Parameter Analysis for BOTL Drift Detectors

Helen McKay

1 Local Drift Detection

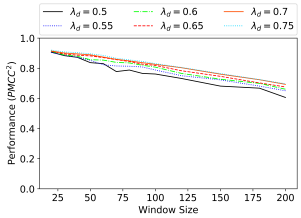
BOTL relies on a concept drift detection algorithm in each domain. Although any window-based drift detection strategy can be used, it is desirable for the chosen strategy to learn as few models as possible to represent each concept in the data stream. Limiting the number of models learnt in each domain reduces the number of input features to the OLS meta-learner, helping to prevent overfitting caused by the curse of dimensionality [3]. Additionally, reducing the number of models needing to be transferred across domains eredices communication and computational overhead of combining knowledge, which may impact the feasibility of using BOTL in real-world applications that require on-device learning.

A key characteristic of drift detection strategies that allow fewer models to be learnt includes the use of single model based approaches instead of ensemble based approaches. If an ensemble based drift detection strategy was used, such as Dynamic Weighted Majority (DWM) [5], or Adaptive Windowing Online Ensemble (AWOE) [6], the knowledge learnt to represent a single source concept may be encompassed across multiple models in the ensemble, therefore all models, and their ensemble weights, would need to be transferred. Additionally, the number of models learnt for a single source concept can be reduced by using a sliding window based approach instead of a fixed-size block approach. A fixed-size block approach may create multiple models over the duration of a single concept, particularly if the duration of each concept is not know prior to learning, whereas the use of a sliding window allows a single model to be continually used until a drift is encountered. Finally, the number of models learnt for a single source concept can be reduced by allowing previously learnt models to be reused in the presence of recurring concepts. To achieve this, a history of models can be retained to prevent redundant models being learnt and transferred.

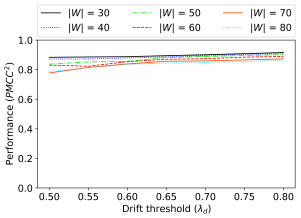
We consider three different concept drift detection techniques to underpin BOTL. Firstly, we adapt RePro [8] to a regression setting, secondly we apply ADWIN [2], and thirdly we propose our own local concept drift detection algorithm, Adaptive Windowing with Proactive drift detection (AWPro) which combines elements of RePro and ADWIN. We evaluate how each of the user defined parameters required impact the concept drift detection strategies on synthetic drifting hyperplane datasets, generated with uniform noise, containing sudden and gradual drifts, simulated smart home heating data, and real-world vehicle following distance data.

1.1 RePro

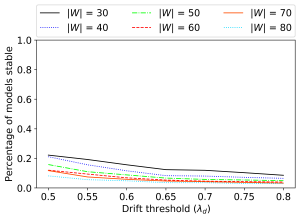
We consider an adaptation of RePro [8] for regression as the underlying drift detector. Although RePro requires some domain expertise to select appropriate parameter values, such as window size and drift threshold, it encapsulates key characteristics that allow few models to be learnt in each domain. RePro is a sliding window based detection algorithm that learns a single model for the current concept [7]. Additionally, RePro prioritises the reuse of existing models over learning new models by retaining a history of previously learnt models, H^T, and concept transitions, TM^T, to proactively determine which concept is likely to occur next [8].



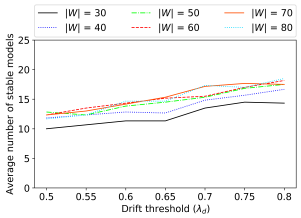
Performance of RePro with varying W_{max}



Performance of RePro with varying λ_d

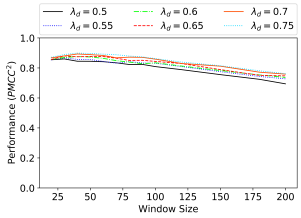


% models stable using RePro with varying λ_d

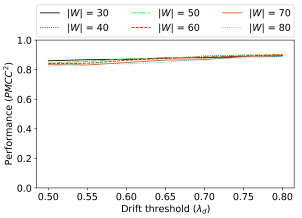


stable models using RePro with varying λ_d

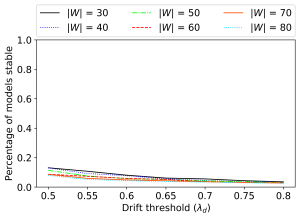
(a) Sudden drifting hyperplane data



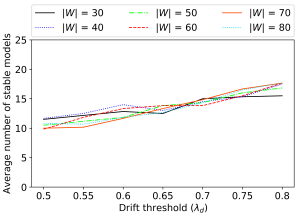
Performance of RePro with varying W_{max}



Performance of RePro with varying λ_d

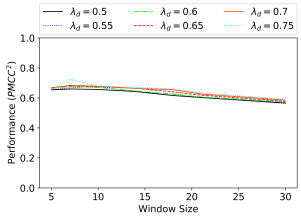


% models stable using RePro with varying λ_d

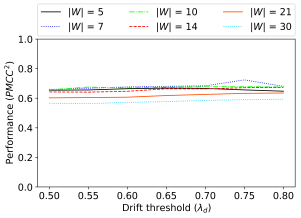


stable models using RePro with varying λ_d

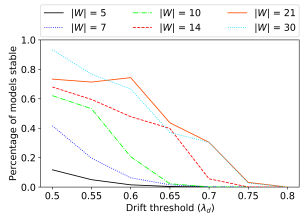
(b) Gradual drifting hyperplane data



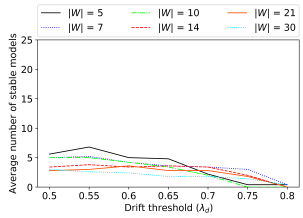
Performance of RePro with varying W_{max}



