Optimizing the training of time series forecasting models: A change point detection approach

Pantelis Chronis*†, Giorgos Giannopoulos*, Spiros Athanasiou*, Spiros Skiadopoulos†
*Athena Research Center,†University of Peloponnese

***Abstract*—A plethora of machine learning algorithms have been proposed for modelling time series data and handling several sub-tasks, such as forecasting, pattern recognition, event/anomaly detection and clustering. Most machine learning algorithms applied on time series data assume that the true underlying model, from which the time series is generated, remains the same throughout the dataset. For cases where this assumption does not hold, in order to model the time series, there are methods that partition the time-series by identifying change points (change point detection or segmentation methods) and methods that integrate the process of identifying changes in the model within the algorithm (e.g. State Space Models).**

The effect that a change in the underlying model has to the performance of the machine learning algorithm depends on the properties of the time series, the properties of the change and the algorithm. In this paper, we argue that the training of a machine learning model on a time series can be favoured by considering the changes in the underlying model *with respect to the generalization error* of the machine learning model to be trained. To this end, we propose an algorithm that, given a machine learning model, scans the time series backwards and identifies a proper point in time from which on to include samples in the training set of the model. The criterion for the selection of this optimal point is the minimization of the generalisation error of the trained machine learning model.

The proposed method can be applied on top of different machine learning algorithms. In this paper, we demonstrate the effectiveness of our method by applying it with three machine learning algorithms for time series forecasting: Support Vector Regression, Elastic-net Regularized Linear Regression and Neural Network. We compare our proposed method with three baselines: a naive one that trains the machine learning model on the whole available dataset and two change point detection based baselines which identify the optimal point in time by applying state of the art change point detection methods. Our method outperforms all baselines in each experimental configuration, for every one of the three assessed machine learning algorithms, by achieving significantly higher forecasting precision.

I. Introduction

Time series data mining is an extensive field with applications ranging in several scientific areas, including medicine, economics, astronomy, energy, water demand, etc. Several problems related to these fields involve the batch or real-time analysis of time series data produced by measurements of a quantity in (usually) fixed time intervals. The analysis may aim to identify the future values of the time series (forecasting), to identify interesting patterns (pattern recognition), to group together time series or parts of them with similar behaviour (clustering), to detect changes/events/anomalies in a time series that might indicate a specific real-world cause (event detection). A large number of approaches and algorithms that have been proposed in the past cover the aforementioned time series analysis tasks.

Further, it is often the case that auxiliary methods have to be applied that increase the effectiveness of machine learning algorithms. Indicatively, time series smoothing [17] is applied to adjust oscillations of different magnitude through the time series values; trend, cycle and seasonality removal approaches [14] remove predictable, temporal changes of the time series so that the analysis is performed on the remaining components; methods such as dynamic time warping [18] are applied to

Algorithm 1: Obtain Activity Zones

Data: TimeSeries Ts
Result: ActivityZones $Zones$
$Patterns$ = IdentifySignificantPatternsForEachDay(Ts)
N=IdentifyNumberOfPattersPerDay($Patterns$)
$Hours$ = CountPatternPeaksPerHourOfDay($Patterns$)
$Zones = \emptyset$
for i *from* 1 *to* N **do**
  $Peak$=SelectHourWithMaxPatterns($Hours$)
  $Zone$=FindZoneAboveThreshold($Hours$,$Peak$)
  AddNewZone($Zones$,$Zone$)
  RemoveZoneFromHours($Hours$)
end

Algorithm 2: Train Model for Activity Zones Prediction

Data: TimeSeries Ts, ActivityZones $Zones$
Result: ModelForActivityZones $Model$
$Thresh$ = FindThresholdsForActivity(Ts,$Zones$)
$ActTs$ = ObtainActivityTimeSeries($Thresh$,Ts)
$Model$ = InitEmptyModel()
forall the *ActivityZones* $Az \in Zones$ **do**
  $ActModel$ = TrainModelForActivity($ActTs$,Az)
  AddToModel($ActModel$,$Model$)
end

Algorithm 3: Forecast Water Consumption

Data: BaselineSvrForecast $SvrPred$, TimeSeries Ts, ModelForActivityZones $Model$
Result: Prediction $Pred$
$AzPred$ = ForecastActivityZones($Model$,Ts)
$Pred$ = ModifySvrPrediction($SvrPred$,$AzPred$)

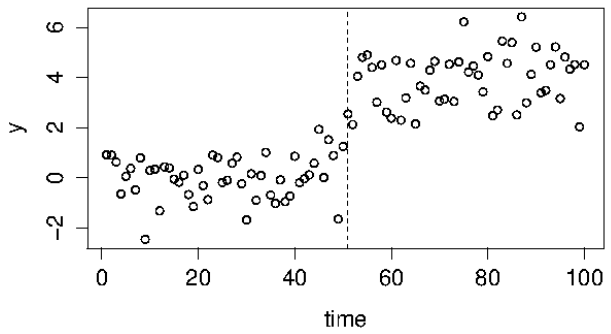

Fig. 1. An example of a time series where the underlying model changes at time 50

align time series so that potentially undesirable time shifts between two time series are handled.

Algorithms used for time series modelling assume that the time series is generated from an underlying model, which they attempt to estimate. In cases where that underlying model changes at some point in time (for example Figure 1), the algorithm needs to identify the change and account for it. There are several methods that handle that problem, either by dividing the time series and training different models on its parts or by utilizing more complex models that can incorporate the change. Which of those two approaches is more suitable depends on the specific characteristics of the problem. The choice of a more complex model is suitable for cases where the changes are smooth and predictable, while the segmentation/change point detection approach is suitable for more sharp and unpredictable changes in the underlying model. In our analysis, we do not consider cycle, trend or seasonality detection, since there are several orthogonal methods that effectively handle such components of time series, when they exist.

In this paper we propose an algorithm for identifying the optimal point to partition the time series, in order to get the most accurate estimation of the *current state* of the underlying model. Our algorithm considers a training dataset of past time series observations and a machine learning model that is to be trained on the dataset. It starts from the current point in time, t_n and scans the time series backwards in order to identify the first historical point, t_h where a change in the underlying model is detected. The change is detected by examining the variations in the error of the model, while adding previous samples in the training set. If the change is considered significant, the algorithm isolates the sub-time series $\{t_h, ..., t_n\}$ and provides *only* this specific part as training input for the machine learning model to be trained.

The novelty of our approach is that it selects the change point that optimizes the generalisation performance of the learned model, which it evaluates through the use of a validation set. Our argument is that simply identifying a change in the underlying model is not sufficient for increasing the effectiveness of the learned model, that is trained on the "changed" part of the time series. In order to optimize the effectiveness, one has to take into account the effect that the change has to the machine learning model.

For example, a complex model would require a large amount of samples to be adequately trained. If we detect a change in the time series, that could in fact influence the model, but its effect is not very severe, it may not be beneficial to discard all the observations prior to the change, until we have enough samples to confidently estimate the new state of the time series. A simpler model on the other hand, could be trained from a smaller dataset. In this case it could be beneficial to discard the samples prior to the change as soon as the change is detected. The optimal choice depends on the magnitude of the change in the underlying model and the complexity of the trained model/algorithm. Our method accounts for the effect that each change has to the learned model, by examining the model's generalisation error.

