DEAL: Data-Efficient Active Learning for regression under drift

Béla H. Böhnke[0000−0002−3779−3409] bela.boehnke@kit.edu, Edouard Fouché[0000−0003−0157−7648], and Klemens Böhm[0000−0002−1706−1913]

Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract. Current work on Active Learning (AL) tends to assume that the relationship between input and target variables does not change, i.e., the oracle is static. However, oracles can be stream-like and exhibit concept drift, which requires updating the learned relationship. Standard drift detection and adaption methods rely on constantly observing the target variables, which is too costly in AL. Current work on AL for regression has not addressed the challenge of frequently drifting oracles. We propose a new AL method that estimates its error due to drift by learning statistics about how often and how severe drift occurs, based on a Gaussian Process model with a time-variant kernel. Whenever the estimated error reaches a user-required threshold, our model measures the target variables and recalibrates the learned relationship as well as the drift statistics. Our drift-aware model requires up to 20 times fewer measurements than widely used methods.

Keywords: Concept Drift · Active Learning · Regression.

1 Introduction

Active Learning (AL) refers to data collection methods aimed at estimating the relationship between input and target variables under the assumption that measurement of the variables is expensive. Therefore, AL seeks to estimate the relationship with as few measurements as possible. Our setting, in particular, is the one of stream-based AL [26]. Stream-based AL assumes input variables arrive as a continuous stream X, e.g., sensor data like temperature and humidity acquired for real-time environmental monitoring. A *learner* observes the stream at time t_i, resulting in one *potential query* x_i. The learner then decides whether to perform a *query*, i.e., to measure the target variable Y like soil quality for agriculture, or not. While the cost of observing the stream X is negligible, measuring the target is expensive. Many scenarios share this assumption [26], such as industrial process control, resource allocation for environmental monitoring, or demand forecasting for energy management [4].

AL methods often omit details of how the target variable is measured. This abstraction is called *oracle*. In practice, such oracles can *drift*: Evolving, unobservable environmental parameters can affect the relationship between input and

target variables over time. This phenomenon of changing relationships between variables is known as *concept drift (CD)*.

CD poses a significant challenge to statistical models, requiring frequent recalibration to maintain estimation error below a user-required threshold. However, conventional methods for drift correction, like continuous model recalibration or drift detection by monitoring the target variables, are impractical for AL due to the high cost of observing the target variable.
(1) The first challenge arises from this mismatch between the goal of AL of reducing measurements and the need for additional measurements for model recalibration, which requires an AL method that can adjust for drift without continuously monitoring the drifting variables. While such drift-correcting AL methods exist for classification tasks, many real-world scenarios involve continuous target variables requiring regression models. These regression scenarios lack adequate AL methods [25, 31], highlighting a gap in current AL research.
(2) The second challenge becomes clear when comparing classification and regression tasks under drift. Classification only has to monitor a fixed number of class distributions. In contrast, regression has to estimate the model error due to drift for each point in a continuous input space, target space, and time.
(3) The final challenge is an appropriate selection of the measurement time based on the estimated error, ensuring that the error due to drift remains below a user-required threshold.

We contribute by proposing DEAL, the first AL regression method that learns data efficiently under oracles that exhibit frequent drift while keeping the estimation error below a user-required threshold. DEAL estimates its own error due to drift, using a Gaussian process model (*GP*) with a time-variant kernel. The *GP* learns the *drift behavior*, i.e., the time-dependent distribution of the target variable. Figure 1 shows that DEAL queries the oracle whenever the estimated error exceeds a user-required threshold. DEAL uses the new data to recalibrate, and combines it with the old data to update the learned drift behavior. In this way, DEAL minimizes the number of measurements required for drift correction.

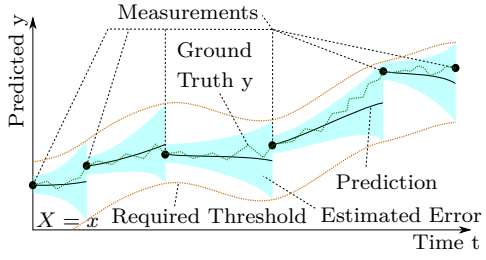


Fig. 1: DEAL estimates its error and measures once the error reaches the user-required threshold. The input value x of the data stream X is kept constant for better visualization, the target y still varies due to frequent incremental drift.

We evaluate DEAL against multiple baselines, on multiple drift-affected time series, and provide the code[1] for reproduction.

