n \right)^{1/2}, \tag{1}$$

i.e., we consider the root mean squared error and formulate the following problem.

Problem 1 Given a regression function f trained using the dataset D_{tr}, and a threshold σ, predict whether the generalization error E of f on the testing data D as defined by Eq. (1) satisfies $E \geq \sigma$ when only the reduced testing data D' is known and the true dependent variable y'_i, $i \in [n]$, is unknown.

As discussed in the introduction, without the ground truth we can only detect virtual concept drift that occurs as a consequence of changes in the covariate distribution $p(x)$. We therefore need a distance measure $d(x)$ indicating how "far" a vector x is from the data D_{tr} used to train the regressor. Small values of $d(x)$ (later called the *concept drift indicator* value) mean that we are close to the training data and the regression estimates should be reliable, while large values of $d(x)$ mean that we have moved away from the training data and regression accuracy may be degraded.

We can now list some properties of a good distance measure. On one hand, we are only interested in the changes in the covariate distribution $p(x)$ that may affect the behavior of the regression. If there is an attribute not used by the regressor, then changes in the distribution of that attribute alone should be irrelevant. On the other hand, if a changed, i.e., drifted, attribute is important for the output of the regressor, then its fluctuations may cause concept drift and the value of $d(x)$ should be large.

We propose to define this distance measure as follows. We first train different regression functions, say f and f', on different subsets of the training data. We then define the distance measure to be the difference between the predictions of these two functions, e.g., $d(x) = [f(x) - f'(x)]^2$. The details of how we select the subsets and compute the difference are given later in Sect. 4. We can immediately observe that this distance measure has the suitable property that if some attributes are independent of the dependent variable, then they will not affect the behavior of the regression functions

and, hence, the distance measure d is insensitive to them. Next, we show that at least in the case of a simple linear model, the resulting measure is, in fact, monotonically related to the expected quadratic error of the regression function.

3.1 Theoretical motivation

In this section, we show that our method can be used to approximate the ground truth error for an ordinary least squares (OLS) linear regression model. Assume that our covariates $x_i \in \mathbb{R}^m$ have been sampled from some distribution for which the expected values and the covariance matrix exist with the first term being the intercept, or $x_{i1} = 1$. Hence, we rule out, e.g., the Cauchy distribution for which the expected value and variance are undefined. Given the parameter vector $\beta \in \mathbb{R}^m$ and the variance σ_y^2, the dependent variable is given by $y_i = \beta^T x_i + \epsilon_i$, where ϵ_i are independent random variables with zero mean and variance of σ_y^2.

Now, assume that we have trained an OLS linear regression model, parametrized by $\hat{\beta}$, on a dataset of size n and obtained a linear model f, and that we have also trained a different linear model on an independently sampled dataset of size n' and obtained a linear model f' parametrized by $\hat{\beta}'$, respectively. For a given x, the estimates of the dependent variable y are then given by $\hat{y} = f(x) = \hat{\beta^T} x$ and $\hat{y}' = f'(x) = \hat{\beta}'^T x$, respectively. We now prove the following theorem.

Theorem 1 *Given the definitions above, the expected mean squared error, i.e.,* $E\left[(f(x) - y)^2\right]$*, is monotonically related to the expectation of the squared difference between the two regressors* f *and* f'*, i.e.,* $E\left[(f(x) - f'(x))^2\right]$*, by the following equation to a leading order in* n^{-1} *and* n'^{-1}*:*

$$E\left[(f(x) - y)^2\right] = (1 + n/n')^{-1} E\left[(f(x) - f'(x))^2\right] + \sigma_y^2. \tag{2}$$

Proof First, we observe that Eq. (2) and specifically the values of the terms $\beta^T x_i$ remain unchanged for arbitrary translations and rotations of the covariate vectors x_i if the parameter vector β is adjusted appropriately. More specifically, translations and rotations are absorbed in the parameter β as follows. Translations a_j of coordinates $j \neq 1$ are defined by a constant shift $x_{ij} \leftarrow x_{ij} - a_j$ and are absorbed by redefining $\beta_1 \leftarrow \beta_1 + \sum_{j=2}^m \beta_j a_j$; recall that $x_{i1} = 1$ is the constant intercept term. A rotation of the data by an orthogonal matrix $U \in \mathbb{R}^{m \times m}$ $x_i \leftarrow U x_i$ can on the other hand be absorbed by redefining $\beta \leftarrow U\beta$. We can therefore, without loss of generality and to simplify the proof below, assume that the distribution from which the covariates have been sampled has been centered so that all terms except the intercept have an expectation of zero, or $x_{i1} = 1$ and $E[x_{ij}] = 0$ for all $j \neq 1$. We can further assume that the axes of the covariates have been rotated so that they are uncorrelated and satisfy $E\left[x_{ij} x_{ik}\right] = \sigma_{xj}^2 \delta_{jk}$, where the Kronecker delta satisfies $\delta_{jk} = 1$ if $j = k$ and $\delta_{jk} = 0$ otherwise and with $\sigma_{x1} = 0$.

For a dataset of size n, the OLS estimate of β, denoted by $\hat{\beta}$, is a random variable that obeys a distribution with a mean of β and a covariance given by $n^{-1}\Sigma$, where

$$\Sigma = \sigma_y^2 \text{diag}(1, \sigma_{x2}^{-2}, \ldots, \sigma_{xm}^{-2}) + \mathcal{O}(n^{-1}), \tag{3}$$

where the terms of the order n^{-1} or smaller have been included in $\mathcal{O}(n^{-1})$. The covariance $n^{-1}\Sigma$ is therefore proportional to n^{-1} and hence, at the limit of a large dataset we obtain the correct linear model, i.e., $\lim_{n\to\infty} \hat{\beta} = \beta$. For finite data there is always an error in the estimate of $\hat{\beta}$. The expected estimation error is larger for small data, i.e., if n is small.

It follows from Eq. (3) that the expected mean squared error for a model evaluated at x is given by

$$E\left[(f(x) - y)^2\right] = x^T \left(n^{-1}\Sigma\right) x + \sigma_y^2, \tag{4}$$

and the expected quadratic difference between the linear model estimates is given by

$$E\left[(f(x) - f'(x))^2\right] = x^T \left[\left(n^{-1} + n'^{-1}\right) \Sigma\right] x. \tag{5}$$

Equation (2) then follows by solving $x^T \Sigma x$ from Eq. (5) and inserting it in Eq. (4). □

We hence postulate that the squared differences between the estimates given by regressors trained on different subsets of the data—either sampled randomly or obtained by other means—can be used to estimate the mean squared error even when the ground truth y is unknown. Of course, in most interesting cases the regression functions are not linear, but as we show later in Sect. 5, the idea works also for real datasets and complex non-linear regression models. Our claim is therefore that the difference between the estimates of regressors trained on different subsets of the data in the point x defines a distance function which can be evaluated even when the ground truth is unknown. If a data point x is close to the data points used to train the regressors the distance should be small. On the other hand, if the data point is far away from the data used to train the regressors, the predictions of the regressors diverge and the distance and also the prediction error will be larger.

