Frank Rosenthal, Peter Benjamin Volk, Martin Hahmann, Dirk Habich, Wolfgang Lehner

Drift-Aware Ensemble Regression

Erstveröffentlichung in / First published in:

Machine Learning and Data Mining in Pattern Recognition. 6th International Conference. Leipzig, 23.-25.07.2009. Springer, S. 221-235. ISBN 978-3-642-03070-3.

DOI: https://doi.org/10.1007/978-3-642-03070-3_17

Final edited form was published in "Machine Learning and Data Mining in Pattern Recognition. 6th International Conference. Leipzig 2009", S. 221-235, ISBN 978-3-642-03070-3
https://doi.org/10.1007/978-3-642-03070-3_17

Drift-Aware Ensemble Regression

Frank Rosenthal, Peter Benjamin Volk,
Martin Hahmann, Dirk Habich, and Wolfgang Lehner

Technische Universität Dresden
Database Technology Group
01062 Dresden, Germany
dbinfo@mail.inf.tu-dresden.de

Abstract. Regression models are often required for controlling production processes by predicting parameter values. However, the implicit assumption of standard regression techniques that the data set used for parameter estimation comes from a stationary joint distribution may not hold in this context because manufacturing processes are subject to physical changes like wear and aging, denoted as *process drift*. This can cause the estimated model to deviate significantly from the current state of the modeled system. In this paper, we discuss the problem of estimating regression models from drifting processes and we present *ensemble regression*, an approach that maintains a set of regression models—estimated from different ranges of the data set—according to their predictive performance. We extensively evaluate our approach on synthetic and real-world data.

Keywords: Ensemble Method, Regression, Process Drift.

1 Introduction

Regression models are important tools in scientific, industrial, and many other application areas. They are used to learn a functional dependency between a numeric target variable and other variables of a given data set. Knowledge of such dependencies is often required to control industrial production processes.

A concrete example in *semiconductor manufacturing* is etching, a process where material is physically removed with means like acid or plasma to create the layout of integrated circuits. In this setting, the time that is required to etch a specific amount of material is an important process parameter. The time heavily influences the width of the structures that are etched, which defines the electrical properties of the final integrated circuit. Hence, the correct etch time has to be determined to attain a product that fulfills defined specifications on this most important quality measure. Naturally, etch times that deviate from the ideal time will result in lower product quality. Semiconductor manufacturing is technologically growing fast. Companies enhance their manufacturing processes continuously while creating products of smaller and smaller structure widths. Maintaining production processes that operate in such small dimensions—e.g., 22nm—requires extensive use of regression models for control, since theoretical research alone cannot provide the required models [1,2,3].

Creating regression models for process control requires a sufficiently large data set. In *single-lot* semiconductor manufacturing, where only few expensive products of a certain design are manufactured in complex and long-running processes, this creates a new challenge. Since the acquisition of the data set expands over a long period of real time, elements of the production process change in the meantime and the examples basically represent different processes. In our example of etching, chemicals used in the process are subject to *aging*. Hence, an acid might react more slowly, thereby removing less material per unit of time. A control model that predicts etch time therefore ideally has to consider the age of the chemical. However, the knowledge to describe such aging processes is typically not readily available.

In this paper, we consider *drifting* processes, i.e., processes that observably change over time. Aging is an example of a *continuous drift*, since it continuously changes the process. Estimating a regression model from examples of a drifting process violates the basic assumption that the data set has been drawn from one joint distribution of the involved variables. If regression is done anyway, the resulting model will not reflect the current state of the process but an average state over time. Note that such drifts occur in many industrial processes, since every machine wears and ages. In the context of single-lot semiconductor manufacturing, drift may result in direct monetary loss, since even small errors in the estimates of process parameters are critical. Besides continuous drift, there are also *abrupt drifts*. In our example, they may result from a change of the etching chemical. The effect on estimated regression models is similar to the effect of continuous drift, i.e., the estimated model does not reflect the current state of the modeled system.

One basic approach to handle drift is to restrict the data set to the most recent examples, which represent the current state of the system best. However, this restricted data set may not contain enough information to estimate complex functional dependencies, like those that occur in semiconductor manufacturing. Therefore, a tradeoff between a current model and a stable model, in terms of average prediction error, must be found. In this paper, we describe drifting processes and how drift influences regression models that have been estimated from examples generated by such processes (Section 2). In Section 3, we then propose *ensemble regression*, an approach that learns a composite regression model that reduces the prediction error when used as a model of the current state of the system. Ensemble regression uses a set of regression models estimated from different ranges of the complete data set and maintains this set according to the predictive performance of the models. *Predictions* are drawn from the ensemble using a weighted average of the predictions of all ensemble *members*. Our approach can be used in connection with any regression method. We extensively evaluate the effects of certain types of drift as well as the predictive performance of models estimated by our approach on synthetic and real-world data in Section 4. Section 5 describes related work and Section 6 concludes the paper.