Performance of RePro with varying λ_d

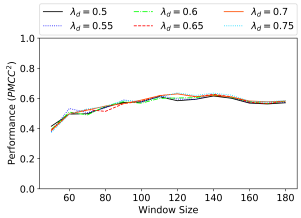


% models stable using RePro with varying λ_d

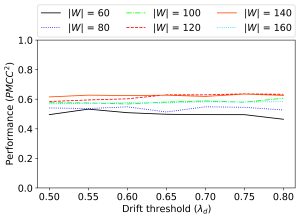


stable models using RePro with varying λ_d

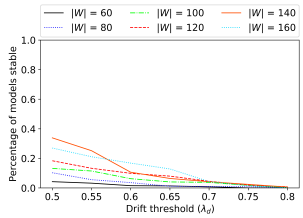
(c) Heating simulation data



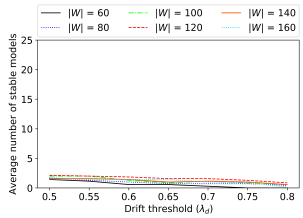
Performance of RePro with varying W_{max}



Performance of RePro with varying λ_d



% models stable using RePro with varying λ_d



stable models using RePro with varying λ_d

(d) Following distance data

Figure 1: Performance of RePro with varying window size, W_{max}, and drift threshold, λ_d, and the percentage of models learnt that are considered stable.

RePro was initially developed specifically for classification tasks, therefore modifications are required for regression settings, highlighted in Algorithm 1. The original RePro algorithm detects drifts by measuring the target model's classification accuracy across the sliding window, W. When the classification accuracy drops below an error threshold a drift is detected [8]. If the window is full, $|W| = W_{max}$, but the classifica-

Algorithm 1 Adapted RePro for regression.

Input: W_{max}, λ_l, λ_d, χ^T, H^T=$\emptyset$ (historical concepts), TM^T=$\emptyset$ (transition matrix).
Learn f_1^T using $x_1 ... x_{W_{max}}$, add to H^T
for t=W_{max}+1,W_{max}+2,... **do**
  Receive x_t and predict $\hat{y}_t^{'}$=$f_i^T(x_t)$
  Receive y_t, add $\langle x_t, \hat{y}_t^{'}, y_t \rangle$ to W
  if f_i^T is new and stable **then**
    Add f_i^T to H^T and $f_{i-1}^T \rightarrow f_i^T$ to TM^T
  if $R^2(f_i^T, W) < \lambda_d$ **then**
    f_{i+1}^T = SelectModel(H^T,TM^T,W) or learn new model over W
  else if $|W| \geq W_{max}$ **then**
    Remove $x_{(t-|W|)}$ from W
    while $|f_i^T(x_{(t-|W|)}) - y_{(t-|W|)}| \leq \lambda_l$ **do**
      Remove $x_{(t-|W|)}$ from W

tion accuracy does not drop below the error threshold, the sliding window is maintained by discarding one incorrectly classified instance, and all subsequent correctly classified instances. To apply RePro to regression, we maintain the sliding window by introducing ϵ-insensitivity through a loss threshold, λ_l, that allows instance $x_{(t-|W|)}$ to be discarded from the window if the predicted value of the regressor, $\hat{y}_{(t-|W|)}^{'}$, statisfies

$$|\hat{y}_{(t-|W|)}^{'} - y_{(t-|W|)}| \leq \lambda_l.$$

The R^2 performance of the target model, f_i^T, across W is used to detect drifts, where a drift is said to have occurred if the performance of the target model drops below a predefined drift threshold, λ_d, akin to observing the classification accuracy dropping below an error threshold.

Originally RePro used the notion of a stable learning size, specifying how much data is required to learn a stable model. This was necessary for the simulated classification tasks presented by Yang *et al.* as small window sizes allowed drifts to be detected quickly, however, this meant that insufficient instances were available in the window to learn a model that adequately represented the current concept [7]. Yang *et al.* suggest a stable learning size of $3W_{max}$. As real-world environments are often considerably more noisey than simulated or synthetic environments, using a small window size can cause drifts to be falsely detected, therefore a larger window size is necessary [2]. Increasing the window size also increases the stable learning size, however, if the stable learning size is increased, the data used to create a model may encapsulate multiple underlying concepts. To overcome this challenge, our adaptation of RePro for regression defines a stable model to be one that is learnt from W_{max} instances and is used to make predictions for $2W_{max}$ instances without a drift being detected.

To proactively determine future concepts, RePro maintains a transition matrix, TM^T, to determine the likelihood of encountering recurring concepts. To prevent the reuse of unstable models that make poor predictions, only those that are considered to be stable are added to the transition matrix. If the transition matrix indicates that it is equally likely that two or more concepts may be encountered next, RePro evaluates the performance of each model on the current window of data and selects the model with the highest accuracy. If the transition matix does not indicate a likely successor concept then each historical model is considered. A new model is only learnt when all historical models perform worse than the drift threshold λ_d.

1.1.1 Parameter Selection

The characteristics of RePro as a drift detection strategy are desirable for BOTL, however, RePro is dependent on user defined parameters that may not be intuatively selected. Parameter values including window

size, W_{max}, loss threshold, λ_l, and drift threshold, λ_d, must be carefully chosen for the specific online data stream. These parameters require domain expertise to consider appropraite values and trade-offs associated with each parameter. For example, selecting a large window size, W_{max}, allows more data to be retained to build local target models, increasing their accuarcy and stability [1], however, a small window size can allow RePro to react more quickly to concept drifts as a small window is more representative of the current distribution of the data stream. Similarly, large values chosen for the drift and loss thresholds, λ_d and λ_l, may prevent concept drifts from being detected, however, smaller values may cause drifts to be falsly detected in the presence of noise. Figure 1 highlights how the window size, $|W|$, and drift threshold λ_d, impact the performance of RePro when evaluated on different datasets.