4 The Drifter algorithm

In this section we describe our algorithm, called `drifter`, for detecting concept drift when the ground truth is unknown. We start with the general idea using a simple data set shown in Fig. 1 as an example. We then provide the algorithmic details of the training and testing phases of our algorithm, and discuss how to select a suitable value for the drift detection threshold in practice.

By Theorem 1, we can estimate the generalization error using the terms $[f(x) - f'(x)]^2$ instead of the terms $[f(x) - y]^2$, where f' is another regressor. Our approach to obtain a suitable f' is to train several regression functions, called *segment models*

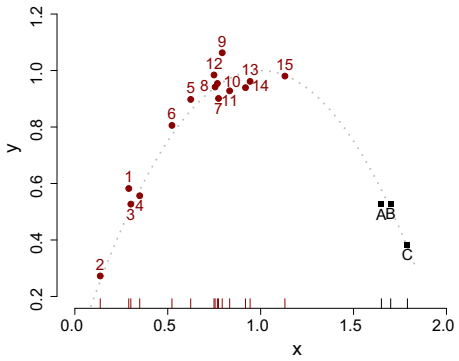


Fig. 1 Example data set with covariate x and response variable y. The training data D_{15} is shown with filled circles labelled with numbers and the testing data D_{ABC} with filled squared labelled with letters

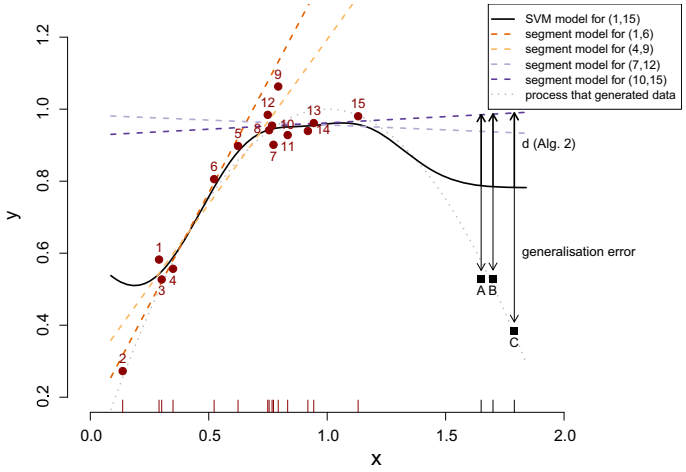


Fig. 2 The models trained using D_{15} (solid black line), and using four segments of D_{15} (lines with long dashes) (Color figure online)

using subsequences of the data (i.e., segments). Recall that segments are defined as continuous intervals of time indices. We call a distribution of covariates in a segment a *concept*. We assume here that *due to autocorrelation, a segment is more likely to contain samples from the same distribution of covariates (i.e., concept)* than, e.g., a subset of data sampled at random. Further, we assume that the *segment models trained on autocorrelated data provide a piecewise approximation of the full function*.

Note that if the underlying assumption that the data in a segment comes from the same distribution is violated the `drifter` algorithm may exhibit suboptimal performance. For this reason it is important to validate the choice of model parameters, such as segment lengths and the types of segment models, for any new dataset and regression model.

To illustrate the idea, let us consider $D_{15} = \{(i, x_i, y_i)\}_{i=1}^{15}$ of 15 data points, with the one-dimensional covariate x_i and response y_i, as shown in Fig. 1, and assume that D_{15} has been used to train a Support Vector Machine (SVM) regressor f. The SVM model estimate of y is shown with a black solid line in Fig. 2. Our testing data D_{ABC}

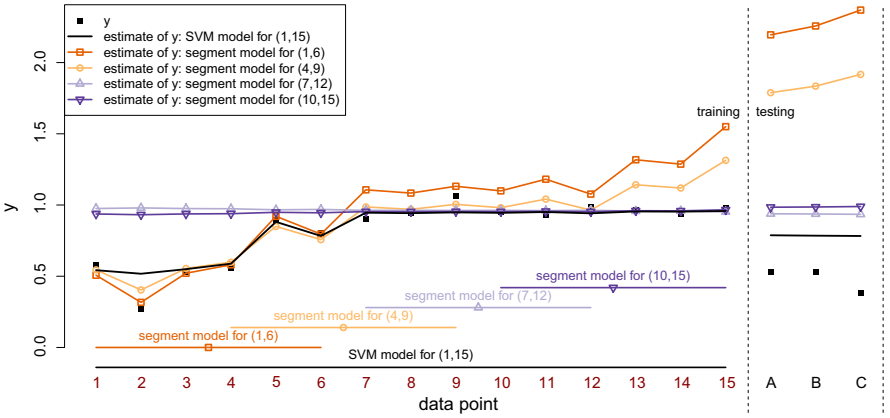


Fig. 3 The response variable y and the estimates of y using different models for the training data D_{15} and the testing data D_{ABC}

then consists of the data points labeled with A, B, and C in Fig. 1, and we want to estimate the generalization error of f, when we only have access to the covariates of D_{ABC}.

We can consider, e.g., the overlapping segments $s_1 = (1, 6)$, $s_2 = (4, 9)$, $s_3 = (7, 12)$, and $s_4 = (10, 15)$, and train the segment models using OLS regression. The choice of overlapping segments aims for robustness, i.e., we assume it is unlikely that the overlapping segmentation splits very clear concepts such that they would not appear in any of the segments. The linear segment models are shown in Fig. 2 using colored dashed lines, and the estimates are shown in Fig. 3 using the same colors. We observe that the segment models are good estimates for the SVM model on their respective training segments.