The paper is organized as follows. In Section II we review the literature on methods that address changes in time series models. In Section III we first provide an overview of our method and then we formally define the problem and describe our method in detail. In Section IV we demonstrate the effectiveness of our method used on top of three machine learning algorithms and compared to three baseline methods. Finally, Section V concludes the paper.

II. Related Work

The problem of time series modelling and forecasting has been extensively studied and numerous different approaches and models can be found in the literature. The state of the art for time series forecasting includes ARIMA models, Exponential Smoothing, Artificial Neural Networks and Support Vector Machines, among others [13], [14], [15], [12], [**?**]. In the case of time series of water consumption on individual households, which is the dataset we use in our study, evidence in the literature suggests that Linear Models and Support Vector Machine algorithms perform best [16].

The problem of change point detection has also received a lot of attention [3]. There are two main approaches that appear in the literature. In the statistical approach ([1], [2], [4], [7], [8], [10]), a series of hypotheses related to model change are evaluated, using some statistical test, in order to divide the time series in separate parts. In this line of work [1] use the Mann-Whitney test that detects differences in population means, to identify points where the level of the time series shifts. [7] extends this work to also capture changes in the variance of the time series. The other approach addresses the issue of a changing model as an optimization problem ([5], [6], [9]). In [6] the authors present a framework where they divide the time series by minimizing the residual error of models trained on the resulting parts. Similarly, [9] define the change points in the time series in order to minimize the residual error of a linear model, while penalizing the frequent changes in the model to avoid over-fitting.

The novelty of our approach is that it selects the change point so that the resulting model has the lowest *generalisation error*, for the current state of the time series. This is a desired feature, particularly in cases where change points are detected for the purpose of forecasting.

III. Optimizing Model Training

The proposed algorithm optimizes the training process of machine learning models, on time series where the underlying model may change though time. Specifically, it identifies the best point in time, from which on it is beneficial to include samples in the training-set. In this section, we first provide the overview of the method, by intuitively explaining it through a running example. Next, we provide the formal problem definition and a detailed analysis of our proposed algorithm. Finally, we discuss the parameters of our method and the effect we expect them to have on its effectiveness.

A. Method Overview

The time series shown in Figure 2 depicts daily measurements of water consumption from a single household. In our setting, we want to build a model that predicts the behaviour of this time series, based on some features that are available (the consumption of the previous day, the day of week etc.). We also suppose we are at day 205, which means that we have knowledge of days $1-205$ (training set) but not $206-365$, and we want to predict the consumption for the next 15 days, i.e. the interval $206-220$. The most simple approach would be to select a proper machine learning algorithm and train it on the entire training set to create a model that describes the time series. We know, however, that changes may occur in a household (e.g. vacations, changes in work schedule etc.). Those changes influence water consumption, therefore the patterns of water consumption may be different for different periods in time, requiring, thus, different models to describe the time series. So, if we have no external/explicit information about changes in the household, how do we know if such changes exist? Further, how do we train our model so that it captures the patterns of the current state but it is not skewed by patterns from previous periods, that correspond to different consumption models?

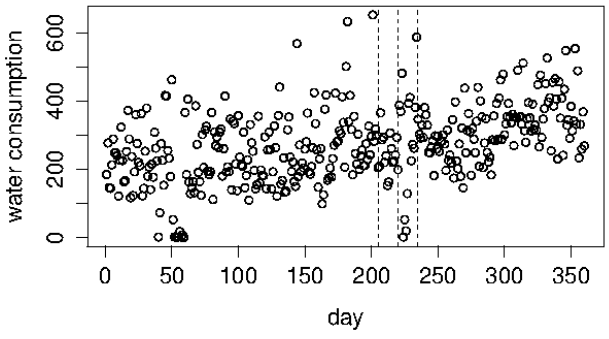

Fig. 2. Time Series of daily water consumption. The dashed lines at 205,220 and 235 define the two intervals of prediction described in Section III .

To see into these issues, we begin from day 205 (current time), and go backwards in time, iteratively including samples into the (existing) training set. Each time we add a new point, we consider a new training set, train a model on it and measure the prediction error, for days $206-220$. In Figure 3, we can see the prediction error for each of these iterations (where in x-axis we vary the starting day of the training set and the end day is fixed at day 205). We clearly see that the minimum error does not appear when the entire dataset is used (i.e. $x = 1$); instead it appears around $x = 115$. This implies that at around day 115 a change may have occurred in the household that affected the patterns of water consumption. So, in this case, by reducing the training set, we achieve a better predicting performance for the model.

In Figure 4 we can see the prediction error on the interval $221-235$ for different starting days of the training set. We observe that, if we select day 110, that minimizes the error on the interval $205-220$ (Fig. 3), we will also achieve a very significant improvement in the prediction error for the interval $221-235$. That means that the behaviour of the time series in the interval $205-220$ is very similar to the behaviour in $221-235$, in this case. So if we are at day 220 and want to predict days $221-235$, we can use the error for the interval $221-235$ to find a good start for the training set of our model.

Our algorithm, in order to find the best start for the training set, iteratively adds one previous sample to the training set and calculates the error on a validation set, that is selected to be as recent as possible, and, thus, as close as possible to the current state of the time series. The use of a validation set is based on the assumption of *temporal locality*, which essentially prescribes that the behaviour of the time series in the near future will likely be similar to the behaviour in the recent past, like we observed in the example above. For example, if at some point in time an inhabitant of the household leaves for a trip, since we have no information regarding his return, the property of temporal locality suggests that we should consider he will be absent, until we observe he has returned. Temporal locality is a natural assumption for time series and is necessary when the changes in the underlying model can not be predicted. Through the use of a validation set we evaluate the generalisation error of the model. Therefore we can select the training set that results in the model with the best forecasting performance.

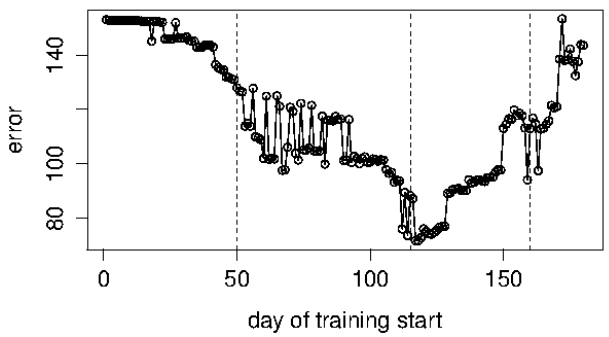

Fig. 3. The sequence of errors (MAE) for different starts of the training set. The value at $x = i$ is the error of the model trained on samples from i to 205. The error is measured on the interval 206-220

B. Problem Formalization

We consider time series Y, for which we have measurements up to the current time t: $Y = \{y_i, 1 \leq i \leq t\}$, where i is the index of time. For each y_i we have a vector of associated