2 Related Work

To our knowledge, existing stream-based AL methods for regression (e.g., [1, 10, 12, 25, 28, 29, 31, 32]) do not consider drift, i.e., the oracle is assumed to be static. Such methods stop learning once the regression model performs well and never start learning again, even if the model performance decreases due to drift. We show this undesirable behavior in our experiments.

There exist AL methods that consider drift [13–16, 18, 19, 22, 24, 27, 33, 35–37], but they are restricted to classification. Further, to apply [14, 18, 22, 24, 35, 37] a user needs to set a measurement budget and additional parameters without knowing drift behavior, and the resulting estimation error. Improper set parameters lead to either too costly or inaccurate models. Drift behavior that changes causes the same problem because the methods cannot adapt. Further, those methods primarily monitor the distribution of input variables per class. This is intractable for regression and impedes the transfer to the regression case.

There exist change detection methods that can adapt to changing drift behavior. For example, AAIL [24] detects changes in the input variables and measures the target variable at each change. While the authors claim that AAIL adapts to concept drift (CD), it can only adapt to covariance shift. Covariance shift only refers to changes in the distribution of the input variables, which is cheap to observe, while CD describes a change in the relationship between the input and target variables. AAIL can be adapted to regression tasks, and we include an adaptation in our experiments. Drift detection approaches not designed for AL, as surveyed in [7, 9, 20, 34], typically assume that the target variables are cheap to observe, which violates a basic assumption of AL. Thus, such approaches require additional methods for strategic under-sampling of the target variable, essentially what the existing AL techniques for drifting oracles do.

In summary, the only methods viable in practice to deal with drift in AL regression is some form of under-sampling with consecutive measurements at a user-defined frequency, as in [4–6]. A poor frequency choice leads to missed drift or higher measurement costs. We use such a method as one of several baselines.

3 Problem statement and Notation

Random Functions $F : x \mapsto F(x)$ associate each value x_i of a variable $x \in \mathbb{R}$ with a random variable $F(x_i)$ [3]. As such, a random function extends the notion of random variables to a continuous function space. *Sample Functions* $f(x) \leftarrow F(x)$ are functions $f : x \mapsto f(x)$ drawn from a random function $F(x)$.

Stochastic Processes S or random processes are random functions $S : t \mapsto S(t)$ where $t \in \mathbb{T}$ is interpreted as time with domain $\mathbb{T} = \mathbb{R}_+$ [3]. *Brownian*

[1] https://github.com/bela127/alsbts-experiments

Motions B are stochastic processes $B : t \mapsto B(t)$ [3] characterized by random increments $\delta B(\delta t) = B(t + \delta t) - B(t) \forall t$, where these increments follow the normal distribution $\delta B(\delta t) \sim \mathcal{N}(0, \delta t)$. *Gaussian Processes* ($GP$) are stochastic processes defined as $GP(x) \sim \mathcal{N}(m(x), v(x))$, illustrating that the random function $GP(x)$ comes from a normal distribution with mean $m(x)$ and variance $v(x)$ both depending on x [17]. One often uses GPs as a probabilistic prior over functions.

Kernel Functions $k(x, x')$, also called covariance functions, can define a GP instead of using a mean and variance function. *The Radial Basis Function (RBF) Kernel* $k(x, x') = v \cdot \exp\left(-\frac{(x-x')^2}{2 \cdot l^2}\right)$, has two parameters: $v \in \mathbb{R}_+$ (target variance) scaling the target of the random function, and l (length scale) determining function smoothness. *The Brownian (Bridge) Kernel* $k(t, t') = v_b \cdot \min(t, t')$, with points in time t, t' has a variance parameter $v_b \in \mathbb{R}_+$, which translates to the drift speed, i.e., it scales $\delta B(\delta t)$ by v_b [17].

Observed Data Streams or time series are sample functions $x(t) \leftarrow X \mid t \in \mathbb{T}$. Here, X is a random input variable, called the *stream*, and $x(t)$ provides a distinct value $x_i = x(t_i)$, drawn from X at a point in time t_i. *Covariance Shift* is present if the distribution of the stream changes over time. Such a time-dependent stream is, in fact, a random process, and one writes $X(t)$.