Final edited form was published in "Machine Learning and Data Mining in Pattern Recognition. 6th International Conference. Leipzig 2009", S. 221-235, ISBN 978-3-642-03070-3
https://doi.org/10.1007/978-3-642-03070-3_17

2 Drifting Processes and Regression

We consider the problem of *regression* on data from *drifting processes*. Regression is the task of estimating or learning a function $\hat{y} = f(x)$ from a *data set* D consisting of n examples $d_i = (y_i, x_{j,i}), j = 1, \ldots, k, i = 1, \ldots, n$ where y is the numeric label (the dependent variable) and x_j are the k features (the independent variables). In our etching example, the time of etching is the dependent variable, while the structural width—and thereby the amount of material—is the independent variable. Each time the etching process is executed, we can acquire an example of an etch time and the structural width that has been reached.

The basic assumption in regression is that the data set has been drawn from exactly one joint distribution of dependent and independent variables [4] that captures the dependencies between them. If the joint distribution is stationary when the examples are acquired, the examples represent exactly one *functional dependence*, i.e., one *function*, disregarding the uncertainty introduced by *noise* that may be introduced by measurement equipment.

In this paper, we consider the case when the data set is acquired from a drifting process. Such a data set is not distributed according to just one distribution. Instead, the distribution function is a function of time and *drift* is the change that occurs over time. Therefore, the examples acquired from the process represent a set of functional dependencies between dependent and independent variables. Each of these functions governed the process at a time when an example was drawn. Therefore, we denote a system that yields data according to a changing underlying functional dependency as a *drifting process*. The effect of drift is that the same input values may result in different output values when determined at different points in time. In terms of the governing function, drift may be viewed as a change in functional form or as a change in the parameters. We now illustrate this view with a simple example from physics.

The electrical resistance of an object is a measure of the amount of opposition against an electric current, which is described by Ohm's law: $R = UI^{-1}$. If we want to determine the electrical resistance of an object experimentally, one way is to put it into an electrical circuit, set a certain voltage U and measure the current I. Performing this repeatedly for different values of U enables the estimation of R using linear regression, where $U = y$ and $I = x$. However, electrical resistance also depends on the temperature of the resistor; higher temperatures will result in higher resistance. Therefore, if the temperature during the experiment is not constant, the examples (U, I) will represent different underlying functions $U = R_i I$, where R_i is the resistance at the time a particular measurement was taken.

In physics, this problem has been solved by analyzing the impact of temperature on the electrical resistance and extending the basic dependence in a form that treats the resistance as a function of temperature $R(\theta)$. However, deriving such laws requires significant insight into the observed system and is not feasible for complex real-world processes like single-lot semiconductor manufacturing. Even if the influence factors and their functional form were all known, some factors might not be observable at all or not observable with acceptable cost. Additionally, real-world drifts may behave non-deterministically with abrupt

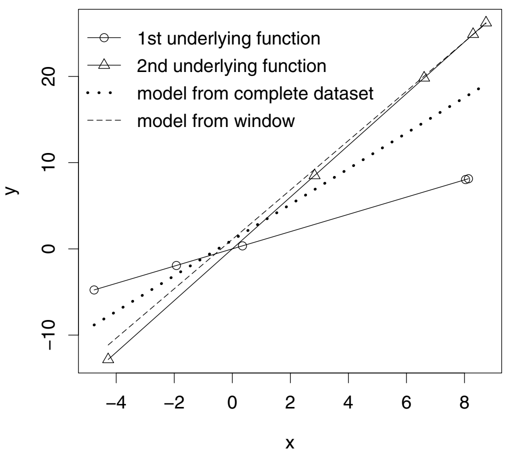


Fig. 1. Effect of an abrupt drift from the first to the second true dependency on a regression model estimated from mixed samples

changes and may be subject to random fluctuations. In our example of etching, these obstacles circumvent the modeling of the behavior of the drift. Therefore, an approach is required to estimate regression models from the data of drifting processes while minimizing the error introduced by the drift. We now present the characteristics of this error using another simple example that is depicted in Figure 1.

Consider a linear functional dependence $y = f(x) = ax + b$ from univariate input x to univariate output y. Suppose further that the functional form stays constant, but drift occurs in parameter a. Hence, $a(t)$ is a function of time and $y = f_t(x) = a(t)x + b$. Figure 1 depicts two such functions at different times t. An abrupt drift has occurred while gathering the examples. For the first five examples (the circles) $a(t) = 1$ and for the last five examples (the triangles) $a(t) = 3$. For presentation purposes, we excluded the presence of any measurement noise.