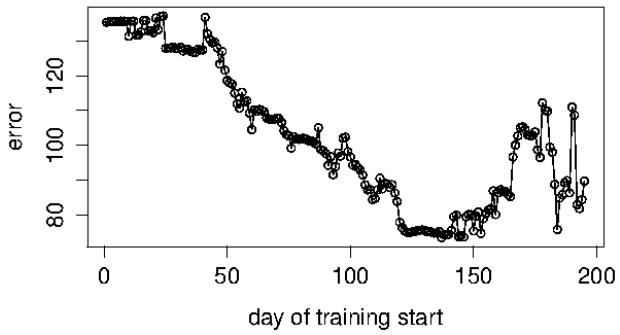

Fig. 4. The sequence of errors (MAE) for different starts of the training set. The value at $x = i$ is the error of the model trained on samples from i to 220. The error is measured on the interval 221-235

features $\boldsymbol{x_i}$. In the water consumption example, y would be the water consumption and $\boldsymbol{x}$ would be the variables we use to model it (previous consumption, day of week etc.). From $\boldsymbol{x_i}$ and y_i we form the tuple $(\boldsymbol{x_i}, y_i)$ that we denote $(\boldsymbol{x}, y)_i$. Then, we have a set of observed tuples:

$$D = \{(\boldsymbol{x}, y)_i, 1 \leq i \leq t\} \tag{1}$$

which is our training set. We assume that, at time t, the variables follow a model

$$y = f(\boldsymbol{x}; \boldsymbol{\theta}) + \epsilon \tag{2}$$

This means that y is a function of the features $\boldsymbol{x}$ and a vector of parameters $\boldsymbol{\theta}$, plus random noise ϵ. For example if the model is linear then $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{w} + \epsilon$. The function f and the parameters $\boldsymbol{\theta}$ define the model of y. Our goal is, from the dataset (training set) D, to infer the values of $\boldsymbol{\theta}$. Because of the noise ϵ, it is impossible to recover the true values of $\boldsymbol{\theta}$ from a finite dataset. Using a machine learning algorithm we can obtain an approximation $\boldsymbol{\theta}$*. For this approximation we can either use the entire training set D or some subset of D. We denote as $\boldsymbol{\theta_j^*}$ the approximation that the algorithm obtains from the subset $\{(\boldsymbol{x}, y)_i, j \leq i \leq t\}$ of D. We evaluate $\boldsymbol{\theta_j^*}$ by a metric of its error. One common such metric is the Mean Absolute Error which is defined as:

$$MAE = E[|y - f(\boldsymbol{x}; \boldsymbol{\theta^*})|] \tag{3}$$

where E is the expectation of the absolute difference between the predicted and the real value.

Given the training set D, we aim to learn the parameters $\boldsymbol{\theta}$ that define the best model for the time series, for the *current state* (i.e. at time t). Formally, the problem is to find:

$$\underset{i}{\operatorname{argmin}} \quad E|y - f(\boldsymbol{x}; \boldsymbol{\theta_i^*})|] \tag{4}$$

That is, we want to find the point in time from which on we should include the samples in the training set so as to get the optimal parameters, according to the used error metric.

C. Algorithm Description

Based on the formalization of Section III-B, our goal is to make the optimal use of training set D, in order to produce the most fitting model for the time series at time t.

As previously stated, we assume that the time series exhibits the property of *temporal locality*. With respect to the model, this means that the parameters $\boldsymbol{\theta}$ may change through time but they either change *slowly* or they retain their value for *some time*. Based on this, we define the validation set $D_v = \{(\boldsymbol{x}, y)_i, t - n_v < i \leq t)$ of size n_v. That is, our validation set, independent of it size n_v ends at the current time t. Thus, we expect the model that best fits the validation set to be very close to the optimal model for time t (current state of the time series). By estimating the error of the model on D_v we approximate the generalisation error and assess the importance of possible changes in the underlying model of the time series to our learned model. The size n_v of the validation set is a parameter of the algorithm.

Further, we assume a minimum training set $D_{train\_min} = \{(\boldsymbol{x}, y)_i, t - n_v - n_t < i \leq t - n_v)$, of size n_t. At the first phase, described in Algorithm 4, the algorithm sequentially adds previous data points into the training set, re-trains the model and obtains an error on the validation set. By the end of that procedure, a sequence of error vectors $\boldsymbol{E}$, with each error vector $\boldsymbol{e_i}$ corresponding to different training i of the model, is available:

$$\boldsymbol{E} = [\boldsymbol{e_1}, \boldsymbol{e_2}...\boldsymbol{e_{t-n_v-n_t}}] \tag{5}$$

Each error vector $\boldsymbol{e_i}$ has size n_v, and contains the absolute errors for each point in the validation set, for the specific training i:

$$e_{ij} = |y_{t-n_v+j} - f(\boldsymbol{x_{t-n_v+j}}|\boldsymbol{\theta_i^*})|, \quad 1 \leq j \leq n_v \tag{6}$$

For convenience we also define a sequence with the Mean Absolute Error for each i:

$$\boldsymbol{E_\mu} = [\bar{e}_1, \bar{e}_2...\bar{e}_{t-n_v-n_t}] \tag{7}$$

$$\bar{e}_i = \frac{1}{n_v}\sum_{j=1}^{n_v} e_{ij} \tag{8}$$

At the second phase, described in Algorithm 5, the sequence of errors is examined for significant deviations in the error that would suggest the location of the optimal point for the training of the model.

Within the for loop in Algorithm 5 (lines 4-9) it is decided whether a point in time denotes an important change in the model. Next, we perform an analysis that explains their functionality.

In a setting where the underlying model remains the same, the errors within the sequence of errors $\boldsymbol{E_\mu}$, produced in Algorithm 4, are, generally, expected to decrease as the index of time approaches 1. This is expected because, as the sample grows, a better estimation of $\boldsymbol{\theta}$ is possible. On the same time, several points or sub-sequences where the error increases as we add more samples to the training set are also expected

Algorithm 4: Obtain sequential errors

Data: dataset D, parameters n_v, n_t
Result: sequences of error $\boldsymbol{E}$,$\boldsymbol{E_\mu}$

1 $D_{train} = \{(\boldsymbol{x}, y)_i, t - n_t - n_v < i \leq t - n_v\}$
$D_{val} = \{(\boldsymbol{x}, y)_i, t - n_v < i \leq t\}$
2 **for** i *from* $t - n_v - n_t$ *to* 1 **do**
3 $\quad D_{train} \leftarrow D_{train} \cup (\boldsymbol{x}, y)_i$
4 $\quad \boldsymbol{\theta_i^*} \leftarrow TrainModel(D_{train})$
5 $\quad \boldsymbol{e_i} \leftarrow MeanAbsoluteError(\theta_i^*, D_{val})$
6 **end**

Algorithm 5: Find optimal point

Data: sequence of errors $\boldsymbol{E}, \boldsymbol{E_\mu}$, parameters w, α
Result: optimal point $index$

1 $min_e \leftarrow E_\mu[1]$
2 $index \leftarrow 1$
3 **for** i *from 1 to* $t - n_t - n_v - w + 1$ **do**
4 $\quad$ Let j be the index of the median of the set $\{E_\mu[l], i \leq l < i + w\}$
5 $\quad \boldsymbol{d_j} = \boldsymbol{e_1} - \boldsymbol{e_j}$
6 $\quad$ **if** $Student's\ t\ Test(\boldsymbol{d_j}, \alpha)\ and\ E_\mu[j] < min_e$ **then**
7 $\quad\quad index \leftarrow i$
8 $\quad\quad min_e = E_\mu[j]$
9 $\quad$ **end**
10 **end**

to appear, due to the statistical nature of the procedure. In order to obtain meaningful results, we need to distinguish the fluctuations in the error that occurred randomly from the significant fluctuations that would indicate a change in the underlying behaviour of the time series.