Concepts, i.e., the relations between input and target variables, are represented as random functions $C : x \mapsto C(x) = Y$. If a concept depends on time, it is represented as $C : (x, t) \mapsto C(x, t)$. By definition, *Concept Drift (CD)* is present if $\exists t_1, t_2 : C(x, t_1) \neq C(x, t_2)$, where $t_1 \neq t_2$ are two points in time [7].

Oracles are models for data-generating processes with concept C. A *query* is a request $q(t_i) \leftarrow Q$ to an oracle to return the current value $y_i = c_i \leftarrow C(q(t_i), t_i)$ of the target variable. Q is the stream of performed queries. The input stream X is the stream of possible queries.

Stream-based active learning (AL) [26] iteratively observes the value x_i of the stream X and only performs a query if the uncertainty v_{est} of the prediction y_{est} exceeds a certain user-required threshold v_{target}. The learner then recalibrates the model using the resultant measurement y_i. This is known as *uncertainty sampling*, which is a typical way to decide whether to query or not.

Algorithm 1 The Common vs Our Adapted Stream-Based AL-Cycle

```
1: procedure COMMON(v_target, MODEL)
2:     while running do
3:         x_i ← X
4:         y_est :=MODEL.ESTIMATE(x_i)
5:                 ▷ y_est only for evaluation.
6:         v_est := MODEL.VARIANCE(x_i)
7:         if v_est >= v_target then
8:                 ▷ Uncertainty sampling.
9:             y_i := ORACLE.QUERY(x_i)
10:            DATAPOOL.ADD(x_i, y_i)
11:            MODEL.TRAIN(DATAPOOL)
```

```
1: procedure ADAPTED(v_target, MODEL)
2:     while running do
3:         x_i, t_i ← X,TIME()
4:         y_est := MODEL.ESTIMATE(x_i, t_i)
5:                 ▷ y_est only for evaluation.
6:         v_est := MODEL.VARIANCE(x_i, t_i)
7:         if v_est >= v_target then
8:                 ▷ Uncertainty sampling.
9:             y_i := ORACLE.QUERY(x_i, t_i)
10:            DATAPOOL.ADD(x_i, t_i, y_i)
11:            MODEL.TRAIN(DATAPOOL)
```

4 Our Method: DEAL

4.1 The Adapted Stream-Based AL Cycle

We slightly modify the Stream-based active learning cycle (see Algorithm 1), so that in addition to observing the stream X, we also observe the current time t_i. DEAL's estimation model then takes the time t_i into account when estimating its variance v_{est} to include the additional uncertainty caused by drift (Lines 3 to 6). Whenever the uncertainty reaches the threshold, DEAL recalibrates (Line 9 to 11) with a new data point (x_i, y_i). Here, the drifting oracle provides y_i.

To estimate the uncertainty caused by drift, we require one assumption about the drift behavior: We assume that drift occurs frequently, i.e., within a time series with length t_{end}, at least n_c changes occur. Here n_c needs to be large enough to learn sufficient statistics of the drift behavior.

4.2 Our Drift-Aware Estimation Model

In contrast to methods like discussed in [14, 18, 22, 37], which perform measurements without modeling the drift behavior, we are the first to learn statistics about the drift behavior. These statistics enable us to measure in adaptive time intervals in which drift of a certain magnitude may occur. We derive the statistics from a Gaussian process model (GP) according to the following prior:

Definition 1. *The Brownian drift prior is given by* $C(x,t) = I(x) + W(x) * B(t)$, *with a Brownian motion prior* $B(t)$ *as drift behavior, and random function priors* $I(x)$ *and* $W(x)$ *independent of any drift thereby constant over* t. *Here,* $W(x)$ *is a weighting term that defines the impact of* $B(t)$ *on* $C(x,t)$ *at a position* x.

The intuition behind using a Brownian prior $B(t)$ for the drift behavior is that the combination of many, small, random, and independent external influences results in a combined Brownian overall drift. Further, this model can capture drift with larger changes after a random time, as long as changes occur frequently. Such frequent drift is common in practice, like: (1) Drift due to displacement of machine elements caused by vibration. (2) Drift in large networks, such as the electrical grid, where nodes can (dis)connect from the network at random times.