In general, n examples are acquired that can be indexed by $1, \dots, n$ in the order of the time they have been drawn from the process. When using the complete data set, i.e., $d_1, \dots, d_n$, to estimate the parameters $\hat{a}$ and $\hat{b}$ of the linear model, an average over the underlying functions is the result, since the discrepancies between the examples from the different underlying functions are implicitly discarded as noise by the regression method. Applying this approach to our example results in the dotted line in Figure 1. The estimated model and the current underlying function (line with triangles) are different and there will be an error when using the estimated model for prediction.

The dashed line in Figure 1 represents a model that has been estimated from a window of size 6, i.e., the more recent half of the data set and the last example from the first underlying function. This model resembles the recent state much closer. The sum of the squared prediction errors of this model is about a tenth of the error of the model that had been estimated from the whole data set. This example gives an indication that using older data may result in a greater prediction error. We can minimize the prediction error by using only the last example. However, this is not feasible when the functional dependence is of a more complex nature (e.g., polynomial) or when non-parametric regression is used.

3 Ensemble Regression for Drifting Processes

We are given a data set $D = \{d_i\}, i = 1, \ldots, n$, where d_i are examples from a drifting process and $d_i = (y_i, x_{j,i}), j = 1, \ldots, k$, y is the numeric label and x_j are the k features to be used in the calculation of the prediction $\hat{y}$. The core idea to reduce the effect of drift on the estimated model is to restrict the data set used for parameter estimation to a window of examples, thereby excluding older examples that do not represent the current state of the modeled system. The basic approach consists of estimating a regression model from a *window of examples* $d_{(n-w)}, \ldots, d_n$, where n is the index of the most recent example and w is the number of examples in the window. Whenever a new example is acquired, n is incremented to represent the expanded data set. The right border of the estimation window is fixed to the current value of i, since the most recent example represents the current state of the modeled system best. Since n increases with each example, the whole window is shifted and therefore older data is excluded.

The difficulty with this approach lies in determining the window size w that results in a model with a small prediction error. An analytic solution requires knowledge about the characteristics of the drift of the modeled process, i.e., rate of drift or cycle length. In our application setting—single-lot semiconductor manufacturing—this knowledge is often not available. Additionally, drift may be discontinuous, e.g., abrupt and random, which causes the optimal window size to change as well within the lifetime of a prediction model. Therefore, we need an adaptive solution, where the window size is set as part of the training process. Changes in the characteristics of the drift can then be compensated. Since an exact determination of the window size is not possible in our setting, we avoid determining one best-effort window size. Instead, we use several windows of different size for training several regression models that are maintained according to predictive performance, i.e., when a model is not sufficient anymore, it is replaced.

To maintain this set of models, the following statistics are needed for each model $model_m$. First of all, there is l_m, the left border of the window of examples for estimating the associated model. Second, there is the number of positive pos_m and negative neg_m predictions that were made with the associated model. The counters are based on tests performed on new examples and a local threshold t_e for the acceptable error. Finally, there is the weight w_m, which is used for calculating overall predictions from the ensemble. We describe the statistics more detailed in the following. The statistics and the model form a member M_m that can be uniquely identified by an index m. A set of members forms an ensemble $E = \{M_m\}, m = 1, \ldots, |E|$.

Final edited form was published in "Machine Learning and Data Mining in Pattern Recognition. 6th International Conference. Leipzig 2009", S. 221-235, ISBN 978-3-642-03070-3
https://doi.org/10.1007/978-3-642-03070-3_17

Algorithm 1. Training algorithm for ensemble regression.

```
Require: $e_{max}$: the maximum number of members in the ensemble
Require: $t_e$: threshold for prediction error
Require: $t_r$: threshold for the performance ratio
1: $E \leftarrow \emptyset$; $D \leftarrow \emptyset$; $n \leftarrow 1$
2: for each new $d$ do
3:     $n \leftarrow n + 1$; $d_n \leftarrow d$
4:     $D \leftarrow D \cup d_n$
5:     for each $M_m \in E$ do
6:         if not is_stable($model_m$) then
7:             $model_m \leftarrow$ train($l_m$, $n$); next
8:         end if
9:         $err \leftarrow error(y_n, predict(model_m, x_n))$
10:        if $err < t_e$ then
11:            $pos_m \leftarrow pos_m + 1$
12:        else
13:            $neg_m \leftarrow neg_m + 1$
14:        end if
15:        if $pos_m(pos_m + neg_m)^{-1} < t_r$ then
16:            $E \leftarrow E \backslash M$
17:        else
18:            $model_m \leftarrow$ train($l_m$, $n$)
19:        end if
20:    end for
21:    $pos_T = \sum_{m=1}^{|E|} pos_m$
22:    for each stable $M_m \in E$ do
23:        $w_m \leftarrow pos_m pos_T^{-1}$
24:    end for
25:    if $|E| < e_{max}$ then
26:        $E \leftarrow E \cup$ new_member($d_n$)
27:    end if
28: end for
```