We define vector $\boldsymbol{d_i}$ as:

$$\boldsymbol{d_i} = \boldsymbol{e_1} - \boldsymbol{e_i} \tag{9}$$

where $\boldsymbol{d_i}$ measures the improvement in the error if we do not include in the training set points prior to i, against including the whole set of points in the training set. At each iteration of Algorithm 5 we have two competing hypotheses.

Hypothesis 0: The values of $\boldsymbol{d_i}$ represent an insignificant improvement and can be considered random (H_0).

Hypothesis 1: The values of $\boldsymbol{d_i}$ represent a significant improvement (H_1).

If H_1 is true at iteration i then $\boldsymbol{\theta_i^*}$ is considered better approximation to $\boldsymbol{\theta}$ than $\boldsymbol{\theta_1^*}$. Respectively, if H_0 is true then $\boldsymbol{\theta_1^*}$ is considered a better approximation of $\boldsymbol{\theta}$, because it is an estimation from a larger sample.

If we approximate the distribution of $\boldsymbol{d_i}$ with the normal distribution, we can use the Student's t-Test (Algorithm 5, line 6) to evaluate the hypotheses. First we calculate the sample mean and the sample standard deviation of $\boldsymbol{d_i}$:

$$\bar{d}_i = \frac{1}{n_v} \sum_{j=1}^{n_v} d_{ij} \tag{10}$$

$$s_{d_i} = \sqrt{\frac{1}{n_v - 1} \sum_{j=1}^{n_v} (\bar{d}_i - d_{ij})^2} \tag{11}$$

Then we calculate T statistic as:

$$T_i = \frac{\bar{d}_i}{\frac{s_{d_i}}{\sqrt{n_v}}} \tag{12}$$

where T_i corresponds to iteration i.

Under H_0, the T statistic follows a Student's t-distribution with $n_v - 1$ degrees of freedom. We define a level of significance $0 \leq \alpha \leq 1$ (usually we set some small value, e.g. 0.05). For α we can calculate, from the distribution of the T statistic, some threshold value T_α such that:

$$p(T < T_a | H_0) > 1 - \alpha \tag{13}$$

Then for a calculated T_i, we can reject H_0 if:

$$T_i > T_a \tag{14}$$

Equation 13 means that, if H_0 is true, then with high probability $T < T_a$. If we observe $T_i > T_a$ then most likely H_0 is false. That makes H_1 true which, as we stated previously, makes $\boldsymbol{\theta_i^*}$ a better approximation than $\boldsymbol{\theta_1^*}$.

Regardless of the chosen test of significance, there is always at most α probability for the H_1 hypothesis to be erroneously accepted. If, however, we evaluate multiple hypotheses, the probability that, at least one hypothesis H_1 is erroneously accepted, is generally greater than α. We denote as B the event that at least one H_1 hypothesis is erroneously accepted and as A_i the event that $T_i < T_a$:

$$\begin{aligned} p(B) = 1 - p(A_i \cap_i | H_0) = \\ 1 - p(A_1|H_0)p(A_2|H_0, A_1)p(A_3|H_0, A_1, A_2)... \end{aligned} \tag{15}$$

We see in (15) that the probability of A_i is conditioned on $A_{i-k}, 1 < k < i$. A_i and A_{i-1} depend on T_i and T_{i-1} which are calculated from $\boldsymbol{\theta_i^*}$ and $\boldsymbol{\theta_{i-1}^*}$. $\boldsymbol{\theta_i^*}$ and $\boldsymbol{\theta_{i-1}^*}$ are estimated from training sets that differ only by one sample. Thus, we can say that T_i and T_{i-1} are dependent which means that A_i, A_{i-1} are also dependent. In order to set the appropriate T_α, to limit the probability of error B to a desired level, we need to know the joint distribution of $A_i, \forall i$. Because of the dependence, this joint distribution is difficult to calculate and is different for each machine learning algorithm.

In practice there are two ways to handle this problem. The one is to calculate the joint probability through Monte Carlo simulations. This way we can get a reasonable approximation but at great computational cost. The other way is to empirically set some rule that, based on the above analysis and the properties of the problem, would effectively identify the cases of statistical significance from the random fluctuations. We implement the empirical method that is described next.

Through the iterations of Algorithm 5, when the algorithm moves past the optimal point that we are searching for, we expect that the error will increase. Due to the property of temporal locality and the dependence of subsequent errors

discussed above, we expect that the increase will be persistent for some interval. Because of the multiple hypotheses, the probability that a point with significantly low error will appear at random increases. However, the probability that the error remains low, at random, for a continuous sequence of points of length w, diminishes quickly as w increases. Thus, in order to identify important deviations in the error, that would suggest a change in the underlying model, we require that the error remains low persistently for several neighbouring points. To achieve this we scan the sequence of errors using a sliding window of size w and select the median of this sliding window (Algorithm 2, line 4) to perform the significance test (Algorithm 2, line 6). This is a simple and elegant way to ensure that the error consistently remains low, for $\frac{w}{2}$ points in the neighbourhood of the selected point.

For example, for the sequence of errors shown in Figure 4 and for $w = 10$, our algorithm selects the point at $x = 138$. The error at this point is the the median of the errors for the set of points from $x = 137$ to $x = 147$. As we can see, by applying this criterion, we select a point that is part of a subsequence of points with consistently low error. At $x = 184$ there is point that also presents a significant decrease in error which, however, is not persistent and we consider it random. The point at $x = 184$ would not be selected by our algorithm because it is not the median of any subsequence.

D. Parameters of the Algorithm

There are four parameters that need to be set in the algorithm: the size of the validation set n_v, the size of the minimum training set n_t, the confidence level α and the size of the window w. The tuning of these parameters depends on the qualities of the specific problem. Below we will qualitatively describe their effect on the performance of the algorithm.

Parameter n_v controls how confident we can be about the value of the error $\bar{e}_i$. A larger n_v limits the effect of the random variance of the validation error and provides a better approximation to the true generalisation error. This allows smaller shifts of the error to be identified as significant at the given level of confidence α. This can be derived from the equations of the t-Test but is also fairly intuitive. On the other hand with a larger n_v the algorithm requires more time to acquire the necessary samples, so the it would have slower response to changes in the time series.