The first graphs displayed in Figures 1a, 1b and 1c show the performance of RePro on sythetic datasets with varying window size, W_{max}, whereas the first graph in Figure 1d shows the performance of RePro on real-world data. The results presented in these graphs indicate that a smaller window size is preferable for the synthetic datasets, however a larger window size produces better performance on real-world data. This occurs due to the synthetic datasets containing less noise and simpler concepts in comparison to the following distance dataset. Due to these factors a smaller window can be used for the synthetic datasets as less instances are required to build a model that adequately represents the current concept, and drifts can be detected quickly. However, the following distance dataset benefits from larger window sizes due to the complex nature of the concepts to be learnt, requiring more instances to build models that effectively represent each concept in the data stream.

The second, third, and fouth graphs in Figures 1a, 1b, 1c and 1d show the performance of RePro, perentage of models that are considered stable, and the average number of stable models created by RePro repectively as the drift threshold, λ_d, is varied. The performance of RePro increases with λ_d, as it becomes more sensitive to drifts. The impact of λ_d on performance is not significant in comparison to window size, however, the percentage of models that are considered stable and the average number of stable models learnt are impacted by the drift threshold. The percentage of models learnt that are considered stable decreases as λ_d increases across all datasets. Increasing RePro's sensitivity to concept drifts can cause drifts to be detected while the window is still populated with instances from a previous concept. This means that a model is learnt using data belonging to both the new concept, and the previous concept, and therefore cannot make effective predictions when data belonging to the new concept is encountered. This can cause multiple concept drifts to be detected in short succession. As the drift threshold, λ_d, is also used to determine if RePro can reuse historical models, using high λ_d values increases the average number of stable models learnt in the drifting hyperplane datasets as extisting models cannot be reused, however, the average number of stable models learnt in the heating simulation and following distance datasets decreases. This occurs due to the heating simulation and following distance datasets containing more noise than the drifting hyperplane datasets, causing concept drifts to be encountered frequently and preventing models from being considered stable.

From the analysis presented in Figure 1 we find that RePro benefits from shorter window sizes when data streams contain small amounts of noise and simple concepts, however, in real-world environments larger window sizes are needed to ensure enough instances are retained to build a model that effectively represents the current concept. Selecting a larger drift threshold ensures drifts are detected quickly, but does not greatly impact predictive performances given an appropriate window size has been selected. However, using a large drift threshold can increase the number of unstable models learnt during periods of concept drift if the sliding window contains data from both the old and new concepts, which increases computation as multiple drifts are detected in quick succession.

1.2 ADaptive WINdowing (ADWIN)

Although the performance of RePro is not greatly impacted by the parameter values chosen, the number of unstable models learnt is high. This may limit it's use as the underlying concept drift detection algorithm if BOTL is to be used for applications with limited computational power. Although RePro uses a sliding window to detect concept drifts, it is unable to determine where in the window a drift has occurred, therefore, unstable models are often created in quick succession during, or immediately after, a drift has occurred while the sliding window still contains instances belonging to the previous concept. To overcome this challenge the concept drift detection algorithm must be able to determine the point within the sliding window one concept drifts to another. The ADaptive WINdowing (ADWIN) concept drift detection algorithm, presented by Bifet *et al.* [2], achieves this by monitoring the changes in the distribution of a data stream. Instead of using a fixed length sliding window, the size of the window is determined according to the rate of change observed in the online data stream [2].

ADWIN operates on the pricipal that if two large enough sub-windows have distinct enough means, the expected values within each sub-window will differ [2, 4]. A drift is said to be detected when

$$|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_{\text{cut}}, \tag{1}$$

where $\hat{\mu}_{W_0}$ and $\hat{\mu}_{W_1}$ are the means of sub-windows W_0 and W_1, and ϵ_{cut} is defined by the Hoeffding bound

$$\epsilon_{\text{cut}} = \sqrt{\frac{1}{2m} \cdot \ln \frac{4|W|}{\delta}},$$

where m is the harmonic mean of the sub-windows, $m = \frac{2}{1/|W_0| + 1/|W_1|}$, and δ is a confidence value, defined by the user, which determines the sensitivity of drift detection [2, 4].

BOTL uses ADWIN, as presented in Algorithm 2, to detect drifts within the data stream by monitoring the distribution of the predictive error of the locally learnt model, f_i^T,

$$|f_i^T(x_t) - y_t|.$$

Once a drift is detected, a new model is learnt locally, f_{i+1}^T, using the second sub-window such that $W = W_1$. Monitoring the distribution of predictive error allows drifts to be detected rapidly, however, drifts are fequently detected when only a small number of instances from the new concept have been observed, therefore window W is unlikely to be repesentative of the underlying concept. To address this, a new model, f_{i+1}^T, is only created once $|W| = W_{max}$, until then, the old model f_i^T is used to make predictions.

1.2.1 Parameter Selection

Using ADWIN in this way requires two user defined parameters, the maximum window size,W_{max}, and the confidence value, δ. Similarly to RePro, some domain expertise is required to select these parameter values. Figure 2 shows how the performance of ADWIN is effected by different window sizes and confidence values when evaluated on different datasets.

The first graphs displayed in Figures 2a, 2b, 2c and 2d show the performance of ADWIN on these datasets with varying window size. Similarly to RePro, ADWIN suffers from a drop in performance as the window size increases in the synthetic datasets (Figures 2a, 2b and 2c), however, the drop in performance is more significant than observed by RePro, displayed in Figure 1. Although ADWIN is able to detect the point of concept change within the window of data, the model built to represent the previous concept must be used to make predictions until W_{max} instances have been observed after the drift point before a model can be learnt. This delay causes a more significant drop in performance as RePro is able to build unstable models from windows containing instances from both concepts during this period, which typically perform better

Algorithm 2 ADWIN.