We now present the algorithm (see Algorithm 1) that is used to maintain an ensemble by describing several iterations and the resulting state of the ensemble after each of them. The example is based on the data set depicted in Figure 1, which contained an abrupt drift after the fifth of ten examples. As initialization, we start with an empty data set D and an empty ensemble E. The main loop of the algorithm (line 2) is executed whenever a new example d is acquired. The index for the most recent example n is increased by one and the example is added to the data set D as d_n. Hence, in the first iteration, $D = \{d_1\}$. Since the ensemble is empty, lines 5 to 24 have no effect and the first member is added to the ensemble in line 26. Members are added to the ensemble as long as the maximum number of members e_{max} is not reached. This is a technical parameter to limit the required computational resources. The method new_member(d_n) initializes the statistics of the new member and estimates the associated regression model from a minimal window containing only the most recent example d_n.

The current value of n will be stored as left border l_m of the estimation window. After this first iteration, $E = \{M_1\}$.

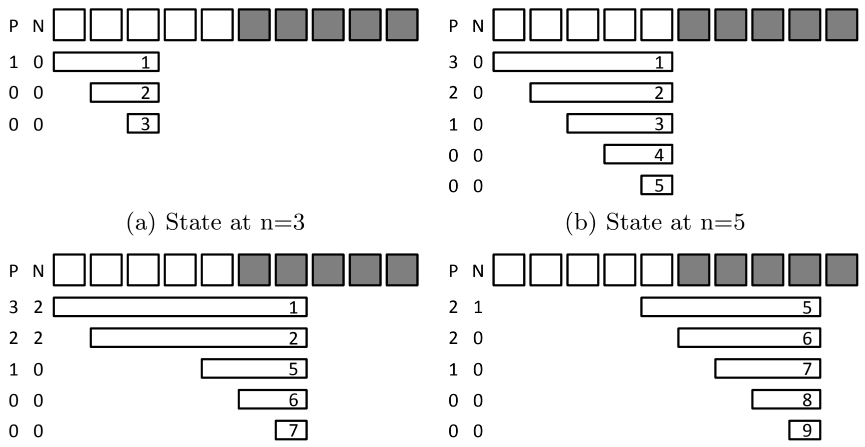
When the next example d is acquired, it is added to the total data set as d_2 and therefore $D = \{d_1, d_2\}$. Since the ensemble now contains one member, line 6 is executed for member M_1. The method is_stable($model_m$) tests whether a member may be considered as stable, i.e whether the estimation window contains enough examples. The exact definition for stability depends on the used regression method. For linear regression, a stable fit requires at least as many examples as coefficients to be estimated. In our example, we use linear regression and two coefficients have to be estimated. Hence, M_1 is still unstable in step two, because the current window for estimation contains only one example. In line 7, the model is now estimated from a larger window $[l_m, n] = [1, 2]$. In general, the window associated with a member grows by one example for each example acquired. After this training, all other steps involving M_1 are skipped. The second iteration of the algorithm is completed after adding the second member M_2, which is associated with a model that was estimated from d_2. Hence, $E = \{M_1, M_2\}$.

In the third step, M_1 has finally become stable and therefore the following steps from lines 9 to 19 are performed on it. First, M_1 is tested by using the associated model $model_m$ to predict the known label of the new example (line 9). Then, the prediction error is calculated using a given metric *error*, e.g., the *root mean squared error*. We use this testing scheme, since the most recent example represents the current underlying function best and since the most recent example has not been used in model estimation so far. In our example, no drift has occurred in this third iteration. Therefore, the error is $err = 0$.

In lines 10 to 14, the error determined in the test is classified as positive or negative and the according counter (pos_m or neg_m) is incremented. The parameter t_e that defines this local error threshold is intuitive to set. A larger threshold will result in more predictions classified as positive. In some application areas, like semiconductor manufacturing, this parameter is derivable from application knowledge, like quality specifications. In our example, the number of positive tests for M_1 is $pos_1 = 1$ in the third iteration, since $err = 0$ and because we set $t_e = 1$.

In line 15, the test for eviction is performed. We propose a threshold-based criterion, where a member is evicted when $pos_m(pos_m + neg_m)^{-1} < t_r$, i.e., when the ratio of positive test predictions to all test predictions falls below a given value $0 \leq t_r \leq 1$. This criterion has several important properties. It is resistant to outliers, since single negative predictions, e.g., those caused by noise, have limited influence when a member has reached a stable state. A member may even recover from a series of negative predictions, e.g., in cases of cyclic drift. Most importantly, members that have accumulated a large number of positives, and therefore have been in the ensemble for a long time, are harder to evict than members with fewer positives, since they can accumulate more negatives before fulfilling the eviction criterion. Therefore, mature members are favored but can still be evicted if a permanent drift occurs that makes them unusable.