Parameter α controls the level of confidence required in order to identify a change as significant. The probability that a single hypothesis of Algorithm 5 is erroneously accepted is, in the worst case, equal to α. Therefore a lower α limits the probability of error. The downside of using a very small α is that it decreases the sensitivity of the algorithm to shifts in the error. With a lower alpha only larger shifts in the error are considered as significant.

Parameter n_t controls the size of the initial training set for the algorithm. Each algorithm requires a minimum set of points in order to provide confident predictions. So, depending on the algorithm, we should set n_t large enough, so that the variance of the error in the initial iterations of Algorithm 4 remains at an acceptable level.

Parameter w is used in Algorithm 5 in order to compensate for the multiple hypotheses problem. Larger w makes the algorithm less sensitive to shifts in the level of the error. The value of w is dependent on the length of time series Y so longer Y would require larger w. To our experience the results are not very sensitive to changes in w so a proper value can be identified after a few tuning trials.

E. Alternative Significance Test

By experimenting on the water consumption dataset we have found that the normality assumption for the error is often violated by the existence of outliers. This hinders the performance of the t-Test used for significance testing in our algorithm. Also, except for the statistical significance, it may be beneficial to include the magnitude of the error change in the change point selection process. To this end we propose an empirical method to assess the significance of a change in the error. We consider change at time i significant if:

$$\frac{E_\mu[1] - E_\mu[i]}{E_\mu[i]} > \alpha_{thresh} \tag{16}$$

where α_{thresh} is a parameter that controls the sensitivity of the algorithm to changes in the error, similar to α. This test replaces the Student's t-Test in Algorithm 5 line 6. All other parts of the analysis and the algorithm remain the same.

IV. Experiments and Evaluation

In this section, we evaluate the effectiveness of the proposed method, on optimizing the training of machine learning models on time series. As we have previously argued, the proposed method can be applied on top of different machine learning algorithms. To demonstrate this, we apply it on three algorithms used for time series forecasting: Support Vector Regression, Elastic-net Regularized Linear Regression and Neural Network. In our setting, each of these algorithms is trained on a part of dataset D and aims at predicting the next time series values. In our experiments, we compare the training performed using our method against three baselines: the naive method of training on the whole available training set and two change point detection based methods that identify the point in time (in the training set) past which the time series changes, and denote this point as the beginning of the training set.

A. Algorithms

Next, we provide a brief description of the three machine learning algorithms that we use in our experiments to assess our method, against the three baselines for training machine learning models.

1) Linear Model with Elastic Net Regularization : Elastic Net Regularization (ENET) is a state of the art method for linear regression models [12], that combines L1 and L2 regularization along with the squared loss function. A linear model has the form:

$$y = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{\beta} \tag{17}$$

The $\boldsymbol{\beta}$ recovered by the Elastic Net method are:

$$\boldsymbol{\beta}^{*} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}(\|y - X\boldsymbol{\beta}\|^2 + \lambda_2||\boldsymbol{\beta}||_2 + \lambda_1||\boldsymbol{\beta}\ ||_1)$$

λ_1 and λ_2 are parameters that control the L1 and L2 regularization respectively. We use cross validation to tune the parameters on each training case. We note that with $\lambda_1 = 0$ we get the Ridge Regression algorithm and with $\lambda_2 = 0$ the LASSO algorithm. This method has very low variance and is particularly effective for cases with small number of samples.

2) Support Vector Regression: Support Vector Regression (SVR) is the regression version of the popular Support Vector Machines algorithm. SVR can be used with various kernels that transform the data before fitting the model. We use the linear kernel so the the model is linear as in Equation (17). SVR is a low variance algorithm as well. Unlike the ENET it uses only L2 regularization and L1 loss function. It also applies the so called ϵ -insensitive tube. That means that errors smaller than ϵ are ignored in the estimation of $\boldsymbol{\beta}$. The parameters of the model are ϵ and C. We use a default value of 0.1 for ϵ, after the data are normalized to zero mean and unit variance. For the tuning of parameter C, which controls the level of regularization, we use the heuristic function provided by liblinear [19], for each training case of the algorithm.

3) Feed-Forward Artificial Neural Network: The Feed-Forward Artificial Neural Network (ANN) is a complex, high variance model. As such, it can model arbitrary functions but at the cost of requiring a large dataset. The building block of ANN is the node/neuron, where a linear combination of some inputs is computed, passed through a non-linear activation function and given as output. The nodes are organized in layers, with the first layer taking the features $\boldsymbol{x}$ as input and each next layer taking as input the output of the previous. The last layer provides the output of the network and, in the case of regression, has no activation function. All intermediate layers between the input and the output are called hidden layers. The weights of the linear combinations are usually computed by gradient descent using the back-propagation algorithm. There are many parameters that need to be set for the operation of ANN the most important of which are the number of hidden layers, the number of nodes per layer, and the activation function. In our implementation we used one hidden layer with 5 nodes and the sigmoid activation function.

B. Baselines

We compare our algorithm with three baselines. The first and most simple is to use the entire dataset as the training set. The other two are based on a change point detection algorithm that detects changes in the distribution of a sequence of values. There are several different such algorithms that can be used, depending on the assumptions about the data and the aspects of the distribution that we want to examine. The change point detection algorithm we select utilizes the Mann-Whitney test [1] which can detect shifts in the level (mean value) of the time series, without requiring an assumption about the distribution. It is a state of the art algorithm that is used for change point detection.

The Mann-Whitney test is a non-parametric test that is applied on two samples and evaluates whether they come from a population with the same mean or from populations with different means. As a non parametric test, it makes no assumption about the distribution of the values, so it can be used with arbitrary distributions. Intuitively, it sorts the joint population of both samples and then it calculates the rank of each value, on the joint population. Then it examines whether there is a significant difference in the ranks between the two samples. If the ranks of the points in one sample are consistently higher than those in the other, that indicates that there is a difference in the means of the distributions.

In order to identify if a point of the time series is a change point the algorithm compares the part of the time series before that point and the part after, using the Mann-Whitney test. If there is a difference between the two parts, the point is identified as a change point. In order to examine the entire time series, the algorithm performs the same procedure for every point. For a more detailed description we refer the reader to [1].

Baseline 1: Baseline 1 (BL1) is the naive approach where we do not consider any change in the model and use the entire dataset D as training set.

Baseline 2: In Baseline 2 (BL2) we identify the changes in the distribution of Y using the Mann-Whitney test as is and set as start of our training set the most recent change in the distribution. We also exclude change points that leave less than a minimum set of samples for training. Specifically, let $\{c_j, 1 \leq j \leq k, c_1 = 1\}$ be the set of change points obtained by the test. We find $c_p = max\{c_j : t - c_j > n_{min}\}$. We set as minimum training set size $n_{min} = 20$. We use as training set the points after c_p.

We note that Baseline 2 detects changes in the distribution. However, there may be changes in the underlying model that do not cause changes in the distribution. Those are not detected by Baseline 2. In order to account for those we use propose Baseline 3.

Baseline 3: In Baseline 3 (BL3) we first obtain the training residual errors of the model. The training residual errors are the difference between the actual value and the prediction of the learned model, at each point of the training set. Then we apply the change point detection algorithm on the time series of the residual errors. This way we can detect arbitrary changes in the model. This procedure is very similar to the one described in [6].