We instantiate the GP with a kernel composition according to the three components $I(x)$, $W(x)$, $B(t)$, from Definition 1, i.e., one kernel per component. We model the drift behavior $B(t)$ with the Brownian kernel [17]. Because of its universal approximation property [21], we use distinct RBF kernels to model $I(x)$ and $W(x)$. In general, the choice of kernels is a parameter that one can easily tune to match prior knowledge about the data or drift. For the sake of generality, we stick to our choice in this study. Further, we enforce that both RBF kernels have the same length scale and variance, which reduces the number of learnable parameters and makes learning more stable. This reduction assumes similar smoothness of $I(x)$ and $W(x)$ and the resulting learned function. DEAL's parameters are Brownian variance v_b, RBF variance v_r, RBF length scale l_r.

The only hyperparameter DEAL takes is a user-required threshold v_{target}. DEAL trains according to Algorithm 1. Every time the variance v_{est} estimated by DEAL becomes greater than v_{target}, DEAL measures the target value (Line 9) and adds it to the training set. DEAL then estimates the most likely kernel parameters with GPy's [8] gradient-based maximum likelihood optimizer (Line 11). We use the standard optimizer configuration with 5 restarts, 4 times with random kernel parameters, and once with the most likely kernel parameters from the previous iteration. In the first iteration, we initialize the kernel parameters v_b, v_r, and l_r with random values and use an initial training set of measurements from the first 10 time steps. We use this initialization for each baseline as well.

In AL, training complexity tends to be neglected, because the oracle usually is much more expensive than the active learning decision-making. We use the standard *GP* model from GPy, with a complexity of $O(n^2)$, where n is the training set size. Since n is kept small, the actual runtime is consistently low.[2]

5 Experimental Design

5.1 Baselines

We consider the following baselines as competitors for DEAL:

Consecutive Measurement (CM): This approach carries out measurements at regular user-specified time intervals of size δt_{meas}. The approach is sensitive to δt_{meas} and does not adapt to the data-generating process. In our experiments, we evaluated values of $\delta t_{meas} \in [1, 20]$, with logarithmic increments.

Classic AL (CAL): This standard AL approach with uncertainty sampling yields an estimate and an estimation variance. It uses a Gaussian process (*GP*) with an RBF kernel and kernel parameters length scale l and variance v. Unlike CM, the *GP* can automatically adjust these parameters using maximum likelihood estimation in the same way DEAL does. But unlike DEAL, this approach does not model the data behavior over time, i.e., it assumes a static oracle. To obtain a strong baseline, we initialized the *GP* with kernel parameters identical to the parameters of the ground truth data, see 5.2. Given enough data, the *RMSE* of such an approach approaches zero on a stream with no noise or drift.

Change Ideal (CI): This approach is an adaptation of AAIL [24], which uses change detection on the input variables X. Whenever CI detects a change in X, it measures y. To make our evaluation independent of any specific detector and obtain a strong baseline, we simulate an "ideal" detector. It uses the (normally unobservable) ground truth to correctly and immediately report any change in X. This baseline has no configurable parameters.

Change Error (CE): There are three types of errors in change detectors: undetected change, detection without an actual change, and delayed detection. To study their effect, we created variants of CI using an imperfect "pseudo"

[2] Note one can reduce the complexity of DEAL down to $O(n \cdot i)$ (with learning epochs $i \ll n$), by using gradient-based *GP* models and batch training. Further, one can cap the number of measurements used for training, reducing the factor n to a constant.

change detector which lets us control each error type with three parameters: p_{wrong} is the proportion of spurious detection at any time. p_{miss} is the chance of discarding a correct detection. std_{offset} is the standard deviation of a normal distribution. For any detected change, we take the absolute value of a random point from this distribution as an offset to delay detection.

In our experiments, we vary these parameters independently according to the given interval and step size while keeping the others at the given default value:

	Interval	Step Size	Default
p_{wrong}	$[0, 0.10]$	0.005	0.015
p_{miss}	$[0, 0.80]$	0.05	0.015
std_{offset}	$[0, 15]$	1	1

5.2 Evaluation Data

Stream mining frameworks such as MOA [2] and River [23] focus on stream classification, as indicated by the streams they offer. We investigate regression which is why we require time series of the form: $gt(t) = ((t, x(t)), c(x(t), t))$. Here $t \in \mathbb{T}$ is the time, $x(t) \leftarrow X \mid t \in \mathbb{T}$ the observed input stream, and $c(x(t), t) \mid t \in \mathbb{T}$ the observed values of the target variable.