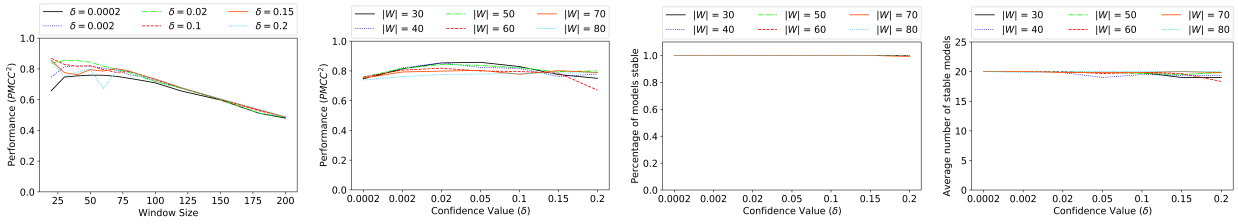
Input: W_{max}, χ^T, δ (confidence value), changeAlarm = False.
Learn f_1^T using $x_1 ... x_{W_{max}}$
Initialise ADWIN(δ)
for each t=W_{max}+1,W_{max}+2,... **do**
  Receive x_t and predict $\hat{y}_t^{'}$=$f_i^T(x_t)$
  Receive y_t, and calculate diff=($\hat{y}_t^{'}$−y_t)
  Add $\langle x_t, \hat{y}_t^{'}, y_t, \text{diff} \rangle$ to W
  if not changeAlarm **then**
    changeAlarm, W= ADWIN.update(diff)
  if changeAlarm and $|W| \geq W_{max}$ **then**
    Learn f_{i+1}^T using W
    changeAlarm = False
    Re-initialise ADWIN(δ)

than continuing to use the model of the previous concept. ADWIN also suffers from a drop in performance when the window size is very small. This is due to ADWIN detecting drifts by monitoring the distribution of the predictive error. When the window size is small, there are not enough instances within the window to build a model that adequately represents the current concept. This increases the predictive error observed throughout the datastream. However, as ADWIN only monitors the distribution of predictive error, rather than the performance of each model, if a model consistently performs poorly, the model will continue to be used until a drift is encountered, whereas RePro will build a new model if the performance of the model is below the drift threshold.

ADWIN also suffers from a loss in performance when used with a high confidence value and a window size between 50 and 150 instances on the drifting hyperplane datasets containing gradual drifts. This is caused by the synthetic nature of the gradual drifting hyperplane data. In these datasets a gradual drift occurs over a period of 100 instances. During this period the probability of an instance belonging to the old concept is initially high, and gradually decreases until the drifting period is over. Therefore, with high confidence values, a concept drift can be detected early within this drifting phase. This means that a model is learnt using instances belonging to both the old concept, and the new concept. Predictions made using this model will suffer from a high variance in predictive error, therefore, ADWIN is not easily able to detect the drift.

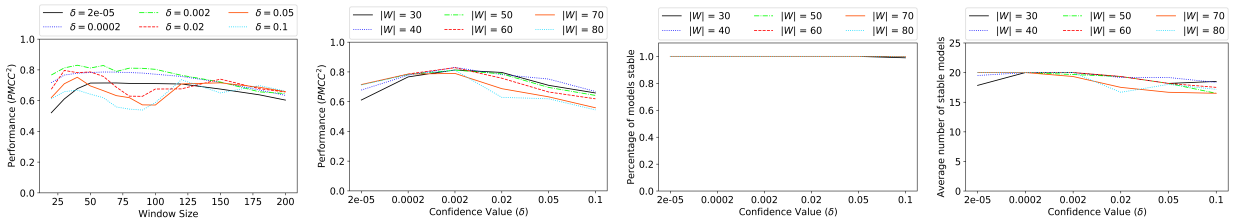
Similarly to RePro, ADWIN experiences an increase in performance as the window size increases for the vehicle following distance dataset due to containing more complex concepts that require more data to learn a model that adequately represents the current concept.

The second, third and fourth graphs displayed in Figures 2a, 2b, 2c and 2d show the performance of ADWIN, percentage of models that are considered stable, and the number of stable models created by ADWIN respectively as the confidence value, δ, varies. The confidence value selected determines how sensitive ADWIN is to changes in the distribution of the predictive error. Although varying the confidence value for both sudden and gradual drifting hyperplane dataset does not impact the performance of ADWIN greatly, a slightly higher confidence value of 0.05 is preferred for the sudden drifting data streams, observed in Figure 2a, in comparison to the perferred value of 0.002 for the gradual drifting data streams, shown in Figure 2b. This can be attributed to the transition period between two concepts within the gradual drifting data stream. During periods of gradual drift, the probability of instances belonging to the initial concept is high, however, as the data stream progresses, the likelihood of an instance belonging to the new concept increases. By decreasing the confidence threshold, ADWIN is able to tolerate more changes to the distribution of prediction error before detecting a drift, therefore the window of data used to build a model for the new



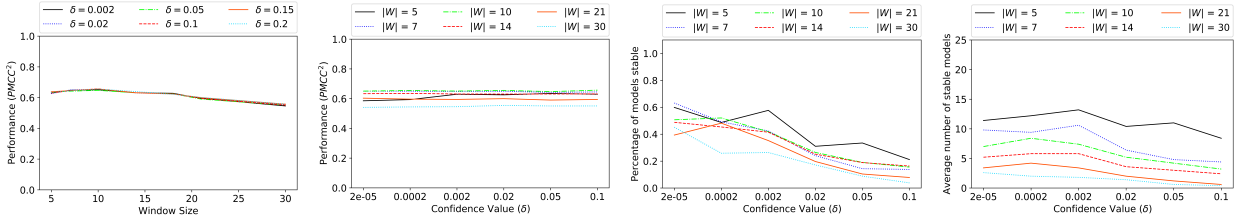
Performance of ADWIN with varying W_{max} | Performance of ADWIN with varying δ | % models stable using ADWIN with varying δ | # stable models using ADWIN with varying δ