Similarly to Baseline 2, we consider as start of the training set the most recent change point identified, but exclude change points that leave less than a minimum set of samples in the training set. Specifically, we train the model on the whole training set D and obtain θ_0^*. Then the training residuals are defined as:

$$Y_{res} = \{y_i - f(x_i; \theta_1^*), 1 \leq i \leq t\}$$

Applying on Y_{res} the exact same procedure described for Baseline 2 we obtain its change points $c_p = max\{c_j : t - c_j > n_{min}\}$ and select $c_p = max\{c_j : t - c_j > n_{min}\}$ as the start of the training set.

C. Datasets and Evaluation Measures

We evaluate the proposed algorithm on two datasets, one from a real-world application and a synthetic one. On the synthetic dataset we examine the generalisation performance of the algorithms, with respect to the bias-variance trade-off, on time series where the underlying model changes. We showcase an important advantage that our algorithm has over state of the art change detection methods on effectively handling this trade-off.

1) Real Dataset: The real dataset consists of 600 water consumption time series from individual households. The quantity that we wish to model (y) is the daily water consumption of the household and the features ($\boldsymbol{x}$) are the day of week and the hourly water consumptions of the previous day. Each time series corresponds to one year of water consumption with the measurements starting on 1 July 2013 and ending on 30 June 2014. We have found water consumption of individual households to be difficult to model. Its behaviour displays both gradual and steep changes. Gradual changes may be due to seasonal effects, which can be dealt with by traditional algorithms, but may also be random due to other factors, such as a slow change in the water consumption habits of the inhabitants. The cases that we are mostly interested in are those of random, unpredictable, changes. These are due to changes in household activity (e.g. a guest arriving, a vacation/off-work period, an illness, a change in the daily routine, etc.). Such changes constitute a challenge to the modelling of the time series, because data prior to the change may or may not be beneficial to include in the training of our model. On top of that, water consumption exhibits significantly random behaviour and is difficult to measure exactly, making the problem of model change more difficult.

Water consumption data is actually the use case that provided us with the motivation to study the effects of changing behaviour on the modelling of time series.

2) Synthetic Dataset: We use a synthetic dataset to highlight the importance of evaluating the generalisation error of the model that results from the change point detection process and the shortcomings of existing change point detection methods in that aspect.

In machine learning algorithms there are two sources of reducible error: *bias* and *variance*. Bias is the error that appears when the model is not able to accurately represent the data (e.g when a linear model is used to capture a quadratic relationship). Variance is the error that results from estimating a parameter from a sample of *finite size*, in the presence of random noise. The effect of variance becomes more significant as the sample size becomes smaller. In order to evaluate both aspects of the bias-variance trade-off we use the synthetic dataset described below.

Let time series $Y = \{y_i, 1 \leq i \leq 350\}$ be generated by the following process: We generate 350 vectors $\boldsymbol{x_i}, 1 \leq i \leq 350$ of size 100, with each element j of each vector uniformly distributed in $[0, 10]$: $x_{ij} \sim unif(0, 10), 1 \leq j \leq 100$. We generate one vector $\boldsymbol{w}$ of size 100 with $w_j \sim unif(0, 5), 1 \leq j \leq 100$. From $\boldsymbol{w}$ we generate $\boldsymbol{w_1}$ where $w_{1j} = w_j + N(0, 2), 1 \leq j \leq 100$, and $\boldsymbol{w_2}$ where $w_{2j} = w_j + N(0, 0.5)$ ($N(\mu, \sigma)$ is Gaussian distributed noise with mean μ and variance σ). $\boldsymbol{w_1}$ and $\boldsymbol{w_2}$ represent two changes of the model, one slighter and one more intense, so that one introduces a significant bias to the model and the other a smaller bias. We define y_i as:

$$y_i = \begin{cases} \boldsymbol{x_i}^T \boldsymbol{w_1} + N(0,1), i \in [1, 80] \\ \boldsymbol{x_i}^T \boldsymbol{w_2} + N(0,1), i \in (80, 200] \\ \boldsymbol{x_i}^T \boldsymbol{w} + N(0,1), i \in (200, 350] \end{cases} \tag{18}$$

The underlying model is designed in this way so that it has two changes. One that introduces a significant bias and one that introduces a smaller bias. By testing the algorithms on different time-points of time series generated from this model, we can evaluate their behaviour with respect to the bias and variance aspects of the generalisation error.

The synthetic dataset consists of 100 time series generated by the above process.

Evaluation Metric: In order to evaluate the algorithms we first calculate the prediction error of each one on a specified forecast horizon, using the MAE metric:

$$\hat{MAE} = \frac{1}{n_{test}} \sum_{i=t+1}^{t+n_{test}} |y_i^* - y_i| \tag{19}$$

Where y_i^* is the prediction of the model for time i, y_i is the actual value and n_{test} is the size of the forecast horizon. We then use the normalized improvement of each algorithm over Baseline 1 (the naive method of using the whole available dataset as training set) as the evaluation metric. For algorithm A the improvement I is:

$$I_A = \frac{\hat{MAE}_{BL1} - \hat{MAE}_A}{\hat{MAE}_{BL1}} \tag{20}$$

Day of Year	80	120	160	200	240	280	320	340	Avg
ITM-stat	1.6	2.2	5.4	5.2	3.4	2.5	2.4	2.7	3.2
ITM-emp	0.5	2.5	5.3	5.9	5.0	0.9	4.4	3.4	3.5
BL2	0.0	2.3	2.6	2.2	2.2	0.4	2.3	-0.1	1.5
BL3	0.2	2.5	1.0	2.9	2.3	0.0	1.8	0.3	1.4

TABLE I
ENET RESULTS ON WATER CONSUMPTION DATASET

Day of Year	80	120	160	200	240	280	320	340	Avg
ITM-stat	0.3	1.5	3.5	3.1	1.5	0.4	0.4	1.0	1.5
ITM-emp	0.0	1.8	4.8	3.4	2.2	3.2	2.3	3.1	2.6
BL2	-1.5	1.6	2.2	0.9	-0.4	-0.9	-0.8	-0.6	0.0
BL3	0.0	2.3	1.6	0.7	-0.9	-2.0	-1.4	-2.1	-0.2

TABLE II
SVR RESULTS ON WATER CONSUMPTION DATASET

D. Experiments

In this section we present the results of the experimental evaluation of the algorithms.

In the results we denote our algorithm as ITM (Iterative Training Method) with the suffix "stat" for the statistical version, that uses the t-Test, and"emp" for the empirical method described in Subsection III-E.