Final edited form was published in "Machine Learning and Data Mining in Pattern Recognition. 6th International Conference. Leipzig 2009", S. 221-235, ISBN 978-3-642-03070-3
https://doi.org/10.1007/978-3-642-03070-3_17



(a) State at n=3 (b) State at n=5

(c) State at n=7 (d) State at n=9

Fig. 2. Window size, number of positive and negative predictions of the ensemble members. Data set from Figure 1.

Note that the prediction weight of such a member can decrease even while it is still in the ensemble, which accounts for the fact of decreasing predictive performance. In our example, M_1 stays in the ensemble since the ratio of positive test predictions to all test predictions is one and we set $t_r = 0.5$. Members with sufficient predictive performance remain in the ensemble and the associated model is trained on the window $[l_m, n]$, thereby including the new example. The threshold on the performance ratio influences how fast the ensemble reacts to drift, whereby a larger ratio implies that few negative predictions are tolerated and eviction may happen early.

Figure 2 depicts the further evolution of the ensemble. White squares represent examples from the first underlying function, while gray squares represent examples from the second underlying function. The examples are ordered from left to right with ascending index. The rectangles below depict the window that each member in the ensemble used to estimate its associated model. The number in the rectangle is the member identification m and at the left, the column P shows pos_m, while N shows neg_m. Figure 2(a) shows the state of the ensemble after the completion of the third iteration. Three members have been added so far. M_1 uses the largest window and is the only stable member.

Figure 2(b) depicts the state after the fifth example has been incorporated. There are five members now, which means that the set e_{max} is reach. No further member will be added. Each member has been estimated from a different window, with the first member still using the largest window. It also has the largest number of positive predictions, since no drift has occurred yet. Member M_5 is still considered unstable. Member M_4 is stable but has not been tested yet on a new example, since the fourth and fifth examples were used in parameter estimation. No member has made any negative predictions.

Figure 2(c) depicts the state after the seventh example has been incorporated. There are again five members, but members M_4 and M_3 have been replaced by the new members M_6 and M_7. Member M_4 was evicted in the previous step, while member M_3 was evicted in the seventh step. Interestingly, member M_5 is still in the ensemble, although its window covers an example from the previous underlying function.

Figure 2(d) depicts the state after the ninth example has been incorporated. The members M_1 and M_2 have been evicted, since they were unable to perform any more positive predictions, thereby dropping below the required performance ratio. Member M_5 remained in the ensemble, since more and more current examples helped to improve the fit of the underlying function.

So far we have not discussed the determination of overall predictions $\hat{y}$ from the ensemble. $\hat{y}$ is determined by calculating $\hat{y}_m = predict(model_m, x)$ for each member in the ensemble and then calculating a weighted average $\hat{y} = \sum_{m=1}^{|E|} w_m \hat{y}_m$, whereby the weights w_m are determined as part of Algorithm 1 in two steps. First, in line 21, the total number of positive predictions $pos_T = \sum_{m=1}^{|E|} pos_m$ of the members in the ensemble is determined. Then, this sum is used to determine the prediction weight of each member in lines 22 to 24. It is defined as the ratio of positive predictions of the member to the sum of positive predictions of all members: $pos_m {pos_T}^{-1}$. Hence, if a member has made only a small number of positive predictions, while other members have made large numbers of positive predictions, its weight will be low and vice versa. Mature members are therefore favored. Most importantly, the weight of a member can decrease if it does not make positive predictions while other members do (e.g., newer members).

4 Evaluation

In this section, we evaluate our approach on several types of synthetic data as well as a real-world data set. Our goal is to determine the prediction error when using different approaches to estimate regression models from drifting processes. We compare *ensemble regression* using different weighting schemes with the *baseline* (performing regression on the complete data set) as well as with the sliding *window* approach. A simulation environment with these algorithms was implemented in `R`. We used `lm`, included in the standard `stats` package, to estimate linear regression models. Since we restricted our experiments to the use of linear models in the ensembles, we also restricted our experiments to linear underlying functions. This ensures that the models are capable of representing the governing functions and no error can be induced from an improper functional form. Our systematic evaluation is therefore only valid for data sets from drifting processes that are governed by a linear dependency. However, we also present results based on a non-linear, real-world data set that indicate that our approach performs well in that setting, too.

Formally, our synthetic data sets were generated using a governing function $y = f(x) = \sum_{k=1}^{K} w_k(t) x_k + N(0, \sigma^2)$ where K is the number of independent variables, $w_k(t)$ is the time-dependent weight of attribute x_k and $N(0, \sigma^2)$ is a normal random variable that represents measurement noise. In this setting, drift manifests itself in changing values of $w_k(t)$, while the functional form of $f(x)$ stays fixed. We define three different functional forms of $w_k(t)$:

Linear Drift. $w_k(t) = at + b$, where slope a determines the speed of the drift. We fix b since it has no influence on the prediction error, while a is varied. Results are presented in Figures 3(a) and 3(b).