Now, we can compute an estimate of the generalization error with the terms $[f(x) - f_i(x)]^2$ instead of $[f(x) - y]^2$ for each segment model $f_i \in \{f_1, \ldots, f_4\}$. This then allows us to compute some statistics based on the estimates from this ensemble of segment models. Here, we choose the statistics, namely the *concept drift indicator* value, to be the *second smallest error*. The intuition is that if the test data resembled some concept in the training data, and an overlapping segmentation scheme was used, at least two of the segment models should provide a reasonably small indicator value. If there exists only a single small indicator value, it could well be due to chance, and using the second smallest value as the indicator value increases the robustness of the method. For our example, f_4 trained using the segment $(10, 15)$ is the second-best linear model for D_{ABC}. In Fig. 2 we visualize the terms $[f(x) - f_4(x)]^2$ for D_{ABC} using black vertical arrows. Our estimate for the generalization error of f in D_{ABC} is large even for the second-best linear model f_4 and we conclude that concept drift in D_{ABC} is indeed likely.

4.1 The `drifter` algorithm

Now, we are ready to formalize the ideas presented above, and describe in detail the `drifter` algorithm consisting of the (i) *training* and (ii) *testing* phases.

(i) Training phase In the training phase of `drifter` (Algorithm 1), we train the segment models for the subsequences, i.e., segments, of the training data. As input, we assume the training data $D_{\text{tr}} = \{(i, x_i, y_i)\}_{i=1}^{n_{\text{tr}}}$, a segmentation S of $[n_{\text{tr}}]$, and a function `tr_f` for training segment models.

Input : $D_{\text{tr}} = \{(i, x_i, y_i)\}_{i=1}^{n_{\text{tr}}}$,
segmentation S,
training function `tr_f`
Output: array F of segment models
1 **Function** `dr_tr(`D_{tr}, S, `tr_f)`
2 $(s_1, \ldots, s_k) \leftarrow S$
3 **for** $i \in [k]$ **do**
4 $f_i \leftarrow$ `tr_f(`$D_{\text{tr}|s_i}$`)`
5 **return** $F \leftarrow (f_1, \ldots, f_k)$

Algorithm 1: `drifter` training

Input : $D'_{\text{te}} = \{(j, x'_j)\}_{j=1}^{n_{\text{te}}}$, model $f : \mathbb{R}^m \mapsto \mathbb{R}$,
segment models F, integer n_{ind}
Output: indicator value d
1 **Function** `dr_te(`D'_{te}, f, F, n_{ind}`)`
2 $(f_1, \ldots, f_k) \leftarrow F$
3 **for** $i \in [k]$ **do**
4 $z_i \leftarrow \text{RMSE}^*(f, f_i, D'_{\text{te}})$
5 **return** $d \leftarrow$ `sort_inc(`$z_1, \ldots, z_k$`)`$[n_{\text{ind}}]$

Algorithm 2: `drifter` testing

Hence, we assume that the user provides a segmentation S of $[n_{\text{tr}}]$ such that when the segment models are trained, the data used to train a model *approximately* corresponds to only one *concept*, i.e., the models "specialize" in different concepts. Here there might, of course, be overlap so that multiple models are trained using the same concept. We show in Sect. 5 that using a scheme in which the segmentation consists of equally-sized segments of length l_{tr} with 50% overlap, the `drifter` method is quite robust with respect to the choice of l_{tr}, i.e., just selecting a reasonably small segment length l_{tr} generally makes the method perform well and provides a simple baseline approach for selecting a segmentation. However, the segments could well be of varying length or non-overlapping. For instance, by using a segmentation that is a solution to the basis segmentation problem (Bingham et al. 2006), one would know that each segment can be approximated with linear combinations of the basis vectors.

The training phase essentially consists of training a regression function f_i for each segment $s_i \in S$ using $D_{\text{tr}|s_i}$ (lines 3–4 in Algorithm 1). These regression functions are the *segment models*. Note, that the model family of the segment models is chosen by the user and provided as input to Algorithm 1. Natural choices are, e.g., linear regression models or, if known, functions from the same model family as f used in the testing phase.

(ii) Testing phase The tester function of `drifter` (Algorithm 2) takes as input the testing data D'_{te}, the model f, the segment models F from Algorithm 1, and an integer n_{ind} (*indicator index order*). For each of the k segment models f_i, we then determine the RMSE between the predictions from f_i and f on the test data (lines 3–4), i.e.,

$$\text{RMSE}^*(f, f_i, D'_{\text{te}}) = \left(\sum_{j=1}^{n_{\text{te}}} \left[f(x'_j) - f_i(x'_j) \right]^2 / n_{\text{te}} \right)^{1/2}, \tag{6}$$

where $D'_{\text{te}} = \{(j, x'_j)\}_{j=1}^{n_{\text{te}}}$. This gives us k values z_i (line 4) estimating the generalization error, and we then choose the n_{ind}th smallest value as the *concept drift indicator* value d (line 5). If this value is large, then the predictions from the full model on the test data in question can be unreliable.

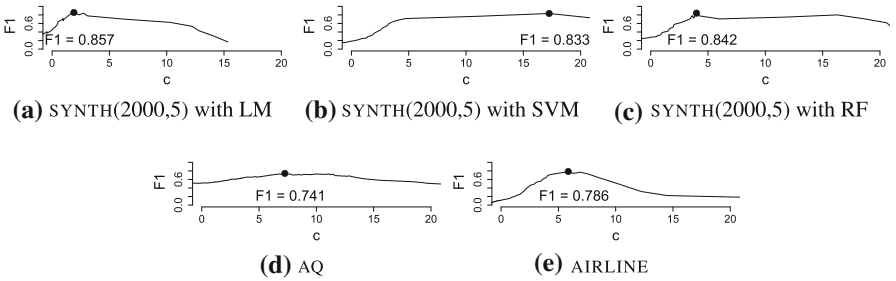
In this paper we use $n_{\text{ind}} = 2$ by default. The intuition behind this choice is that, due to the overlapping segmentation scheme we use, it is reasonable to assume that at least two of the segment models should have small values for z_i's, if the testing data has no concept drift, while a single small value for z_i could still occur by chance even in the presence of concept drift. Other types of data may require a different n_{ind} value.

In the testing phase, there is an implicit assumption that $n_{\text{te}} \leq l_{\text{tr}}$ should hold, where l_{tr} is the length of a segment in the training phase, i.e., the testing data can be covered by a single segment. This is due to the assumption that the segment models are trained to model concepts present in the training data. If $n_{\text{te}} \gg l_{\text{tr}}$, the testing data might consist of several concepts, resulting in a large value for the concept drift indicator value d, implying concept drift even if none was present. This can be easily prevented, e.g., as done in the experimental evaluation in Sect. 5, by dividing the testing data into smaller (non-overlapping) *test segments* of length $l_{\text{te}} \leq l_{\text{tr}}$ and applying the tester function (Algorithm 2) on each of the test segments. In this way we thus obtain a concept drift indicator value for each smaller segment in the testing data.