We generate $c(x(t), t))$ with a Gaussian process (GP) according to the following priors, where $C_{sin}(x(t), t))$ and $C_{rbf}(x(t), t))$ are adaptions from classification to regression used in [11] and [30], and similar to River RBF streams:

$$C_b(x(t), t)) = RBF(x \mid l_{gr}, v_{gr}) + RBF(x \mid l_{gr}, v_{gr}) * B(t \mid v_{gb}) \tag{1}$$

$$C_{sin}(x(t), t)) = RBF(x \mid l_{gr}, v_{gr}) + RBF(x \mid l_{gr}, v_{gr}) * Sin(t \mid s * v_{gb}) \tag{2}$$

$$C_{rbf}(x(t), t)) = RBF(x \mid l_{gr}, v_{gr}) + RBF(x \mid l_{gr}, v_{gr}) * RBF(t \mid s * v_{gb}) \tag{3}$$

$$\begin{aligned} C_{mix}(x(t), t)) = {} & RBF(x \mid l_{gr}, v_{gr}) + RBF(x \mid l_{gr}, v_{gr}) * RBF(t \mid s * v_{gb}) \\ & + RBF(x \mid l_{gr}, v_{gr}) * Sin(t \mid s * v_{gb}) \\ & + RBF(x \mid l_{gr}, v_{gr}) * B(t \mid v_{gb}) \end{aligned} \tag{4}$$

$$\begin{aligned} C_{bmix}(x(t), t)) = {} & RBF(x \mid l_{gr}, v_{gr}) + RBF(x \mid l_{gr}, v_{gr}) * B(t \mid v_{gb}) \\ & + RBF(x \mid l_{gr}, v_{gr}) * B(t \mid v_{gb}) \\ & + RBF(x \mid l_{gr}, v_{gr}) * B(t \mid v_{gb}) \end{aligned} \tag{5}$$

Here l_{gr}, v_{gr}, v_{gb} are kernel parameters, chosen as follows: $l_{gr} = 0.1$; $v_{gr} = 0.25$; $v_{gb} \in \{0, 0.005, 0.01, 0.02\}$. From these parameters v_{gb} controls the drift speed, $s = 2000$ scales v_{gb} so that each prior has a similar drift speed for a given v_{gb}. We visualize example time series drawn from such priors in Figure 2. For evaluation one time series is $t_{end} = 1000$ simulation units (su) long. The input stream $x(t)$ changes $n_c \in \{50, 100, 200, 400\}$ times within this 1000 su time frame, with the time of a change $t_c \leftarrow U[0, t_{end}]$ and $x(t_c) \leftarrow U[-1, 1]$. While tested on the $[-1, 1]$ range our method is compatible with any value range.

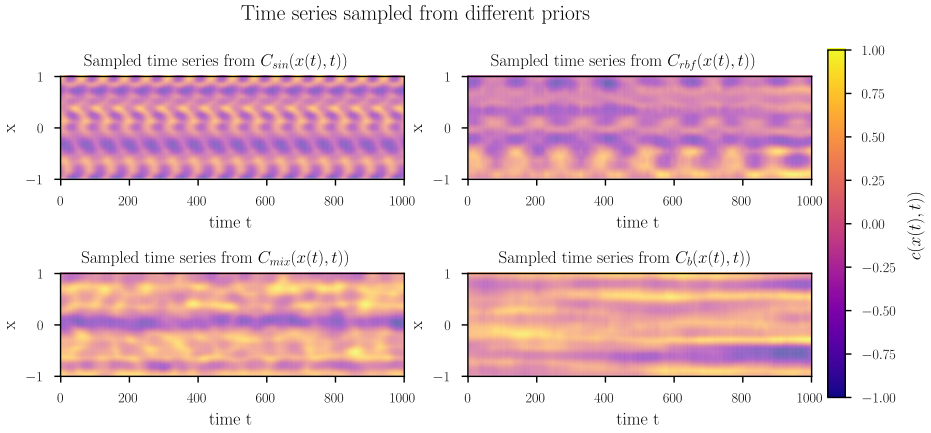


Fig. 2: Example time series sampled from the random function priors. Here, we show $c(x(t), t))$ normalized for better visualization.