Autoregressive Drift. $w_k(t) = w_k(t-1) + N(\mu, \sigma^2); w_k(0) = N(\mu, \sigma^2)$, where $w_k(t-1)$ is the weight determined for the previous example and $w_k(0)$ is the random starting weight. $N(\mu, \sigma^2)$ is a normally distributed random variable with mean μ and variance σ^2. In our experiments, we vary μ while we fix σ. Results are presented in Figures 3(c) and 3(d).

Mixture Drift. $w_k(t) = w_k(t-1) + a + I(p)N(\mu, \sigma^2); w_k(0) = N(\mu, \sigma^2)$, where $w_k(t-1)$ is the weight determined for the previous example, $w_k(0)$ is the random starting weight and a is the slope. $I(p)$ yields either 1 with probability p or 0 with probability $(1-p)$, i.e., $I(p)$ follows a binomial distribution with n and k fixed to 1. $I(p)N(\mu, \sigma^2)$ represents abrupt drift of random magnitude. This drift contains a fixed linear and a stochastic component and is therefore more realistic than the other two. In our experiments, we vary a and μ while we fix σ and p. Results are presented in Figures 3(e) and 3(f).

For each of these three types of drift, we fixed a set of values for the parameters that are varied, which resulted in 97 parameter sets. For each of them, 10 data sets were created, with each data set containing 1,000 examples. The values of the x_n were drawn from a uniform distribution, while the $w_k(t)$ were calculated and the weighted sum yielded the associated y_n as defined above. We then simulated the repeated estimation of a regression model from each of these data sets and recorded the root squared error for overall predictions from the ensemble. The results are distributions of the error per regression approach over all parameter sets and they are depicted in Figure 3 with one row of box plots per drift type. Since the variance in the error differs significantly, the results are shown on two scales. The maximum displayed root squared error RSE_{max} is $1,000$ and 100 in the left and right column respectively.

In each diagram, five box plots are shown. The labels on the x-axis correspond to the following five approaches:

bl. This is the baseline approach of estimating a regression model from the complete data set whenever a new example is added. This approach is clearly the worst because of the large median error and large error variance for all three drift types. This approach is neither accurate nor stable. However, even for this approach small errors were observed, which can be explained by the dependence of the error on the value of the independent variables x_n. This was illustrated in Figure 1, where the smallest error for the model estimated from the whole data set is zero at the intersection with the most recent underlying function.

win. The windowing approach achieved very good results on the linear data set, where the median and the first quartile are the lowest of all approaches.

This changes for the autoregressive data set where the median and the first quartile are equal to our proposed approach (labeled as `ens-g`). For the mixture drift, window is the second-best approach. These results also confirm our assumption that a restriction of the data set used for estimation can result in a smaller prediction error.

ens-e. This approach is identical to our proposed ensemble regression scheme, except that it does not use the calculated weights. Instead, the overall predictions are determined as an *equally weighted* linear combination. The approach outperforms baseline on the linear and the autoregressive data sets, but it is worse on the mixture drift data set. This can be explained by the relatively strong changes caused by the abrupt component in the mixture drift, which invalidates members very fast. Since it takes some time until the members are evicted, they contribute a large error that is propagated into the final prediction, since all weights are equal.

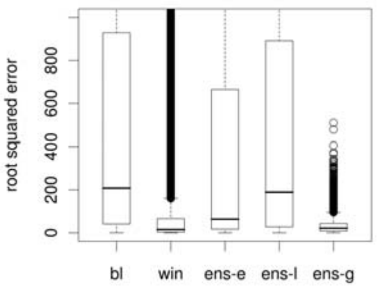
ens-l. This approach is identical to our proposed ensemble regression scheme, except that it employs the *local* performance ratio $pos_m(pos_m + neg_m)^{-1}$ as weight, which seems to be an intuitive choice. However, the variance of the prediction error is even larger than when using equal weights.

ens-g. This is the approach proposed in this paper and it uses the *global* weight $pos_m pos_T^{-1}$. It yields the best overall performance, since it has the lowest error variance for all three drift types. For linear and autoregressive drift, the median error is slightly worse or equal to the median error of the windowing approach. However, the median error is significantly lower for the mixture drift.

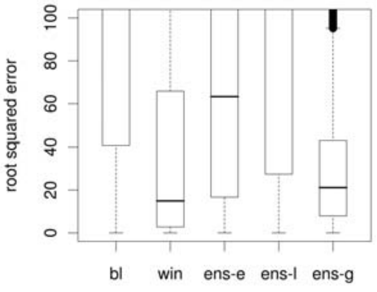
In a second set of examples, we examined the influence of the maximum number of ensemble members e_{max} and the eviction ratio t_r on the distribution of the prediction error. We executed the simulation using different values for e_{max} and t_r on the data sets of all three drift types. The results were similar with respect to the observed influence. We therefore show only the error distribution based on the linear drift data sets.