4.2 Selection of the drift detection threshold

To solve Problem 1, we still need a threshold for the concept drift indicator value d (Algorithm 2) that estimates the threshold for the generalization error in Problem 1. A good concept drift detection threshold δ depends both on the dataset and the application in question. A validation set D_{val} with known ground truth values (not used in the training of f) could be used to compute the generalization error $\text{RMSE}(f, D_{\text{val}})$ and determine a suitable threshold δ, e.g., using *receiver operating characteristics* (ROC) analysis, which makes it possible to balance the number of false positives and false negatives (Fawcett 2006). However, one needs to assume that there is no concept drift in D_{val}, and consider using, e.g., cross-validation when training f to prevent overfitting.

We propose here a general method for obtaining a threshold δ using only the training data, which according to our empirical evaluation (see Sect. 5) performs well in practice for the datasets used in this paper. However, a user knowledgeable of a particular data



(a) SYNTH(2000,5) with LM **(b)** SYNTH(2000,5) with SVM **(c)** SYNTH(2000,5) with RF

(d) AQ **(e)** AIRLINE

Fig. 4 The effect of the multiplier constant c for δ in Eq. (7), c_{opt} from Table 1 shown with the corresponding *F1*-score

and an application can use this knowledge to select and potentially adjust a better threshold for the data and the application in question.

For computing the threshold δ, we first split the training dataset into $\lfloor n_{\text{tr}}/n_{\text{te}} \rfloor$ (non-overlapping) segments of the same length as the testing data. Next we compute the concept drift indicator value d_i for each of these segments s_i in the training data $D_{\text{tr}|s_i}$ using Algorithm 2, i.e., $d_i = \texttt{dr\_te}(D_{\text{tr}|s_i}, f, F, n_{\text{ind}})$. We then choose a concept drift detection threshold δ by using the mean and standard deviation of the indicator values of these segments

$$\delta = \text{mean}(d_i) + c \times \text{sd}(d_i), \tag{7}$$

where c is a *constant* multiplier of choice. The optimal value of c depends on the properties of a particular dataset, but our empirical evaluation (see Fig. 4 in Sect. 5) shows that the performance with respect to the *F1*-score is not overly sensitive with respect to the choice of c (e.g., $c = 5$ works well for all datasets we used here).

4.3 Using `drifter` to solve Problem 1

We now summarize, how the `drifter` method is used in practice to solve Problem 1.

Assume that a model f has been trained using $D_{\text{tr}} = \{(i, x_i, y_i)\}_{i=1}^{n_{\text{tr}}}$, and we know that the *concept length* in the training data is approximately l_{tr}. We use this knowledge to form a segmentation S of $[n_{\text{tr}}]$ such that there are k segments of length l_{tr}. We also need to choose the model family of the segment models (function `tr_f`). In practice, linear regression models seem to consistently perform well (see Sect. 5). Then, the *training phase* consists of a call to Algorithm 1 to obtain an ensemble $F = (f_1, \ldots, f_k)$ of segment models.

Once the segment models have been trained, we can readily use these to detect concept drift in the testing data $D'_{\text{te}} = \{(i, x_i)\}_{i=1}^{n_{\text{te}}}$. In the *testing phase*, we should apply Algorithm 2 on a testing data with $n_{\text{te}} \leq l_{\text{tr}}$, where l_{tr} is the segment length used to train the segment models F in Algorithm 1. In practice, this is achieved by splitting the testing data into small segments of length l_{te} (e.g., we use constant $l_{\text{te}} = 15$ in Sect. 5) and applying Algorithm 2 individually on each small test segment.

If we have split the testing data into k' segments, and obtained a vector of concept drift indicator values $(d_1, \ldots, d_{k'})$ using Algorithm 2, we can then compare these values to the concept drift detection threshold δ, which is either user-specified or obtained using the approach described in Sect. 4.2, and classify each segment in the testing data, either as a segment exhibiting concept drift ($d_i \geq \delta$) or not ($d_i < \delta$).

The time complexity of `drifter` is dominated by the training phase, where we need to train k regressors using data of size $\mathcal{O}(n_{\text{tr}}/k)$ and dimensionality m. For OLS regression, the complexity of training one segment model is hence $\mathcal{O}\left(n_{\text{tr}}m^2/k\right)$ and the complexity of the training phase is $\mathcal{O}\left(nm^2\right)$.

5 Experiments

In this section we experimentally evaluate `drifter` in the detection of concept drift. We first present the datasets and regressors used. Then, we discuss the generalization error and default parameters used in the experiments in Sect. 5.1. In Sect. 5.2 we pin down suitable combinations of the remaining parameters of `drifter`. In Sect. 5.3 we assess the runtime scalability of `drifter` on synthetic data, and finally in Sect. 5.4 we look at how `drifter` finds concept drift on our considered dataset and regression function combinations. The experiments were run using R (v. 3.5.3) on a high-performance cluster (FCGI 2019): 2 cores from an Intel Xeon E5-2680 2.4 GHz with 256 Gb RAM. An implementation of the `drifter` algorithm and the code for the experiments presented in this paper has been released as open-source software (Tiittanen et al. 2019).

Datasets and regressors We use the datasets described below. During preprocessing we removed rows with missing values, and transformed factors into numerical values. For each dataset, we then use the first 50% of the data as the training set, and the remaining 50% as a testing dataset, i.e., $n_{\text{tr}} = \lfloor 0.5n \rfloor$ and $n_{\text{te}} = \lceil 0.5n \rceil$. We split the testing data into non-overlapping *test segments* of length $l_{\text{te}} = 15$.

Air quality data ($n = 7355$, $m = 11$) The AQ dataset (Vito et al. 2008) contains hourly air quality sensor measurements spanning approximately 1 year. We trained a regressor for hourly averaged concentrations of carbon monoxide CO(GT) using Support Vector Machine (SVM) from the '*e1071*' R package with default parameters.

Flight delay data ($n = 38042$, $m = 84$) The AIRLINE dataset collected by U.S. Department of Transportation (2017) contains data related to flight delays. We used arrival delay as the target variable and a subset of the other attributes as covariates. To keep computation time manageable we used every 150th sample and trained a regressor for the arrival delay using Random Forest (RF) from the '*randomForest*' R package with default parameters.

