Learning under Concept Drift with Follow the Regularized Leader and Adaptive Decaying Proximal

Ngoc Anh Huynh[a,*], Wee Keong Ng[a], Kanishka Ariyapala[b]

[a] *Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore*
[b] *University of Florence, Via Morgagni, 65, 50134, Firenze - Italy*

Abstract

Concept drift is the problem that the statistical properties of the data generating process change over time. Recently, the Time Decaying Adaptive Prediction (TDAP) algorithm[1] was proposed to address the problem of concept drift. TDAP was designed to account for the effect of drifting concepts by discounting the contribution of previous learning examples using an exponentially decaying factor. The drawback of TDAP is that the rate of its decaying factor is required to be manually tuned. To address this drawback, we propose a new adaptive online algorithm, called Follow-the-Regularized-Leader with Adaptive Decaying Proximal (FTRL-ADP). There are two novelties in our approach. First, we derive a rule to automatically update the decaying rate, based on a rigorous theoretical analysis. Second, we use a concept drift detector to identify major drifts and reset the update rule accordingly. Comparative experiments with 14 datasets and 6 other online algorithms show that FTRL-ADP is most advantageous in noisy environments with real drifts.

Keywords: Concept Drift, Decaying Rate, Drift Detector, Online Learning, Follow the Regularized Leader

*Corresponding author
Email addresses: hu0001nh@e.ntu.edu.sg (Ngoc Anh Huynh), wkn@pmail.ntu.edu.sg (Wee Keong Ng), kanishka.ariyapala@math.uni.it (Kanishka Ariyapala)

[1] Scalable Time-Decaying Adaptive Prediction Algorithm. (Tan et al., 2016)

1. Introduction

The problem of concept drift arises in many practical domains such as malware detection (Saxe & Berlin, 2015), fraud detection (Pozzolo et al., 2015), and mobile robots (Thrun et al., 2006). In these applications, the performance of the train-and-forget batch model is known to degrade over time. For example, in the field of malware detection, Saxe & Berlin (2015) conducted two slightly different experiments with batch learning in trying to distinguish between malicious and benign executables. In the 1^{st} experiment, the author discards the temporal order of the executables and randomly splits the dataset into the training set and the test set. The best performance in this case is 95.2% accuracy. In the 2^{nd} experiment, the author respects the temporal order and uses a timestamp to split the dataset into the training set and the test set. The best accuracy in this case is 67.7%. This performance degradation of the trained batch models indicates that the behavior of malware does evolve over time. Similar results were also reported in another study (Bekerman et al., 2015).

The degradation of batch models in nonstationary settings originates from the fact that the formulation of batch learning does not factor in the issue of concept drift. The underlying assumption of batch learning is that the distribution of the data generating process does not change over time (the stationary assumption (Shalev-Shwartz & Ben-David, 2014)). This assumption may be true in established application domains such as face recognition and digit recognition. In highly dynamic applications such as intrusion detection and fraud detection, this assumption may be challenged. On the contrary, online learning, specifically the framework of Follow the Regularized Leader (FTRL in (Shalev-Shwartz, 2011)), is formulated without basing on the stationary assumption. This property makes FTRL a more suitable tool to address nonstationary applications. This framework has been successfully applied to highly dynamic domains such as click-through prediction (McMahan et al., 2013) and video-on-demand prediction (Tan et al., 2016).

The most notable algorithm derived from the FTRL framework is the FTRL-

Proximal algorithm (McMahan et al., 2013). In this algorithm, all learning examples contribute equally to the loss function resulting in the slow adaptation to concept drift. To enhance responsiveness to concept drift, Tan et al. (2016) introduces an exponentially decaying factor to augment the per-coordinate learning rate schedule proposed in FTRL-Proximal. The basic idea in this approach is the usage of a decaying factor to discount the contribution of previous examples. Although interesting and effective, this approach has two drawbacks.

The first drawback is that the proposed decaying rate is fixed and requires manual tuning. This process of manual tuning arises since different problems have different drifting dynamics, hence different rates of decaying are necessary. This tuning process is time-consuming and requires expertise. The second drawback is that a fixed decaying rate may not be optimal in the settings with complex dynamics (e.g. interleaved abrupt drifts and gradual drifts). In these settings, the decaying rate is expected to change from time to time to optimally adapt to the change in the drifting dynamics.

In this paper, we aim to address these two drawbacks by proposing a new adaptive online algorithm, called Follow-the-Regularized-Leader with Adaptive Decaying Proximal. We introduce two novel contributions in this algorithm:

- We derive a formula to adjust the decaying rate based on a rigorous theoretical analysis (Section 4.2). This formula is theoretically justified to ensure that the algorithm has a sublinear regret bound.