(a) Sudden drifting hyperplane data



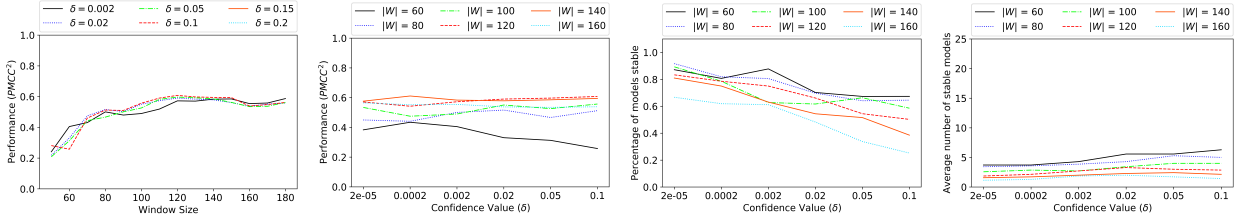
Performance of ADWIN with varying W_{max} | Performance of ADWIN with varying δ | % models stable using ADWIN with varying δ | # stable models using ADWIN with varying δ

(b) Gradual drifting hyperplane data



Performance of ADWIN with varying W_{max} | Performance of ADWIN with varying δ | % models stable using ADWIN with varying δ | # stable models using ADWIN with varying δ

(c) Heating simulation data



Performance of ADWIN with varying W_{max} | Performance of ADWIN with varying δ | % models stable using ADWIN with varying δ | # stable models using ADWIN with varying δ

(d) Following distance data

Figure 2: Performance of ADWIN with varying window size, W_{max}, and confidence value, δ, and the percentage of models learnt that are considered stable.

concept contains fewer instances that belong to the previous concept. However, if the confidence value is too small a drop in performance is observed as drift detection is delayed.

Although the confidence value impacts the performance of the drifting hyperplane datasets, it does not impact the heating simulation or real-world following distance datasets as greatly, assuming a large enough

window of instances is used to create predictive models (see Figures 2c and 2d). If the window size is too small, the performance of ADWIN decreases as the confidence value increases for the following distance datasets. This occurs because increasing the confidence value increases ADWIN's sensitivtity to changes in the distribution of predictive error. When the window size is too small, the model built is not representative of the current concept, therefore the variance in predictive error becomes high, increasing the frequency in drift detection events. Our implementation of ADWIN requires W_{max} instances to be available after a drift has been detected in order to build a model, therefore, when the window size is small, not only does the model have poor predictive capabilities, as the confidence value increases, the frequency of waiting for sufficient data to be collected to build a new model increases. During these periods the previously learnt model must be used to make predictions.

As ADWIN detects the point in the window that a concept drift occurs, the percentage of models learnt by ADWIN that are considered stable is considerably higher than that of RePro across all data streams. This reduces computation as unstable models are not repeatedly learnt immediately after a drift, or during a gradual drift. However, the number of stable models learnt using this approach is higher compared to RePro because ADWIN does not retain a history of models that can be reused in the presence of recurring concepts. This problem can be seen when considering the results obtained from the sudden drifting hyperplane datasets in Figures 1a and 2a. The sudden drifting hyperplane datasets used for this analysis contain 5 distinct concepts, each repeated 4 times. ADWIN successfully creates 20 stable models, one for each concept period, regardless of the confidence value chosen. However, RePro creates between 10 and 18 stable models, depending on the drift threshold, λ_d, which is used to determine when existing models can be reused.

From the analysis presented in Figure 2 we find that ADWIN creates fewer unstable models in comparison to RePro, leading to reduced computation. This may make ADWIN more applicable to use as the underlying drift detection method for BOTL when used for on-device learning with computational limitations. However, ADWIN does not retain a history of models, therefore models must be unnecessarily re-learnt when encountering a recurring concept. This may impact the performance of BOTL as the likelihood of the OLS meta-learner overfitting increases as the number of transferred models becomes large in comparison to the number of instances available in the window [3]. Additionally ADWIN exhibits a poorer performance when compared to RePro as ADWIN must wait for sufficient instances of the new concept to be observed when the number of instances within the second sub-window is small, whereas RePro typically creates unstable models using instances belonging to both the old and new concepts during this period, whic frequently performs better than using the model learnt to represent the previous concept.

1.3 Adaptive Windowing with Proactive Predictions (AWPro)

The use of RePro as a concept drift detection algorithm can be computationally demanding due to the creation of high volumes of unstable models, caused by it's inability to detect the precise point of drift within the sliding window. ADWIN allows this point to be indentified by splitting the sliding window into two sub-windows where the first sub-window contains instances belonging to the old concept, which can be removed, while the second sub-window contains instances belonging to the new concept. However, if the remaining instances inthe second sub-window is small, ADWIN must wait until sufficient instances have need observed before a new model can be learnt, negatively impacting the performance of ADWIN. Additionally, ADWIN does not reuse previously learnt models, therefore models must be re-learnt for recurring concepts. This increases the number of models transferred between domains and prevents previously learnt models from being used to make predictions when few instances from a recurring concept have been observed.

To reduce the computation exhibited by a local drift detection algorithm, while preventing duplicate models being learnt for recurring concepts, we introduce an alternative concept drift detection strategy, Adaptive Windowing with Proactive predictions (AWPro), presented in Algorithm **??**, which combines desirable characteristics from ADWIN and RePro that better suit the BOTL framework.

Algorithm 3 AWPro.