Bike rental data ($n = 731$, $m = 8$) The BIKE dataset (Fanaee-T and Gama 2014) contains daily counts of bike rentals and covariates related to weather and date types for a period of about 2 years. Exploratory analysis indicated *real concept drift* in the form of an increasing trend in the counts of bike rentals. Thus, we prepared an alternative version of the data by removing the trend by multiplying

each $y'_i \in D_{\text{te}}$ by $m = \text{mean}_{y \in D_{\text{tr}}}(y)/\text{mean}_{y' \in D_{\text{te}}}(y')$. In the dataset BIKE(RAW) we use the original rental counts y_i, whereas in the dataset BIKE(DETR) we use the modified rental counts $m \times y_i$. We trained OLS linear regression models (LM) for predicting the rental counts for BIKE(RAW) and BIKE(DETR).

Synthetic data We constructed the SYNTH(n,m) datasets as follows. The covariate matrix $X \in \mathbb{R}^{n \times m}$ is sampled columnwise from AR(1) with correlation length $h = 150$, defined as the number of steps after which the expected autocorrelation drops to 0.5, and the amplitude $amp = 1$. The elements of a noise vector $e \in \mathbb{R}^n$ are sampled from a normal distribution $N(0, \sigma_N^2)$, where $\sigma_N = 0.3$. The target variable is $Y = g(X^T) + e$, where $g = \sin$ is used to introduce non-linearity. In Sect. 5.3, we vary the data dimensions n and m when generating datasets SYNTH(n,m), and otherwise use the dataset SYNTH(2000,5) with a concept drift component at [1700, 1800] added by using $amp = 5$ during this period. We trained LM, RF, and SVM regressors with the SYNTH data.

5.1 Generalization error threshold and parameters of `drifter`

The datasets we use do not have predefined ground truth values and we hence first need to define what constitutes concept drift in the test datasets. The user should choose the threshold σ: in some applications a larger generalization error could be tolerated, while in some other applications the user might want to be alerted already about smaller errors. In the absence of a user, we determined the error threshold σ_{emp} for the datasets as follows. We used 5-fold cross-validation, where we randomly split the training data into five folds, and estimated the value of the ith dependent variable y_i by a regressor trained on the four folds that do not contain i, thereby obtaining a vector of estimates $\hat{y}_i$ for all $i \in [n_{\text{tr}}]$. We then computed the generalization error for the training data as in Eq. (1) and then chose $\sigma_{\text{emp}} = 2 \times (\sum_{i=1}^{n_{\text{tr}}} \left(\hat{y}_i - y_i\right)^2 / n_{\text{tr}})^{1/2}$. All values exceeding σ_{emp} in the test dataset are consequently considered concept drift. While this cross-validation procedure does not fully account for possible autocorrelation in the training data we found that in our datasets it gives a reasonable estimate of the generalization error in the absence of concept drift.

To find a suitable value for c for selecting the detection threshold δ (Sect. 4.2) and to assess how well the scheme works in practice, we compute the "optimal" detection threshold δ_{opt} in terms of the $F1$-score for a given error threshold σ_{emp} as follows. We vary the concept drift detection threshold δ and evaluate the true and false positive rates on the test dataset, allowing us to form a ROC curve. We then pick the δ_{opt} maximizing the $F1$-score, i.e., $F1 = 2TP/(2TP + FP + FN)$, where TP are true positives, FP are false positives and FN are false negatives. For the other parameters, we use in the training phase the segmentation scheme with 50% overlap between consecutive segments. In the testing phase, we split the testing data into non-overlapping segments of fixed length ($l_{\text{te}} = 15$), and evaluate the concept drift indicator value on each test segment using $n_{\text{ind}} = 2$. In preliminary experiments, we also tested a segmentation scheme with no overlap between segments in the training phase, and values $n_{\text{ind}} \in \{1, 2, 3, 5\}$. The effect of these parameter options was rather

small in practice, and we chose the values for which the `drifter` method performed most robustly in detecting virtual concept drift for our datasets.

5.2 Effect of concept length, segment models, and drift detection threshold

We next investigate the effects of the remaining input parameters, i.e., (i) the constant c in Eq. (7), (ii) the concept length (i.e., the segment length l_{tr} in the training phase), and (iii) the effect of the model family for the segment models.

We varied $k = \lfloor n_{\text{tr}}/l_{\text{tr}} \rfloor$, which means that there are $2k - 1$ segments in the training phase in the overlapping segmentation scheme. The maximum value for k was determined by the requirement $l_{\text{tr}} \geq l_{\text{te}} = 15$. For each k and each dataset, we determined the value for c_{opt} that leads to δ_{opt} in Eq. (7) maximizing the *F1*-score.

For the model family, we considered two choices: either the segment models were trained using the same model family as the model f given as input, or linear regression was used for the segment models. Our evaluation showed that the linear segment models consistently performed the best (both in terms of performance, e.g., *F1*-score, robustness, and computational cost, i.e., time needed to train the models). We hence focus on linear regression models as segment models in the rest of this paper. Observe that there is an intuitive reason why LM outperforms SVM and RF as segment models. While SVM and RF give accurate predictions on the training data covariate distribution, they predict constant values outside of it. The linear OLS regressor on the other hand gives (non-constant) linearly increasing/decreasing predictions the farther from the training data covariate distribution the testing data is. It should also be noted that for SVM the kernel choice makes a difference in terms of generalization behavior. We used here a radial basis function kernel, but if a polynomial kernel or a linear kernel were used, the model would behave more like LM.

The results are presented in Table 1 showing the *number of segments* of length $l_{\text{te}} = 15$ in the testing data identified as true (*TP*) and false (*FP*) positives, and true (*TN*) and false (*FN*) negatives, respectively. We observe, that concept drift is detected with a reasonable accuracy for the AQ, AIRLINE and SYNTH(2000,5) datasets in terms of the *F1*-score, i.e., the number of true positives and negatives is high, while the number of false positives and negatives remains low. For each of these datasets we have identified the best performing combination of k and c (shown with bold in Table 1), and we subsequently use these particular combinations in Sect. 5.4.

BIKE(RAW) has *real concept drift in the data*, i.e., the bike rental counts are higher during the second year likely due to increasing popularity of the service. Since real concept drift does not affect the concept drift indicator values, we observe a high number of false negatives. Note that this is as expected, because any algorithm without access to the ground truth values cannot detect real concept drift. When considering the BIKE(DETR) dataset where the real concept drift has been removed, no concept drift is observed (hence we cannot compute the *F1*-scores). Our algorithm correctly handles this, i.e., all the segments in the testing data are classified as true negatives. For the values of BIKE(RAW) and BIKE(DETR) in Table 1 we have used δ larger than the maximal value of d (similarly as in Sect. 5.4 and Fig. 8c, d).