5.3 Evaluation Metrics

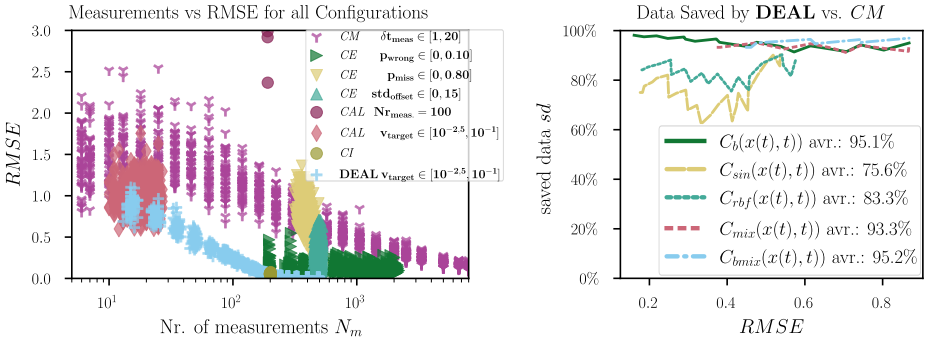
For DEAL and every baseline on every dataset, we perform the following: We evaluate all parameter configurations 50 times, each time on a different time series. For each time series, we compute the *RMSE* of the estimation y_{est} against the ground truth and the total number of measurements N_m performed across each time series. We plot the *RMSE* over the N_m for all different configurations and different time series. Additionally, we calculated the percentage of measurements a competitor (DEAL) saves compared to a baseline (the CM baseline):

Definition 2. *The saved data is* $sd = (Mb(e) - Mc(e))/Mb(e)$*, where* $Mb(e)$ *and* $Mc(e)$ *are the number of measurements a baseline and a competitor need to reach the same mean RMSE value* e *for the first time.*

6 Evaluation

6.1 Comparison of DEAL Against Baselines

Figure 3a shows the *RMSE* against the number of measurements N_m for DEAL and the four baselines (with two variants of *CAL* and three variants of *CE*) across a variety of parameter configurations. We can see that DEAL (+) needs on average fewer measurements to achieve lower *RMSE* than any configuration of any baseline (e.g., $N_m = 100$ for average $RMSE = 0.25$). Next, DEAL has less variance across the 50 time series than the baselines. This means for any fixed v_{target}, DEAL adapts better to the individual behavior of a time series, while the baselines depend on how well their parameters match the time series behavior. The ideal change detector *CI* (●) shows a similar error as DEAL at $N_m = 200$.



(a) (b)

Fig. 3: Baseline comparison; 3a the relationship between N_m and $RMSE$ for all parameters of the respective approach given in the legend and across different time series; 3b data gain against CM baseline.

But, CI's measurement count depends directly on the input stream's change frequency. Thus, CI will still carry out the same number of measurements (more than needed), even if a higher error would be acceptable. Further, as soon as the change detector is imperfect, as with CE (with offset ▲, missed ▼ and wrong ▶ detections) we observe a sharp increase in $RMSE$ and its variance. The CAL approach (♦) fails because once it reaches the given error threshold, it stops collecting data, never aware of any possible drift. Forcing CAL to collect 100 measurements regardless of the error threshold (●) causes it to learn an average value, rather than the correct relationship between the variables.

The CM baseline is the only baseline that allows users to indirectly control the $RMSE$ by choosing the measurement time interval δt_{meas}. DEAL can directly control the user-required error thresholds v_{target}. For any other baseline, control of the resulting estimation error via a parameter is not possible. Figure 3b shows for a given $RMSE$ how much data DEAL can save compared to CM. In regions of low error ($RMSE \in [0.1, 0.35]$), DEAL saves over 95% data compared to CM. Moving to regions of higher error ($RMSE > 0.5$), we observe a slow decline in saved data. We hypothesize that this is because the larger the allowed error, the less benefit a precisely selected measurement (time based on drift behavior) has, compared to manual selection δt_{meas} of CM. The drop in saved data ($RMSE \in [0.3, 0.45]$) for C_{sin} and C_{rbf} is due to their periodic nature, which is why CM performs acceptable even without dynamic adaption.

6.2 Impact of the User-Required Error Threshold

The true estimation error $RMSE$ should be close to the user-required standard deviation $std_{target} = \sqrt{v_{target}}$, so that the user can trust the model predictions. In Figure 4a, the dotted line shows such proportional behavior. For small thresholds $std_{target} < 0.25$, DEAL overestimates the actual error. For higher errors

$std_{target} > 0.25$, it underestimates the error. At all times, the actual error is close to the given threshold and behaves nearly proportionally as required.