- We anticipate major drifts to reset the learning using a drift detector to monitor the performance of the algorithm (Section 4.3). We base our idea on the Drift Detection Method introduced by Gama et al. (2004).

The experimental result shows that FTRL-ADP achieves state-of-the-art performance in noisy datasets with real drifts owing to these two novelties.

The rest of the paper is structured as follows. In Section 2, we review the related work on concept drift adaptation and online machine learning. In Section 3, we present the framework of FTRL and point out 3 necessary refinements

to this framework to address the problem of concept drift. These refinements are elaborated in Section 4 and lead to the proposed FTRL-ADP algorithm. We evaluate FTRL-ADP in Section 5. Specifically, we present the controlled experiments with 8 synthetic datasets in Section 5.1; and in Section 5.2, we use 6 real-world datasets to evaluate the performance of FTRL-ADP. In Section 6, we conclude the paper and present two directions for future work.

2. Related Work

Our work aims to advance the research of Follow-the-Regularized-Leader algorithms (McMahan, 2014). Specifically, we propose an adaptive version of the state-of-the-art FTRL algorithm, TDAP (Tan et al., 2016). There are two main lines of work that are in direct relation to our work. The first line of work includes online algorithms such as PA (Crammer et al., 2006), Mirror Gradient Descent (Boyd & Vandenberghe, 2010), and FTRL-Proximal (McMahan et al., 2013). The second line of work includes the works on concept drift detection and adaptation such as DDM (Gama et al., 2004) and the work by Ditzler et al. (2015).

2.1. Online Learning

Online algorithms such as PA (Crammer et al., 2006) and CWL (Dredze et al., 2008) are single learners. The basic idea in these algorithms is the objective to find the weight that minimizes the loss associated with the most recent example. Certain constraints are enforced on the weight to stabilize the solution. Passive Aggressive (PA) algorithm (Crammer et al., 2006) ensures that the most recent example is correctly classified. The weight, which least deviates from the most recent weight, is the solution. PA-I and PA-II improve on PA by introducing the loss associated with the misclassified example into the objective function. Confidence Weighted Learning (CWL) algorithm (Dredze et al., 2008) relaxes the requirement of PA, only to ensure that the most current example is correctly classified within a certain probability. The objective is to minimize

the Kullback-Leibler divergence between the new weight distribution and the most recent one. Like PA I and PA II, the Soft Confidence Weighted Learning algorithm (Wang et al., 2012) proposes to modify the objective of CWL by introducing the loss associated with the misclassified example into the objective function.

These algorithms follow the Mirror Gradient Descent style (McMahan, 2014), which only considers the loss associated with the most recent example. As a result, they are robust under concept drift but prone to prediction errors caused by noise.

On the other hand, the FTRL framework considers all previous learning examples in the objective function (McMahan, 2014), aiming to derive the best model in hindsight. Regularized Dual Averaging (Xiao, 2010) tries to minimize the linearized losses associated with all previous learning examples. It regularizes the solution using L_1-norm and an adaptive L_2-norm. Similarly, FTRL-Proximal (McMahan et al., 2013) uses a per-coordinate learning schedule to regularize the proximal terms. As FTRL-derived algorithms factor in all previous losses, they are less responsive to prediction errors than Mirror Gradient Descent style algorithms.

Mirror Descent algorithms and FTRL algorithms are independent learners. They were proved to be equivalent and only different in the regularization scheme used (Boyd & Vandenberghe, 2010). Mirror Descent algorithms are responsive to changes, hence not robust to noise because of their complete forgetting mechanism. On the other hand, the latter approach of FTRL is more conservative and slow in adaptation to concept drifts due to the contribution of all examples. In this aspect, TDAP (Tan et al., 2016) strikes the balance between these two approaches via the usage of a decaying factor.

2.2. Concept Drift Adaptation

In the presence of concept drift, the predictive model is necessary to be updated from time to time. There are two different approaches towards model update: active and passive (Ditzler et al., 2015). The general framework of

active approach is shown in Figure 1. There are two key components to handle drifts in this framework: the component to detect concept drift (Change Detector) and the component to cope with detected drifts (Adaptation).

In supervised drift detection, the changes in the prediction error are monitored for concept drifts. Gama et al. (2004) propose Drift Detection Method (DDM), which keeps track of the minimum classification error and its standard deviation. A drift is detected if the current rate significantly deviates from the minimum rate. To address the insensitiveness of DDM to gradual drifts, Early Drift Detection Method (Baena-García et al., 2006) proposes to instead monitor the maximum distance between prediction errors and its standard deviation. Similarly, a drift is signaled if the current distance significantly deviates from the maximum distance. In another direction, the Page-Hinkley test (Page, 1954) is proposed to detect significant discrepancies between the cumulative accuracy difference and the minimum difference.