Table 1 The effect of segment length l_{tr}, using $k = \lfloor n_{\text{tr}}/l_{\text{tr}} \rfloor$, on drift detection accuracy in terms of the *F1*-score

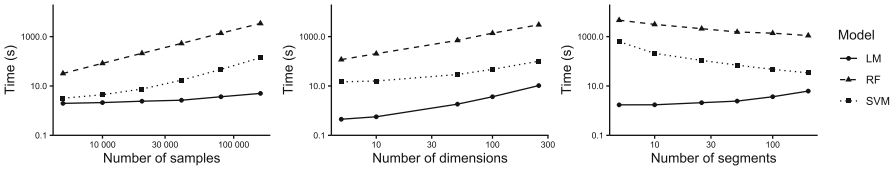
Data	FM	SM	k	c_{opt}	F1	TP	FP	TN	FN
AQ	SVM	LM	2	6.770	0.735	61	20	139	24
			10	**7.144**	**0.741**	63	22	137	22
			20	7.080	0.737	56	11	148	29
			100	5.835	0.688	54	18	141	31
AIRLINE	RF	LM	**2**	**5.632**	**0.786**	11	1	1250	5
			10	5.695	0.786	11	1	1250	5
			20	5.769	0.786	11	1	1250	5
			100	5.424	0.769	10	0	1251	6
BIKE(RAW)	LM	LM	2, 4, 6	–	–	0	0	5	18
BIKE(DETR)	LM	LM	2, 4, 6	–	–	0	0	23	0
SYNTH(2000,5)	LM	LM	2	5.571	0.737	7	1	54	4
			10	4.722	0.778	7	0	55	4
			60	**1.747**	**0.857**	9	1	54	2
SYNTH(2000,5)	SVM	LM	2	6.930	0.769	5	2	58	1
			10	8.817	0.769	5	2	58	1
			60	**17.015**	**0.833**	5	1	59	1
SYNTH(2000,5)	RF	LM	2	5.819	0.750	6	1	56	3
			10	9.649	0.778	7	2	55	2
			60	**3.883**	**0.842**	8	2	55	1

Here, FM and SM stand for the used full and segment model family, respectively, c_{opt} is the value using which Eq. (7) results in δ_{opt} maximizing the *F1*-score, TP (resp. FP) is the count of true positives (resp. false positives), and TN (resp. FN) is the count of true negatives (resp. false negatives)

Finally, we investigate how varying the value of the drift detection threshold c affects the performance of the `drifter` algorithm. For AQ, AIRLINE, and SYNTH(2000,5) datasets we used the fixed parameter values as defined in Sect. 5.1 and selected the concept length based on the previous experiment, i.e., we used k resulting in the best performance in terms of the *F1*-score using the optimal c (shown in bold in Table 1). We excluded the BIKE datasets here, since they do not contain virtual concept drift, which makes the relation between the *F1*-score and c less informative. The results are presented in Fig. 4. The performance of our method is quite insensitive to the value of c, and $c = 5$ seems to be a robust choice for the datasets considered.

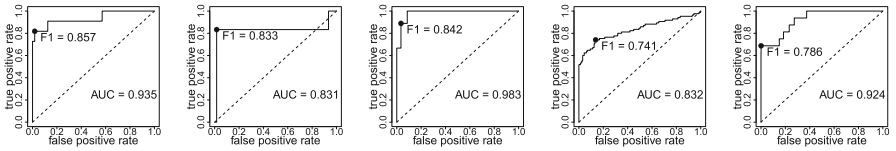
5.3 Scalability

The scalability was studied using the SYNTH(n,m) data, by varying the length $n \in \{5000, 10{,}000, 20{,}000, 40{,}000, 80{,}000, 160{,}000\}$ of training data, the data dimensionality $m \in \{5, 10, 50, 100, 250\}$, and the parameter $k \in \{5, 10, 25, 50, 100, 200\}$ controlling the number of segments (and hence, l_{tr}). We used SYNTH(n,m) with $amp = 1$ as the training data and for testing data we used SYNTH(15,m) with $amp = 5$.



(a) Varying n. **(b)** Varying m. **(c)** Varying k.

Fig. 5 Scalability of the `drifter` algorithm in the training phase using SYNTH(n,m). In each figure, one of the parameters is varied, while the remaining ones are kept constant ($n = 80{,}000$, $m = 100$, and $k = 100$)



(a) SYNTH, LM **(b)** SYNTH, SVM **(c)** SYNTH, RF **(d)** AQ **(e)** AIRLINE

Fig. 6 ROC-curves (Fawcett 2006) for **a**–**c** SYNTH(2000,5), **d** AQ, and **e** AIRLINE datasets

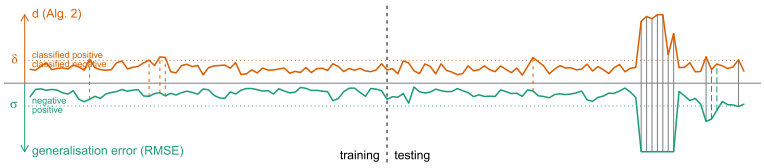
We used the first 500 samples of the training data to train an SVM regressor f, and varied the choice for the model family (LM, SVM, RF) used by `drifter` in training the segment models. The median running time of the training phase of `drifter` over five runs are shown in Fig. 5 (the respective running times for the testing phase are negligible). We observe that in particular when using OLS regression to train the segment models, the `drifter` algorithm is fast for reasonable dataset sizes.

5.4 Detection of concept drift

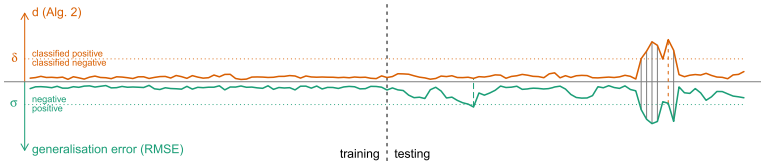
Finally, we consider examples illustrating how the `drifter` method works in practice. In Figs. 7 and 8 , we show the generalization error (green lines) and the concept drift indicator value d (orange line), and in Fig. 6 the ROC-curves for the datasets considered. For SYNTH(2000,5), AIRLINE, and AQ we used the parameters bolded in Table 1, and for BIKE(RAW) and BIKE(DETR) we used $k = 4$ and selected δ to be larger than the maximal value of d.