Input: W_{max}, χ^T, δ (confidence value), changeAlarm = False, λ_r, H^T=$\emptyset$ (historical concepts), TM^T=$\emptyset$ (transition matrix).
changeAlarm, tempModel = False
Learn f_1^T using $x_1 ... x_{W_{max}}$, add to H^T
Initialise ADWIN(δ)
for each t=W_{max}+1,W_{max}+2,... **do**
  Receive x_t and predict $\hat{y}_t^{'}$=$f_i^T(x_t)$
  Receive y_t, and calculate diff=($\hat{y}_t^{'}$−y_t)
  Add $\langle x_t, \hat{y}_t^{'}, y_t, \text{diff} \rangle$ to W
  if not changeAlarm **then**
    changeAlarm, W= ADWIN.update(diff)
  if changeAlarm and $|W| < \frac{1}{2} W_{max}$ and **not** tempModel **then**
    tempModel = True
    Learn f_{temp}^T using W
  else if changeAlarm and $|W| \geq \frac{1}{2} W_{max}$ **then**
    if R^2(SelectModel(H^T,TM^T,W),W)$\geq \lambda_r$ **then**
      f_{i+1}^T = SelectModel(H^T,TM^T,W)
      changeAlarm,tempModel = False
      Re-initialise ADWIN(δ)
  else if changeAlarm and $|W| \geq W_{max}$ **then**
    f_{i+1}^T = SelectModel(H^T,TM^T,W) or learn new model over W
    changeAlarm,tempModel = False
    Re-initialise ADWIN(δ)
  if f_i^T is new and stable **then**
    Add f_i^T to H^T and $f_{i-1}^T \rightarrow f_i^T$ to TM^T

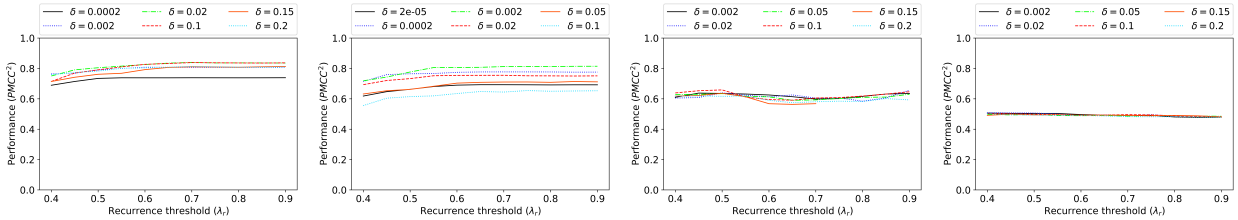
AWPro uses ADWIN to monitor the change in distribution of predictive error, allowing the drift detection strategy to partition instances belonging to different concepts within the sliding window. Concept drifts are identified using Equation 1, which is dependent on a confidence value, δ, that determines ADWIN's sensitivity to changes in the distribution of the predictive error. Once a model is learnt, RePro is used to identify stable models, which are retained in the model history, H^T, and the transition between concepts is added to the transition matrix, TM^T. A stable model is one that is used to make predictions over $2W_{max}$ instances without a drift being detected.

Each time a concept drift is encountered, AWPro drops the first sub-window of instances such that all instances in the window belong to the new concept. If the remaining data in the window is less than $\frac{1}{2} W_{max}$ instances then a temporary model is created. Although these temporay models are akin to unstable models learnt using RePro, the data used to build these models only contains instances belonging to the new concept. Having few instances available increases the likelihood of learning a model that is not representative of the entire concept, however, this may be prefereable to ADWIN's approach of continuing the use of a model that no longer represents the current concept, or RePro's approach where a model may be learnt from data belonging to both concepts. Once a temporary model has been learnt, incoming instances are added to the window.

When $\frac{1}{2} W_{max}$ instances have been observed after a drift the proactive nature of RePro is used by AWPro to determine if an existing model can be used to represent the new concept. This allows AWPro to identify an existing model that has already been learnt to be used for predictions prior to a full window of instances

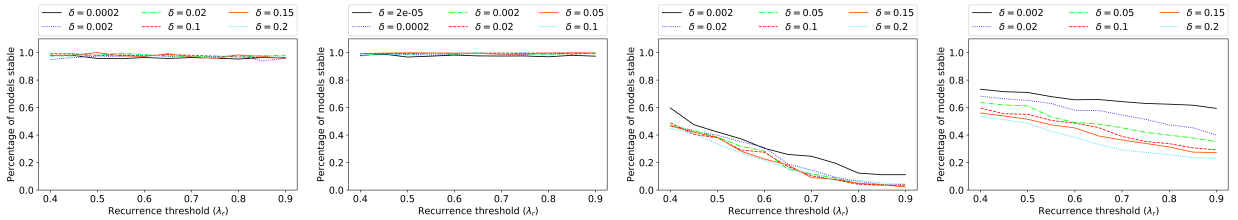
being observed. AWPro uses a recurrence threshold that determines if an existing model can be reused, λ_r, which acts in that same way as the drift threshold, λ_d, used by RePro when considering the reuse of models. If a model's R^2 performance is greater than the recurrence threshold, λ_r, it can be reused, and the process of detecting concept drifts through monitoring changes to the distribution of predictive error is resumed. However, if no model exists, the temporary model is continually used until W_{max} instances of the new concept have been observed.

Finally, when W_{max} instances have been observed after a concept drift, AWPro uses the transition matrix and historical models to identify if any existing model can be used now a sample of data that is more representative of the current concept has been obtained. If no model exceeds the recurrence threshold, λ_r a new model is learnt using the data available within the sliding window.



Synthetic sudden drifting hyperplane data | Synthetic gradual drifting hyperplane data | Synthetic heating simulation data | Real-world following distance data

(a) Performance of AWPro for different confidence values, δ, as the recurrence threshold, λ_r, increases.