Barros et al. (2017) note that DDM suffers from the problem of insensitiveness caused by long concepts. The author proposes to periodically reset the calculation of the statistics used for DDM as a resolve to this problem. Frías-Blanco et al. (2015) explore the use of Hoeffding's bound in detecting concept drifts. The author monitors the moving average and weighted moving average of the performance measure for concept drift signals. Similarly, Pesaranghader & Viktor (2016) use Hoeffding's bound to threshold the probability of true predictions. Nishida & Yamauchi (2007) use a sliding window to keep track of the misclassification probability. The current probability is compared to the probability in the past (window size apart) to detect concept drifts. Pears et al. (2014)

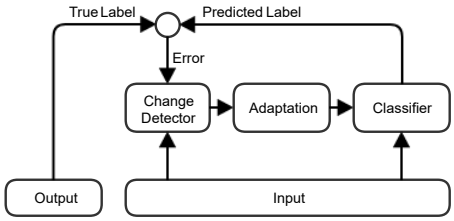

Figure 1: The General Framework of the Active Approach towards Concept Drift Adaptation.

propose a method based on reservoir sampling to detect sequential changes. This is an improvement upon their previous work in Sakthithasan et al. (2013).

In unsupervised drift detection, the underlying distribution of the data generating process (or the input) is monitored for concept drift. Masud et al. (2011) aim to watch out for sizable regions of unknown classes using outlier detection techniques. Similarly, Sethi et al. (2016) leverage on clustering techniques to look for unknown regions. In another direction, independent features are monitored in two different chunks for changes of concept: the current chunk and the reference chunk. Lee & Magoulès (2012) use Pearson correlation to evaluate the correlation between the current chunk and the reference chunk. Similarly, Ditzler & Polikar (2011) use Hellinger distance to achieve the same purpose. In general, supervised drift detection methods are more accurate than unsupervised methods at the cost of labeled data.

In passive drift adaptation, there are two main categories of passive learners: ensemble learner and single learner. The SEA ensemble classifier (Street & Kim, 2001) is among the first proposals for ensemble learners. The ensemble creates a new learner for every new batch of data. To make room for new learners, existing learners are removed based on age or prediction accuracy. In another direction, Pocock et al. (2010) propose the Online Nonstationary Boosting Algorithm (ONSBoost), which is adapted from the Online Boosting Algorithm (Oza & Russell, 2001), to address the challenges in nonstationary settings. Santos et al. (2014) aim to address the problem of frequent changes in data streams. The authors propose a method to more efficiently distribute the instances among the experts to allow rapid recovery of expert's performances. Barros et al. (2016) aim to improve the precision by exploring variations of the voting strategy originally proposed in Santos et al. (2014). Minku & Yao (2012) exploit diversity by maintaining ensembles with different diversity levels to achieve two purposes: lower false positive and faster recovery from drifts. Frías-Blanco et al. (2016) propose an ensemble method to replace the worse learner in the ensemble with a new learner when a drift is detected.

Compared to the ensemble approach, the single learner approach is often less

computationally expensive. Domingos & Hulten (2000) propose Concept Drift Very Fast Decision Tree (CDVFDT) to mine nonstationary streaming data. This algorithm is adapted from the Very Fast Decision Tree, which is a popular tree-based solution for stream data mining. CDVFDT uses an adaptive window size to cut off least recent learning examples (Hulten et al., 2001). In another direction, Lim et al. (2013) propose a modification to the originally stationary Extreme Machine Learning. The basic idea in this work and similar works (Liu et al., 2016; Pavlidis et al., 2011) in this approach is to use a fixed or variable forgetting factor to increasingly discount the contribution of previous examples.

In summary, our approach further develops the idea in TDAP. Specifically, we propose a mechanism to adaptively control the decaying rate. This adaptive mechanism is based on the drift detector DDM and eliminates the need to manually tune the decaying rate (Section 4.3).

3. Problem Setting

In this section, we revise the online learning framework of Follow the Regularized Leader, formulated in the language of online convex optimization. We then present 3 necessary refinements to this framework. The refinements are designed to allow real-time online prediction and quick adaptation to concept drift.

The framework of online convex optimization can be formulated as follows (Shalev-Shwartz, 2011). We need to design an algorithm that makes a series of optimal predictions, each at one time step. At time step t, the algorithm makes a prediction, which is the weight vector w_t. A convex loss function $l_t(w)$ is then exposed to the algorithm. Finally, the algorithm suffers loss $l_t(w_t)$ at the end of time step t (Algorithm 1). The algorithm should be able to learn from the losses in the past to make better predictions over time.