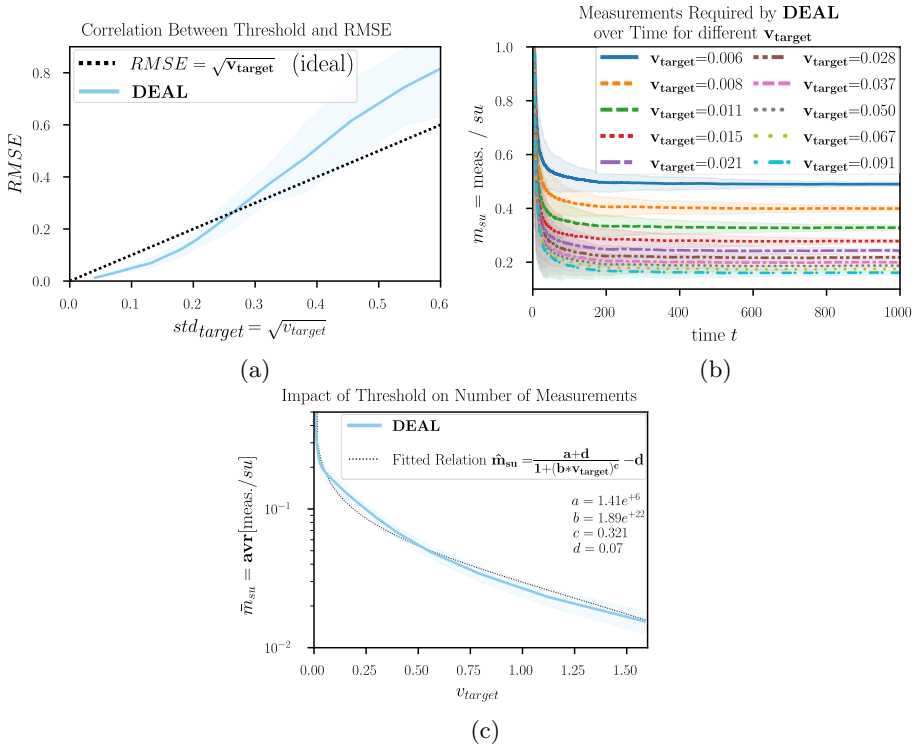


Fig. 4: Impact of the user-required error threshold; 4a average RMSE for a given threshold; 4b average amount of measurements per time step (for different thresholds); 4c average amount of required measurements for a given threshold.

Figure 4b shows the number of measurements DEAL performs per time step (*simulation unit su*), depending on the user-required error threshold. In the early stages (the first 100 *su*), DEAL performs more measurements because it needs to calibrate. Roughly after $t_{conv} = 200$ *su*, it takes a constant number of measurements per *su*, just enough to reach the error threshold. The variance of the number of measurements tends to decrease with time. Namely, the more data DEAL has seen, the closer its learned parameters are to the true ones.

Figure 4c shows how many measurements per *su* ($\bar{m}_{su}$) DEAL requires to reach a given error threshold v_{target}. As expected, if a user requires a lower estimation error, more measurements are needed to reach that error. To provide an estimation (dotted line) of this relation, we used a genetic function fitter. This relationship aids in estimating the required measurements and associated costs of reaching the user-required error threshold.

7 Conclusion

The relationship between input and target variables may drift due to environmental influences that are not observed. If the target variable is continuous, its prediction is a regression task. Current work on active learning tends to focus on classification, and the resulting methods do not easily translate to regression.

We proposed DEAL, a method that adapts the frequency of measurements to the drifting relationship, to reach a given user-required error threshold. DEAL models drift by predicting the target variable and estimating the variance of that prediction at arbitrary points in time. DEAL requires, on average, 20 times fewer measurements over the full range of user-required error thresholds than the CM baseline used in practice, in particular for frequently drifting streams.

Acknowledgments This work was supported by the German Research Foundation (DFG) as part of the Research Training Group GRK 2153: Energy Status Data – Informatics Methods for its Collection, Analysis, and Exploitation and by the Baden-Württemberg Foundation via the Elite Program for Postdoctoral Researchers.