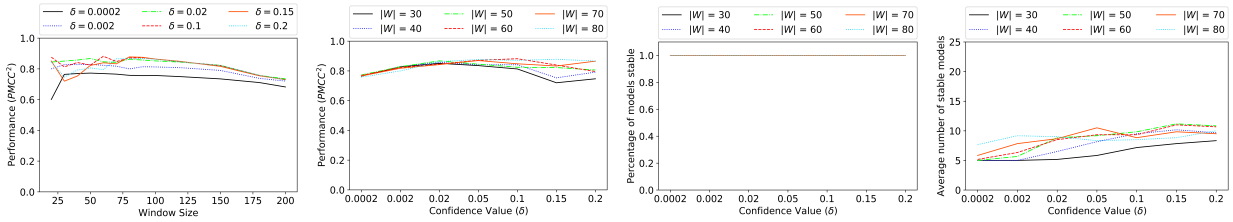
Synthetic sudden drifting hyperplane data | Synthetic gradual drifting hyperplane data | Synthetic heating simulation data | Real-world following distance data

(b) % of models leant by AWPro that are stable for different confidence values, δ, as the recurrence threshold, λ_r, increases.

Figure 3: Performance of AWPro and the percentage of learnt models that are considered to be stable with varying confidence values, δ, as the recurrence threshold, λ_r, increases.

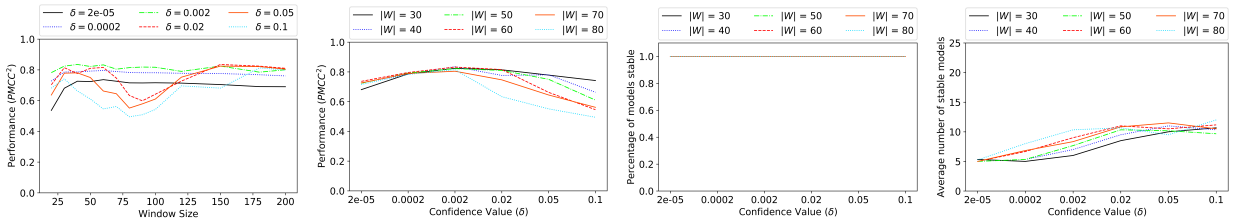
1.3.1 Parameter Selection

AWPro relies on three user defined parameters: the confidence value, δ, the window size, W_{max}, and the recurrence threshold, λ_r. The confidence value, δ, defined by ADWIN [2], determines how sensitive AWPro is to concept drifts, where higher values of δ increase AWPro's sensitivity to changes in the distribution of predictive error. The window size, W_{max} is not used for drift detection as AWPro uses an adaptive sliding window, however, W_{max} is used to determine how many instances are required before a model can be built. Once a model is learnt, W_{max} is also used to determine whether a model is stable, therefore indicating which models should be added to the history of concepts, H^T, and which concept transitions should be stored in the transition matrix, TM^T. Finally, the recurrence threshold parameter, λ_r, is used to determine if historical models can be reused. If the R^2 performance of a historical model on the current window of data is above the recurrence threshold we reuse the model instead of learning a new model.



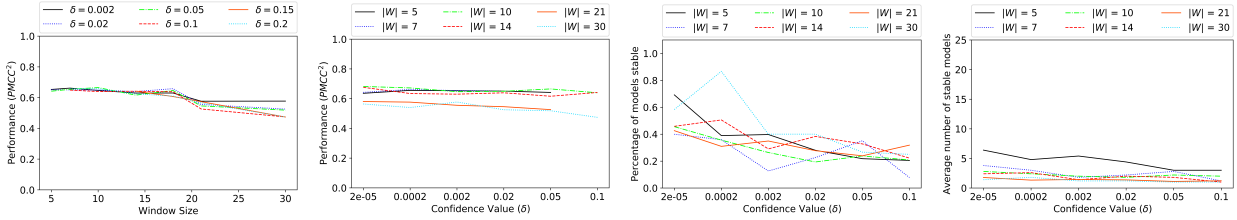
Performance of AWPro with varying W_{max} | Performance of AWPro with varying δ | % models stable using AWPro with varying δ | # stable models using AWPro with varying δ

(b) Sudden drifting hyperplane data with a fixed recurrence threshold, $\lambda_r = 0.6$



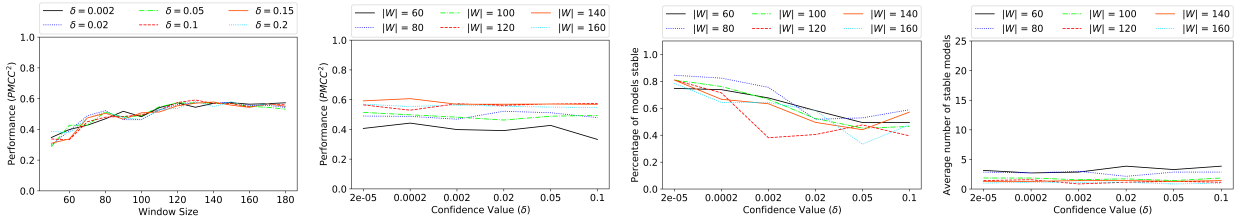
Performance of AWPro with varying W_{max} | Performance of AWPro with varying δ | % models stable using AWPro with varying δ | # stable models using AWPro with varying δ

(b) Gradual drifting hyperplane data with a fixed recurrence threshold, $\lambda_r = 0.6$



Performance of AWPro with varying W_{max} | Performance of AWPro with varying δ | % models stable using AWPro with varying δ | # stable models using AWPro with varying δ

(c) Heating simulation data with a fixed recurrence threshold, $\lambda_r = 0.6$



Performance of AWPro with varying W_{max} | Performance of AWPro with varying δ | % models stable using AWPro with varying δ | # stable models using AWPro with varying δ

(d) Following distance data with a fixed recurrence threshold, $\lambda_r = 0.7$

Figure 4: Performance of AWPro with varying window size, W_{max}, and confidence value, δ, and the percentage of models learnt that are considered stable.