In Figure 4(a) the error distribution using different values of e_{max} is depicted. It can be seen that e_{max} has only a limited influence on the prediction error, although the median error and the error variance seem to increase slightly with increasing e_{max}. However, we cannot conclude that using fewer members is a good strategy, since a smaller value of e_{max} causes an increased chance of a completely unstable ensemble. This situation occurs when all stable members are evicted at once and new unstable members—being introduced one by one—still have to gather examples to be regarded as stable. In our experiments, we defined members with a window size smaller than 4 as unstable, since we used linear regression and since 3 coefficients had to be estimated. Otherwise, the underlying equation system would be under-determined and the resulting model insufficient. Using larger e_{max} can reduce the likelihood of this situation. We can only conclude that future work is required for application settings were this behavior of our approach is unacceptable. In the first set of experiments, depicted in Figure 3, e_{max} was set to 10.

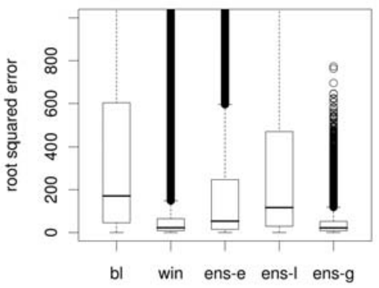
Final edited form was published in "Machine Learning and Data Mining in Pattern Recognition. 6th International Conference. Leipzig 2009", S. 221-235, ISBN 978-3-642-03070-3
https://doi.org/10.1007/978-3-642-03070-3_17



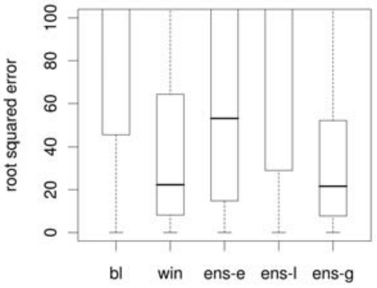
(a) Linear drift; $RSE_{max} = 1,000$



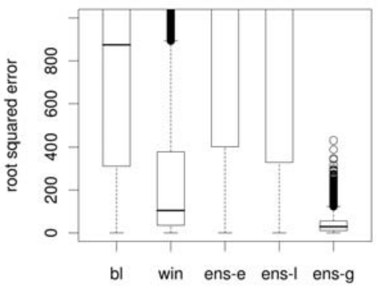
(b) Linear drift; $RSE_{max} = 100$



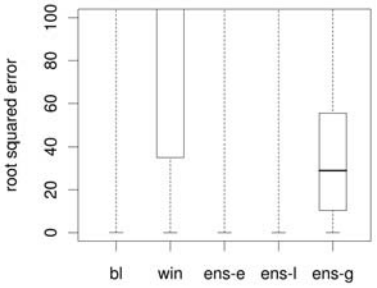
(c) Autoreg. drift; $RSE_{max} = 1,000$



(d) Autoreg. drift; $RSE_{max} = 100$



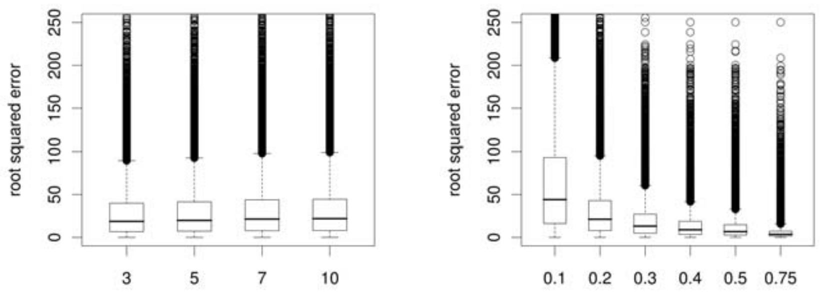
(e) Mixture drift; $RSE_{max} = 1,000$



(f) Mixture drift; $RSE_{max} = 100$

Fig. 3. Distribution of root squared error for different types of drift

Final edited form was published in "Machine Learning and Data Mining in Pattern Recognition. 6th International Conference. Leipzig 2009", S. 221-235, ISBN 978-3-642-03070-3
https://doi.org/10.1007/978-3-642-03070-3_17



(a) Influence of e_{max} (b) Influence of t_r

Fig. 4. Parameter influence on the distribution of root squared error

In Figure 4(b), the influence of the eviction ratio is depicted. Obviously, a higher value of t_r results in a lower median error and a decreased error variance. In the first set of experiments, depicted in Figure 3, t_r was set to 0.2. We did not study the impact of t_e further, since—as we argued in Section 3—if possible, it should be set using application knowledge. In the first set of experiments, we fixed it to be 0.1% of the total spread of the values of y, which translates to an absolute value of $t_e = 5$.