For SYNTH(2000,5) (Fig. 7) we observe that our algorithm correctly detects the virtual concept drift introduced during the period [1700, 1800]. For AQ (Fig. 8a) we observe that a significant amount of the testing data seems to exhibit concept drift, and our algorithm detects this. There is a rather natural explanation for this. The AQ data contains measurements of a period of 1 year. The model f_{AQ} has been trained on data covering the spring and summer months (March to August), while the testing period consists of the autumn and winter months (September to February). Hence, it is natural that the testing data contains concepts not present in the training data. Also observe that the last segments of data again begin to resemble the training data, and hence we do not observe concept drift in these segments.

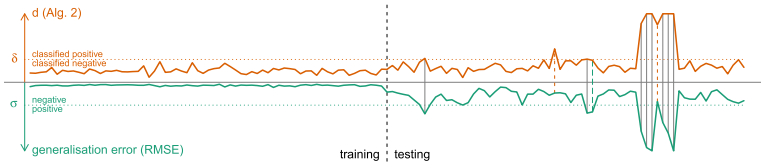
For AIRLINE, also some of the segments in the training data have a rather high value for the generalization error, indicating that there are parts of the training data that the



(a) SYNTH (LM)



(b) SYNTH (SVM)



(c) SYNTH (RF)

Fig. 7 The generalization error and concept drift indicator d for test segments of length $l_{\text{te}} = 15$ in the SYNTH(2000,5) dataset. Here, δ denotes the concept drift detection threshold and σ denotes the generalization error threshold. The vertical lines between the curves indicate the segments that are true positives (gray solid line), false positives (orange dashed line), or false negatives (green longdash line) (Color figure online)

regressor f_{AL} does not model well. However, the concept drift indicator d behaves similarly to RMSE (both for segments in the training and testing data), demonstrating that it can be used to estimate when the generalization error would be high.

For BIKE(RAW) (Fig. 8c) we observe that even though the generalization error is large for most of the segments in the testing data, the drift detection indicator does not indicate concept drift. This is explained by the *real concept drift* present in data, and once we have removed it in the BIKE(DETR) data (Fig. 8d) we observe no concept drift. We hence observe that a considerable number of false negatives can indicate real concept drift in the data. However, in order to detect this, one needs to have access to the ground truth values.

6 Discussion

In this paper, we have presented and evaluated an efficient method for detecting concept drift in regression models when the ground truth is unknown. We here define concept drift as a phenomenon causing larger than expected estimation errors on new data, as a result of changes in the generating distribution of the data. Defining concept drift this way instead of considering all changes in the distribution, makes it possible to detect only the changes that actually affect the prediction quality. Thus, if concept

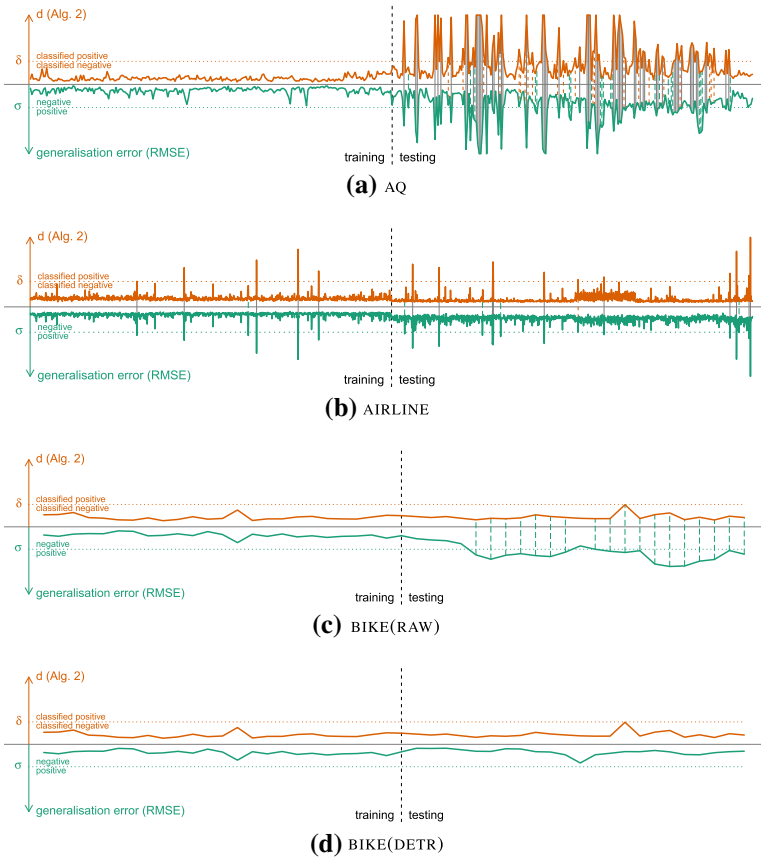


Fig. 8 The generalization error and concept drift indicator d for test segments of length $l_{\text{te}} = 15$ in the AQ, AIRLINE, and BIKE datasets. Here, δ denotes the concept drift detection threshold and σ denotes the generalization error threshold. The vertical lines between the curves indicate the segments that are true positives (gray solid line), false positives (orange dashed line), or false negatives (green longdash line) (Color figure online)

drift detection is used to monitor the performance of a regression model, the false positive rate is reduced. It is surprising how little attention this problem has received, considering its importance in multiple domains.

When the dependent variable y is unknown it is only possible to detect changes in the distribution of the covariates $p(x)$. Our idea is to use the regression functions themselves to study the changes in this distribution. As we have shown for linear models in Theorem 1, we postulate that if we train two or more regression functions on different subsets of the data, then the difference in the estimates given by the regression functions contains information about the generalization error. This method is powerful, while also being simple and straightforward to implement. It, e.g., by design ignores features of the data that are irrelevant for estimating y. The underlying assumption is that by using subsets of the training data we can train regressors that can capture *concepts* in the data, and if the testing data contains concepts not found

in the training data, then it is likely that there is concept drift. The `drifter` method presented in this paper also scales well. Especially high performance is reached using OLS linear segment models.

The main limitation of any concept drift detection method that works without the ground truth information—including ours—is that it is only possible to detect virtual concept drift. We cannot even in princple detect concept drift, if the distribution of the covariates $p(x)$ remains unchanged. Another underlying assumption in our method is that the segment models describe different concepts in the data, after which the differences in predictions between the segment models gives us the concept drift indicator. For this to work the segments should be long enough so that the segment models do not overfit the data. Also, we need the segment models to be different, which can fail, e.g., if the segments are too long and the segment models end up modeling the distribution of the training data instead of individual concepts. In general, the parameters of the `drifter` should be chosen and validated for each dataset and problem separately, because the best choices for segment lengths and threshold value depend on the dataset properties and the requirements of the practical application (e.g., how large errors should be tolerated before warning about potential concept drift).