Figure 3 shows how the performance of AWPro (see Figure 3a), and the percentage of stable models (see Figure 3b), are impacted by increasing the recurrence threshold, λ_r, for varying confidence values, δ. As the recurrence threshold is not used for concept drift detection it does not significantly effect the performance of AWPro. However, it is used to determine if a historical model can be reused, therefore λ_r does impact

the percentage of models that are considered stable for the heating simulation and real-world following distance datasets. As the λ_r increases the percentage of models that are considered stable decreases. When λ_r is high, more unstable models are created as existing models do not achieve an R^2 performance above λ_r, therefore temporary models must be created while W_{max} instances are observed after a concept drift is detected in order to build a model that adequately represents the new concept. A reduction in the percentage of models that are considered stable is not observed for the drifting hyperplane datasets as the concepts that are repeated within the data stream are exact duplicates of one another and the datasets contain less noise, therefore, the models initially learnt retain a high performance when a concept is repeated, preventing the need to learn temporary models.

Figure 4 highlights how the window size, W_{max}, and confidence value, δ, impact the performance, percentage of models created that are stable, and the number of stable models created by AWPro. The first graph in Figures 4b, 4b, 4c and 4d show how the performance of AWPro is impacted by varing W_{max} for sythetic and real-world datasets. AWPro suffers from a decrease in performance as the window size increases, and when the window size is very small, for the sythetic datasets (Figures 4b, 4b and 4c). This reduction in performance is caused by using ADWIN as the underlying drift detection strategy and similar trends can be seen in Figure 2.

A more significant drop in performance is observed when using AWPro compared to ADWIN on the gradual drifting hyperplane datasets using a high confidence value, and window sizes between 50 and 150 instances. This is caused by AWPro using ADWIN to detect point of concept drift, and the synthetic nature of the grdual drifting hyperplane data streams. When the confidence value is high, drifts are detected early within the phase of gradual drift. This means that the window of data used to determine if historical a model can be selected contains a high proportion of instances belonging to the previous concept, and therefore the model associated with the previous concept is likely to be re-selected. As the window contains some instances belonging to the new concept, the variance in predictive error is initally high, therefore AWPro cannot easily detect a drift as more instances belonging to the new concept are observed as drifts are detected by monitoring the change in distribution of predictive error. This phenomenon is only observed within the gradual drifting hyperplane datasets because the drifting period occurs at fixed intervals, for a fixed period of time, and the probability of an instance belonging to the new concept is directly proportional to the number of instances observed during the drifting period. However, this is not observed in the smart home heating simulation data, or the vehicle following distance data as the gradual drifts occur naturally at varying rates, intervals and lengths.

The performance of AWPro on the vehicle following distance data follows a similar trend to RePro and ADWIN as the window size is increased and more data is made available to build a model that adequately represents the current concept. However, AWPro achieves a higher performance than ADWIN for small window sizes as AWPro is able to reuse existing model, or create temporary models after $\frac{1}{2}W_{max}$ instances have been observed, whereas ADWIN must wait for W_{max} instances to be observered prior to building a new model.

The second graphs in Figures 4b, 4b, 4c and 4d highlight the performance of AWPro as the confidence value is varied. The performances achieved by AWPro follow the same trends as those observed when varying the confidence value of ADWIN. However, for the vehicle following distance data, the performance of ADWIN decreases when the confidence value is high for small window sizes, whereas this is not observed to the same extent when using AWPro. This is again due to AWPro's ability to create temporary models, and reuse existing models when only $\frac{1}{2}W_{max}$ instances have been observed, rather than having to wait for W_{max} instances to be observed when using ADWIN.

The third and fouth graphs in Figures 4b, 4b, 4c and 4d display the percentage of models learnt that are considered stable, and the number of stable models learnt by AWPro respectively. Compared to ADWIN, AWPro retains a high percentable of models that are considered stable for the drifting hyperplane datasets, however, the number of stable models is significantly reduced due to AWPro's ability to reuse existing

models for recurring concepts. In comparison to RePro, the percentage of models learnt that are considered stable is considerably higher for AWPro. This indicates that AWPro is benefitting from being able to detect the exact point in the sliding window where a concept drift is encountered as instances belonging to the previous concept are removed from the window, preventing unstable models from being learnt. Additionally, the number of stable models learnt by AWPro is less than that of RePro. These characteristics are also encountered when comparing the percentage of models learnt that are considered stable, and the number of stable models learnt for the smart home heating simulation, and the vehicle following distance datasets when comparing AWPro, RePro and ADWIN.

From the analysis presented in Figures 1, 2, 3 and 4 we find that AWPro achieves slightly better performance than ADWIN while creating fewer stable models due to it's ability to reuse historical models, reducing the communication overhead when used as the underlying local concept drift detection algorithm for BOTL. Although RePro achieves predictive performances greater than AWPro, the number of unstable models created by AWPro is significantly reduced, which may be a desirable trade-off when using BOTL for applications that require on-device learning that have limited computational abilities.

References

[1] Albert Bifet. Adaptive learning and mining for data streams and frequent patterns. *SIGKDD Explor. Newsl.*, 11(1):55–56, November 2009.

[2] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 443–448. SIAM, 2007.

[3] Jerome H. Friedman. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, Mar 1997.

[4] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[5] Jeremy Z. Kolter and Marcus A. Maloof. Dynamic weighted majority: a new ensemble method for tracking concept drift. In *Third IEEE International Conference on Data Mining*, pages 123–130, 2003.

[6] Yange Sun, Zhihai Wang, Haiyang Liu, Chao Du, and Jidong Yuan. Online ensemble using adaptive windowing for data streams with concept drift. *International Journal of Distributed Sensor Networks*, 12(5):4218973, 2016.

[7] Ying Yang, Xindong Wu, and Xingquan Zhu. Combining proactive and reactive predictions for data streams. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 710–715. ACM, 2005.

[8] Ying Yang, Xindong Wu, and Xingquan Zhu. Mining in anticipation for concept change: Proactive-reactive prediction in data streams. *Data Mining and Knowledge Discovery*, 13(3):261–289, 2006.