References

1. Bachman, P., Sordoni, A., Trischler, A.: Learning algorithms for active learning. In: ICML (2017)
2. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. (2010)
3. C., K.F.: Introduction to stochastic calculus with applications. WSPC (2005)
4. Carne, G.D., Buticchi, G., Liserre, M., Vournas, C.: Load control using sensitivity identification by means of smart transformer. IEEE Trans. Smart Grid (2018)
5. Carne, G.D., Buticchi, G., Liserre, M., Vournas, C.: Real-time primary frequency regulation using load power control by smart transformers. IEEE Trans. Smart Grid (2019)
6. Dong, H., Jin, M., Ren-mu, H., Z.Y., D.: A real application of measurement-based load modeling in large-scale power grids and its validation. IEEE Trans. Power Systems (2009)
7. Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. (2014)
8. GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy (since 2012)
9. Iwashita, A.S., Papa, J.P.: An overview on concept drift learning. IEEE Access (2019)
10. Iwata, T.: Active learning for regression with aggregated outputs. CoRR (2022)
11. João, G., Pedro, M., Gladys, C., Pedro, R.: Learning with drift detection. In: artif. intell. adv. – SBIA (2004)
12. Konyushkova, K., Sznitman, R., Fua, P.: Learning active learning from data. In: NIPS (2017)
13. Krawczyk, B., Cano, A.: Adaptive ensemble active learning for drifting data stream mining. In: IJCAI (2019)

14. Krawczyk, B., Pfahringer, B., Wozniak, M.: Combining active learning with concept drift detection for data stream mining. In: IEEE BigData (2018)
15. Kurlej, B., Wozniak, M.: Learning curve in concept drift while using active learning paradigm. In: ICAIS (2011)
16. Kurlej, B., Wozniak, M.: Active learning approach to concept drift problem. Log. J. IGPL (2012)
17. Lindgren, G., Rootzen, H., Sandsten, M.: Stationary Stochastic Processes for Scientists and Engineers. T&F (2013)
18. Liu, S., Xue, S., Wu, J., Zhou, C., Yang, J., Li, Z., Cao, J.: Online active learning for drifting data streams. IEEE Trans. Neural Networks Learn. Syst. (2023)
19. Liu, W., Zhang, H., Ding, Z., Liu, Q., Zhu, C.: A comprehensive active learning method for multiclass imbalanced data streams with concept drift. Knowl. Based Syst. (2021)
20. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE Trans. Knowl. Data Eng. (2019)
21. Micchelli, C.A., Xu, Y., Zhang, H.: Universal kernels. J. Mach. Learn. Res. (2006)
22. Mohamad, S., Sayed Mouchaweh, M., Bouchachia, A.: Active learning for data streams under concept drift and concept evolution. In: ECML-PKDD (2016)
23. Montiel, J., Halford, M., Mastelini, S.M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H.M., Read, J., Abdessalem, T., et al.: River: machine learning for streaming data in python. JMLR (2021)
24. Park, C.H., Kang, Y.: An active learning method for data streams with concept drift. In: IEEE BigData (2016)
25. Riquelme, C., Johari, R., Zhang, B.: Online active linear regression via thresholding. In: AAAI (2017)
26. Settles, B.: Active Learning. Springer Cham (2012)
27. Shan, J., Zhang, H., Liu, W., Liu, Q.: Online active learning ensemble framework for drifted data streams. IEEE Trans. Neural Networks Learn. Syst. (2019)
28. Stefano, M., Bruno, S.: An active-learning algorithm that combines sparse polynomial chaos expansions and bootstrap for structural reliability analysis. Struct. Saf. (2018)
29. Turab, L., V., B.P., Dezhen, X., Ruihao, Y.: Active learning in materials science with emphasis on adaptive sampling using uncertainties for targeted design. Npj Comput. Mater. (2019)
30. Viktor, L., Barbara, H., Wersing, H.: Knn classifier with self adjusting memory for heterogeneous concept drift. In: ICDM (2016)
31. Wu, D., Lin, C., Huang, J.: Active learning for regression using greedy sampling. Inf. Sci. (2019)
32. Yoo, D., Kweon, I.S.: Learning loss for active learning. In: CVPR (2019)
33. Zhang, H., Liu, W., Shan, J., Liu, Q.: Online active learning paired ensemble for concept drift and class imbalance. IEEE Access (2018)
34. Zliobaite, I.: Learning under concept drift: an overview. CoRR (2010)
35. Zliobaite, I., Bifet, A., Holmes, G., Pfahringer, B.: MOA concept drift active learning strategies for streaming data. In: WAPA. JMLR Proceedings (2011)
36. Zliobaite, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with evolving streaming data. In: ECML/PKDD (3) (2011)
37. Zliobaite, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drifting streaming data. IEEE Trans. Neural Networks Learn. Syst. (2014)