Day of Year	80	120	160	200	240	280	320	340	Avg
ITM-stat	1.4	2.7	2.8	3.8	2.4	1.7	1.7	0.7	2.2
ITM-emp	0.5	2.2	2.4	4.1	3.4	1.3	2.1	-0.5	2.0
BL2	0.4	3.1	1.6	2.2	1.3	1.7	1.8	0.1	1.5
BL3	0.2	2.5	1.3	1.5	1.3	0.4	1.6	-0.5	1.0

TABLE III
ANN RESULTS ON WATER CONSUMPTION DATASET

	BL1	BL2	BL3
ITM-stat	6×10^{-17}	2×10^{-7}	4×10^{-7}
ITM-emp	9×10^{-21}	7×10^{-11}	5×10^{-13}

TABLE IV
ENET P-VALUES

	BL1	BL2	BL3
ITM-stat	1×10^{-8}	8×10^{-6}	1×10^{-6}
ITM-emp	1×10^{-11}	1×10^{-4}	2×10^{-6}

TABLE V
SVR P-VALUES

	BL1	BL2	BL3
ITM-stat	6×10^{-10}	0.135	7×10^{-3}
ITM-emp	2×10^{-7}	4×10^{-3}	0.043

TABLE VI
NN P-VALUES

1) Water Consumption Dataset: In Tables I, II, III the experimental results in terms of $\%$ error improvement compared to BL1 are provided. In order to better simulate the setting of an evolving time series, we apply the algorithms at several different parts of the dataset, i.e. points in time. Each column demonstrates the average improvement, over all 600 time series, when the algorithms were applied at the respective day of year, specified in the top row. The parametrization for each instance of our algorithm is shown in Table VII. We experimentally tuned the parameters for each case. The range we searched for the parameters n_v,n_t,n_{test},w,α, was [13, 30],[7, 25],[5, 10], [10, 20], [0.01, 0.3] respectively. In those ranges we tried a total of 15 configurations (combined, for all parameters).

As we see, our algorithm provides a considerable improvement over the naive baseline (BL1), by 3.5%, 2.5 % and 2.2 % on average for ENET,SVR and ANN respectively. It also outperforms the two change point detection approaches (BL2,BL3), by approximately 2% for ENET and SVR and 0.7% for ANN. The improvement is consistent for every algorithm and every period of the year. In order to assess the statistical significance of the results we perform t-tests on the difference of the errors, between our algorithms and the baselines. In Tables IV,V,VI we provide the p-values of the tests, that indicate the probability that the difference is the result of random variance. We see that in almost all cases the statistical significance of the improvement is substantial. Thus we can say that there are, in fact, aspects in the training of a machine learning model that our algorithm captures more effectively than the baselines.

The difference between the empirical and the statistical versions of our algorithm is considerable only in the case of the SVR algorithm, where the improvement of the is above 1%. There is also a slight improvement in the case of the Elastic Net algorithm. This indicates that the assumption of normality for the error, used in the statistical case, may be violated by the presence of outliers. Between the three implemented algorithms the greatest improvement is observed in the case of the Elastic-Net algorithm. This may be due to the fact that the sophisticated regularization scheme of this algorithm makes it able to be trained from very small datasets, thus responding more quickly to changes in the model. SVR is also an algorithm able to be trained from small datasets, however the heuristic function used for the choice of regularization parameter C may not be optimal and better performance could be achieved by cross validation.

We note that the performance of the algorithms vary throughout the year. For example, in the case of the Elastic-Net algorithm, the improvement ranges from 0.5% to 5.9%, which shows that the improvement depends on the characteristics of the time series. The highest improvement is in days 160,200,240, that corresponds to months December,January and March. This can be attributed to the existence of the winter holidays, that affect water consumption and subsequently the training of the model. We can not evaluate the effects of seasonality because we only have one year of measurements for each time series.

2) Synthetic Dataset: We apply all algorithms on 100 time series generated from the model described in Subsection IV-C2. Tables VIII, IX, X demonstrate the results in error improvement. The parameters n_v, n_t, w and n_{test} for the experiments were set to $25, 10, 10$ and 10 respectively. They were set based on the known characteristics of the synthetic dataset. To determine α we performed 3 experiments for each algorithm in the range [0.01, 0.3]. The selected values were 0.2,0.1,0.1,0.01,0.1 and 0.01 for $ENET_{stat}$, $ENET_{emp}$, SVR_{stat}, SVR_{emp}, ANN_{stat} and ANN_{emp} respectively. The synthetic dataset as well as implementations of our algorithms and the baselines are available for experimentation at https://goo.gl/LURI3L .

We see that our algorithm provides an improvement over the naive baseline (BL1), as well as the change point detection based baselines (BL2, BL3). The improvement over BL1 is greater for ENET (20.4%) and SVR(17.4%) but is also considerable for ANN (5.3%). The lower improvement for ANN is most likely due to its requirement for larger sample size. The empirical method outperforms the statistical on SVR and ANN. This suggests that evaluating the magnitude of the change in the error might be beneficial, even in cases where the normality assumption for the error holds. The improvement over BL2 and BL3 is also very significant. Indicatively, for the ENET algorithm the improvement is by 17.5% and 19.7%, for BL2 and BL3 respectively. The statistical significance of all improvement is very high. The p-values for the comparisons of all algorithms fall in the range of $[2.6 \times 10^{-31}, 8.5 \times 10^{-5}]$.

Each time series of the synthetic dataset contains 2 changes in the underlying model. One that happens at time 80 and one that happens at time 200, with the one that happens at time 80 being more intense (Equation 18). In the results (e.g. Table VIII), we see that after time 80 all the algorithms gain an advantage over BL1 (columns 2,3). That is the expected

	n_t	n_v	n_{test}	w	α
E-Net Stat.	10	25	10	20	0.05
E-Net Emp.	20	20	10	20	0.3
SVR Stat.	25	15	10	20	0.2
SVR Emp.	25	15	5	20	0.05
ANN Stat.	13	7	5	10	0.05
ANN Emp.	25	15	5	20	0.05

TABLE VII
PARAMETRIZATION OF ITM

Time	80	120	160	200	240	280	320	Avg
ITM-stat	-0.8	11.7	11.1	14.2	30.4	31.7	44.2	20.4
ITM-emp	-0.8	11.7	11.1	14.2	30.4	31.7	44.2	20.4
BL2	-1.3	6.6	8.7	-2.9	-3.1	-9.5	22.5	2.9
BL3	-0.3	3.0	7.8	2.7	-18.1	-23.8	34.0	0.7

TABLE VIII
ENET RESULTS ON SYNTHETIC DATASET

Time	80	120	160	200	240	280	320	Avg
ITM-stat	0.0	20.2	20.2	4.5	19.2	19.9	23.9	15.4
ITM-emp	-0.2	18.9	26.1	5.7	22.2	24.1	25.6	17.4
BL2	-2.1	17.3	23.4	-3.5	0.5	-7.7	-1.1	3.8
BL3	-0.22	14.8	26.2	0.6	-14.0	-10.5	-7.4	1.3

TABLE IX
SVR RESULTS ON SYNTHETIC DATASET

Time	80	120	160	200	240	280	320	Avg
ITM-stat	-1.8	16.0	11.0	-4.8	3.4	2.5	-0.2	3.7
ITM-emp	0.2	20.5	13.5	-10.9	-0.2	2.0	12.4	5.3
BL2	-1.3	19.1	13.1	-17.7	-19.9	-35.6	-55.3	-13.9
BL3	-0.1	19.5	11.6	-24.4	-27.2	-32.1	-62.9	-16.5

TABLE X
ANN RESULTS ON SYNTHETIC DATASET

behaviour, since the model changes and including the entire dataset skews the model. After time 200 the second change in the model happens, which is of lower intensity. As we see at columns 5-6, after time 200, the performance for our algorithm increases while baselines BL2, BL3 drop below the naive baseline BL1. At time 320 all algorithms have a great improvement over BL1, with the improvement of ITM still being significantly higher. The behaviour of the baselines at times 240 and 280 highlights the advantage that our algorithm gains by evaluating the generalisation error in the change point selection process. In order to better understand this we need to see how each algorithm would behave in a time series generated from the synthetic model (Equation 18), if applied for example at time 280.