We have used models trained using different segments of the data in this paper. An interesting topic for future work is to study how the data could be "optimally" partitioned for this problem. In our examples linear segment models had the best performance. The reason may be that their predictions diverge the further away from the training data we go, while the SVM and RF regression models used in this paper predict constant values far away from the training data. For the SVM and RF models an alternative concept drift indicator could be that the predictions remain constant under small perturbations of the covariate vector. Another alternative—which we have experimented with but not reported here—is to train several regression models from different model families for the whole data, instead of using segment models. We have also focused on estimating the generalization error of a regression function. The same ideas could be applied to detect concept drift in classifiers as well.

The theoretical foundation for this approach is shown to hold in the simple case of linear regression. However, our empirical evaluation with real datasets of various types (and different regressors) demonstrates that the idea also works when there are sources of non-linearity. Our experiments suggest that often the (black-box) regressor given as input can be locally approximated using linear regressors, and the differences between the estimates from these regressors serve as good indicators for concept drift. The current paper represents initial work towards a practical concept drift detection algorithm, with experimental evaluation illustrating parameters that work robustly for the datasets considered in this work. Further work is needed to establish general practices for selecting suitable parameters for `drifter`; even though we can give general guidance of sensible values of the parameters in `drifter` it is still important to validate the parameters of the model for new data sets and regression models.

Acknowledgements We thank Dr Martha Zaidan for help and discussions. This work was funded by the Academy of Finland (decisions 326280 and 326339). We acknowledge the computational resources provided by Finnish Grid and Cloud Infrastructure (FCGI 2019).

Funding Open Access funding provided by University of Helsinki including Helsinki University Central Hospital

References

Bifet A, Frank E (2010) Sentiment knowledge discovery in twitter streaming data. In: Proceedings of 13th international conference on discovery science DS 2010. Springer, LNAI, vol 6332, pp 1–15

Bingham E, Gionis A, Haiminen N, Hiisilä H, Mannila H, Terzi E (2006) Segmentation and dimensionality reduction. In: Proceedings of the 2006 SIAM international conference on data mining, SIAM, pp 372–383

Chandola V, Vatsavai RR (2011) A Gaussian process based online change detection algorithm for monitoring periodic time series. In: Proceedings of the 11th SIAM international conference on data mining, SDM, SIAM, pp 95–106

Dasu T, Krishnan S, Venkatasubramanian S, Yi K (2006) An information-theoretic approach to detecting changes in multi-dimensional data streams. In: Proceedings of symposium on the interface of statistics, computing science, and applications INTERFACE

Fanaee-T H, Gama J (2014) Event labeling combining ensemble detectors and background knowledge. Prog Artif Intell 2(2):113–127

Fawcett T (2006) An introduction to ROC analysis. Pattern Recognit Lett 27(8):861–874

FCGI (2019) Finnish Grid and Cloud Infrastructure. Urn:nbn:fi:research-infras-2016072533

Gama J, Žliobaite I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. ACM Comput Surv 46(4):44:1–44:37

Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference, and prediction. Springer, Berlin

Huggard H, Koh YS, Riddle P, Olivares G (2018) Predicting air quality from low-cost sensor measurements. In: Proceedings of Australasian conference on data mining AusDM 2018, Springer, CCIS, vol 996, pp 94–106

Hyndman RJ, Athanasopoulos G (2018) Forecasting: principles and practice, 2nd edn. OTexts. https://otexts.com/fpp2/. Accessed 15 May 2020

Kadlec P, Grbić R, Gabrys B (2011) Review of adaptation mechanisms for data-driven soft sensors. Comput Chem Eng 35:1–24

Kuznetsov V, Mohri M (2017) Generalization bounds for non-stationary mixing processes. Mach Learn 106:93–117

Lindstrom P, Delany SJ, Mac Namee B (2010) Handling concept drift in a text data stream constrained by high labelling cost. In: Proceedings to the 23rd international FLAIRS conference, pp 32–37

Lindstrom P, Namee BM, Delany SJ (2013) Drift detection using uncertainty distribution divergence. Evol Syst 4(1):13–25

Lu J, Liu A, Dong F, Gu F, Gama J, Zhang G (2019) Learning under concept drift: a review. IEEE Trans Knowl Data Eng 31(12):2346–2363

Maag B, Zhou Z, Thiele L (2018) A survey on sensor calibration in air pollution monitoring deployments. IEEE Internet Things J 5:4857–4870

Mohri M, Medina AM (2012) New analysis and algorithm for learning with drifting distributions. In: Algorithmic learning theory. ALT 2012. Springer, LNCS, vol 7568

Qahtan AA, Alharbi B, Wang S, Zhang X (2015) A PCA-based change detection framework for multi-dimensional data streams. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 935–944

Rudnitskaya A (2018) Calibration update and drift correction for electronic noses and tongues. Front Chem 6:433
Schlimmer JC, Granger RH (1986) Incremental learning from noisy data. Mach Learn 1(3):317–354
Sethi TS, Kantardzic M (2017) On the reliable detection of concept drift from streaming unlabeled data. Expert Syst Appl 82:77–99
Shao J, Ahmadi Z, Kramer S (2014) Prototype-based learning on concept-drifting data streams. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 412–421
Sobolewski P, Wozniak M (2013) Concept drift detection and model selection with simulated recurrence and ensembles of statistical detectors. J Univ Comput Sci 19(4):462–483
Tiittanen H, Oikarinen E, Henelius A, Puolamäki K (2019) Drifter. https://github.com/edahelsinki/drifter. Accessed 15 May 2020
US Department of Transportation (2017) 2015 Flight Delays and Cancellations. https://www.kaggle.com/usdot/flight-delays. Accessed 15 May 2020
Vergara A, Vembu S, Ayhan T, Ryan MA, LHomer M, Huerta R (2012) Chemical gas sensor drift compensation using classifier ensembles. Sens Actuators B Chem 166–167:320–329
Vito SD, Massera E, Piga M, Martinotto L, Francia GD (2008) On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. Sens Actuators B Chem 129(2):750–757
Wang LY, Park C, Yeon K, Choi H (2017) Tracking concept drift using a constrained penalized regression combiner. Comput Stat Data Anal 108:52–69
Žliobaite I, Pechenizkiy M, Gama J (2016) An overview of concept drift applications. In: Japkowicz N, Stefanowski J (eds) Big data analysis: new algorithms for a new society. Springer, Cham, pp 91–114