In Figure 5, we report results on a real-world data set from semiconductor manufacturing. To prevent any deductions about the underlying processes, we just report signed, relative errors. The data set is high-dimensional, non-linear and contains a relatively small continuous drift. The baseline approach yields a median error of about 1.3%, which is reduced to −0.1% by using ensemble regression . The model yielded by ensemble regression is therefore much better centered, while the error variance is identical.

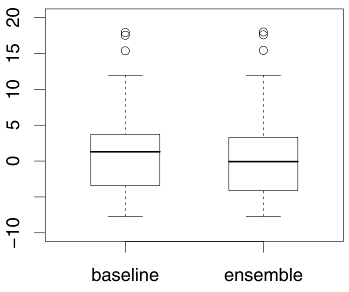


Fig. 5. Distribution of relative error for a real-world data set

5 Related Work

In this paper, we consider the problem of estimating a regression model from data of a drifting process, whereby the most recent underlying function is to be approximated. To the best of our knowledge, this problem has not been considered so far. Current approaches in industrial applications can often use specially designed control models [5] to correct process drift and therefore eliminate the cause of the type of error investigated in this paper. Hence, standard regression techniques can be applied there. However, these approaches are not feasible in our application setting because of the indeterministic nature of drift and still ongoing theoretical research on the underlying physical phenomenons [1,2,3].

Concept drift is a related problem in classification and was introduced in [6]. Concept drift occurs when a *hidden context* exists and changes while examples are gathered. Changes in the hidden context can induce changes in the target concept. Analogous, in our problem setting, the underlying function changes and therefore influences the regression models estimated from a yielded data set. A number of approaches for handling concept drift exist. In [7] a system is presented that uses a sliding window of examples to construct concept descriptions using a description language. In [8] an algorithm for mining decision trees from a sliding window on a continuously changing data stream was presented. Adaptation to concept drift is performed by replacing subtrees or by building alternative subtrees. All these approaches enhance particular classification methods to be able to adapt to concept drift. In [9,10] an approach is presented where ensembles of base classifiers are built from sequential chunks of data. Using such jumping windows is not appropriate in our application context, since the newest examples always have to be incorporated. In [11] dynamic weighted majority was proposed for tracking concept drift. It uses an ensemble of base classifiers that are built from growing windows. However, the different learning task made it necessary to use a different eviction criterion and weighting scheme. Regarding the latter, members start with a fixed weight and each false prediction is penalized by reducing the weight by a constant fraction and normalizing the weights afterwards. In this approach, new members have a larger influence than in our approach, where the predictive weight is gathered slowly.

6 Conclusion and Outlook

Regression is an important, widely used tool, although the implicit assumption of a stationary joint distribution may not be met. Using regression on data from drifting processes results in a model that does not represent the current state of the process well. In application areas like semiconductor manufacturing, this induces an error that can be critical. In this paper, we presented ensemble regression, an approach that uses a set of regression models estimated from different ranges of the complete data set. The ensemble is maintained according to the predictive performance of the members, thereby yielding a low prediction error and a low error variance. In future work, we plan to widen our empirical study to the examination of the effects of drift and the performance of our approach on non-linear dependencies as well. This will be accompanied by the use of nonlinear and nonparametric regression techniques as base learners. Additionally, we plan to investigate the use of sophisticated heuristics for adding new members to the ensemble.

References

1. Spitzlsperger, G., Schmidt, C., Ernst, G., Strasser, H., Speil, M.: Fault detection for a via etch process using adaptive multivariate methods. IEEE Transactions on Semiconductor Manufacturing 18(4), 528–533 (2005)
2. Bunday, B.D., Bishop, M., Donald, W., McCormack, J., Villarrubia, J.S., Vladar, A.E., Dixson, R., Vorburger, T.V., Orji, N.G., Allgair, J.A.: Determination of optimal parameters for cd-sem measurement of line-edge roughness. Metrology, Inspection, and Process Control for Microlithography XVIII 5375(1), 515–533 (2004)
3. Yue, H.H., Qin, S.J., Wiseman, J., Toprac, A.: Plasma etching endpoint detection using multiple wavelengths for small open-area wafers. Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films 19(1), 66–75 (2001)
4. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. Springer, Heidelberg (2003)
5. DiRaddo, R., Girard, P., Chang, S.: Process drift and model-based control of forming operations. In: American Control Conference, 2002. Proceedings of the 2002, vol. 5, pp. 3588–3593 (2002)
6. Schlimmer, J.C., Granger, R.H.: Incremental learning from noisy data. Machine Learning 1, 317 (1986)
7. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23(1), 69–101 (1996)
8. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD 2001: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 97–106 (2001)
9. Street, W.N., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: KDD 2001: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 377–382. ACM, New York (2001)
10. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: KDD 2003: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 226–235. ACM, New York (2003)
11. Kolter, J., Maloof, M.: Dynamic weighted majority: a new ensemble method for tracking concept drift. In: Third IEEE International Conference on Data Mining (ICDM), November 2003, pp. 123–130 (2003)