The baseline algorithms BL2 and BL3, that detect changes in the distribution and the model respectively, generally select the point 200 as the start of the training set. That may seem like the optimal choice, since it represents the most recent change in the model. This choice includes in the training set only samples that follow the same model as the one we want to predict. This choice, however, is optimal only in terms of *bias*. It does not take into account the other aspect of the generalisation error, which is *variance*. It leaves only 80 samples in the training set, which may not be enough for the algorithm to converge to a good approximation of the underlying model. This creates a bias-variance trade-off that defines the optimal choice for the change point.

Shown in Figure 5 is the sequence of errors for the prediction of y_i as we include previous points in the dataset for an example from the synthetic dataset. We see that stopping the training at $x = 200$ does not provide the minimum error. Including several points from the interval $[80, 200]$, although they are generated from a slightly different model, leads to a reduction in the error. Including points before $x = 80$, however, leads to a very significant increase in the error.

This is a result of the bias-variance trade-off discussed above. By including points from $[80, 200]$ we introduce a small bias in our model, because the points of that interval follow a slightly different model, but we benefit from a significant reduction in variance, since we have a lot more samples to train from. On the other hand if we include samples prior to time 80 the introduced bias is larger and, in the time series of the example, the reduction in variance does not overcompensate for it. The identification of the optimal point with respect to this trade off is not straight forward since it depends on the complexity of the model, the characteristics of the change and the number of samples available since the change. While existing approaches focus on bias, our algorithm evaluates both aspects of the generalisation error through the use of a validation set and achieves better performance.

V. Conclusion

We propose an algorithm that optimizes the training process of machine learning models on time series where the underlying model may change through time. Our algorithm evaluates the generalisation performance of the trained model. It accounts for an important aspect of the behaviour of machine learning models, related to the bias-variance trade-off, that is not addressed by other approaches. We evaluate our algorithm on both a synthetic and a real-world dataset and show that it provides a significant improvement.

VI. References

References

[1] D. M. Hawkins, Q. Deng. *A Nonparametric change-point Control Chart*. Journal of Quality Technology, 2010.

[2] R. P. Adams, D.J.C. MacKay. *Bayesian Online Changepoint Detection*. International Journal of Forecasting, 2007.

[3] F. Gustafsson. *Adaptive Filtering and Change Detection*. 1st ed., Wiley, 2000.

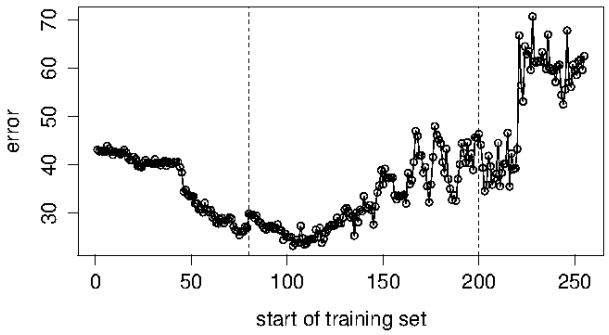

Fig. 5. The sequence of errors (MAE) for a time series generated from the model described in Equation 18, as we move backwards in the training set. The value at $x = i$ is the error of the model trained on samples from i to 280. The error is measured on the interval 281-290. The dashed lines are the points where the model changes.

[4] V. Mithal, A. Khandelwal, S. Boriah, V. Kumar. *Change Detection from Temporal Sequences of Class Labels: Application to Land Cover Change Mapping*. SIAM International Conference on Data Mining, 2013.

[5] Z. Wang, P. Chakraborty, S. R. Mekaru, J. S. Brownstein, J. Ye, N. Ramakrishnan. *Dynamic Poisson Autoregression for Influenza-Like-Illness Case Count Prediction*. Conference on Knowledge Discovery and Data Mining, 2015.

[6] V. Guralnik, J. Srivastava. *Event Detection from Time Series Data*. Conference on Knowledge Discovery and Data Mining, 1999.

[7] G. J. Ross, D. K. Tasoulis, N. M. Adams. *Nonparametric Monitoring of Data Streams for Changes in Location and Scale*. Technometrics Journal, 2011.

[8] E. W. Dereszynski, T. G. Dietterich. *Probabilistic Models for Anomaly Detection in Remote Sensor Data Streams*. Conference on Uncertainty in Artificial Intelligence, 2007.

[9] H. Ohlsson, L. Ljung, S. Boyd. *Segmentation of ARX-models using sum-of-norms regularization*. Automatica Journal, 2010.

[10] G. J. Ross. *Sequential Change Detection in the Presence of Unknown Parameters*. Statistics and Computing Journal, 2014.

[11] Y. Wang, I. Ziedins, M. Holmes, N. Challands. *Tree Models for Difference and Change Detection in a Complex Environment*. The Annals of Applied Statistics, 2012.

[12] H. Zou, T. Hastie. *Regularization and Variable Selection via the Elastic Net*. Journal of the Royal Statistical Society, 2005.

[13] J. W. Taylor, L. M. Menezes, P. E. McSharry. *A comparison of Univariate Methods for Forecasting Electricity Demand Up to a Day Ahead*. International Journal of Forecasting, 2006.

[14] G. P. Zhang, M. Qi. *Neural network forecasting for seasonal and trend time series*. European Journal of Operational Research, 2003.

[15] P. Pai, K. Lin, C. Lin, P. Chang. *Time series forecasting by a seasonal support vector regression model*. Expert Systems with Applications Journal, 2010.

[16] S. Humeau, T. K. Wijaya, M. Vasirani, K. Aberer. *Electricity Load Forecasting for Reisdential Customers: Exploiting Aggregation an Correlation between Households*. International Federation for Information Processing Conference, 2013.

[17] E. S. Gardner, J. Commander. *Exponential smoothing: The state of the art*. Journal of Forecasting, 1985.

[18] M. Mller. *Dynamic Time Warping*. Information Retrieval for Music and Motion, 1st ed, Springer, 2007.

[19] R. E. Fan, K.W. Chang, C. J. Hsieh, X. R. Wang, C. J. Lin. *LIBLINEAR: A Library for Large Linear Classification* . Journal of Machine Learning Research, 2008.

[20] G. J. Ross. *Parametric and Nonparametric Sequential Change Detection in R: The cpm Package*. Journal of Statistical Software, 2015.