The optimality of a series of predictions is conditioned on the minimization of the regret with respect to the best classifier in hindsight (Equation 1). $\mathbf{Regret}_t$ in Equation 1 is the difference between the total loss incurred up to time t and

Algorithm 1 Online Algorithm

1: **for** t = 1,2,... **do**
2: Make a prediction w_t
3: Receive the lost function $l_t(w)$
4: Suffer the lost $l_t(w_t)$

the supposed loss incurred by the best possible prediction in hindsight w^*.

$$\textbf{Regret}_t = \sum_{s=1}^{t} l_s(w_s) - \sum_{s=1}^{t} l_s(w^*) \tag{1}$$

Since the future loss functions are unknown, the greedy approach to achieve the objective of minimizing the regret is to leverage on the prediction that incurs the least total loss on all past rounds. This approach is called Follow-the-Leader (FTL), in which the leader is the best prediction with respect to all past loss functions. In some cases, this simple formulation may result in algorithms with undesirable properties such as rapid changes in the predictions (Shalev-Shwartz, 2011), which lead to overall high regret. To fix this problem, some regularization function is usually added to stabilize the prediction. The second approach is called Follow-the-Regularized-Leader (FTRL), which is formulated in Equation 2 (the version presented here is the proximal version (McMahan, 2014), which can be conveniently modified to handle the problem of concept drift).

$$w_{t+1} = \underset{w}{\operatorname{argmin}} \left\{ \sum_{s=1}^{t} l_s(w) + \sum_{s=1}^{t} \sigma_{t,s} \|w - w_s\|_2^2 \right\} \tag{2}$$

In Equation 2, $\sum_{s=1}^{t} \sigma_{t,s} \|w - w_s\|_2^2$ is the proximal regularization used to stabilize the solution weight, $\sigma_{t,s}$ the coefficient used to control the amount of regularization and w_t the predicted weight at time t.

We next point out 3 refinements to this framework necessary to allow for real-time prediction under concept drift.

1st refinement. In general, Equation 2 does not have a closed-form solution. For example, when $l_s(w)$ is the cross-entropy loss (investigated in Section 4.1), the derivative of the objective function in Equation 2 is a transcendental function

(Boyd, 2014), which does not have a closed-form solution in general. Therefore, we need to run some sort of gradient descent at each time step to solve for w. This operation is computationally expensive and not suitable for real-time applications. This issue can be addressed by approximating the loss function using its linearization at w_t. Consequently, the optimization problem can be solved in closed-form and enjoys an efficient recursive structure. The recursive algorithm is constant in time and memory, which makes it suitable for real-time applications. This refinement is elaborated in Section 4.1.

2nd refinement. The proximal terms in Equation 2 regularize the current solution by enforcing losses on deviations from past solutions. If $\sigma_{t,s}$ is a constant, all past solutions will have an equal regularizing effect. However, in the face of concept drift, we argue that it is more appropriate for the current solution to be biased towards the most recent past solutions, to counter the drifting effect. This idea can be realized by using an exponentially decaying factor $\sigma_{t,s} = \gamma^{t-s}$ (with $0 < \gamma < 1$) to increasingly discount the proximal terms. This refinement is illustrated in Section 4.2.

3rd refinement. Different applications may have different drifting dynamics and require different decaying rates. Therefore, we propose a data-driven mechanism to adaptively adjust the decaying rate based on two novel ideas: sublinear regret analysis and drift detection. The latter idea is adapted from the method introduced by Gama et al. (2004). This drift detection method has been shown to achieve the best performance across different tasks in (Gonçalves Jr et al., 2014). This refinement is elaborated in Section 4.3.

4. Methodology

4.1. Linearization of Loss Function

We cast an online classification problem as an online convex optimization problem as follows. At time t, the algorithm makes prediction w_t and receives input x_t. The true value y_t is then revealed to the algorithm. The loss function

$l_t(w)$ associated with time t is defined in terms of x_t and y_t (Equation 3). Finally, the cost incurred at the end of time t is $l_t(w_t)$. The final optimization problem is shown in Equation 5. The solution to this problem is the Follow the Regularized Leader with Decaying Proximal algorithm (Algorithm 2).

$$\begin{gathered} l_t(w) = -y_t \log(p) - (1 - y_t) \log(1 - p) \\ \text{in which } p = \text{sigmoid}(w^\top x_t) \end{gathered} \tag{3}$$

Compared to Equation 2, Equation 5 has the actual loss function $l_t(w)$ replaced by its linear approximation at w_t, which is:

$$\begin{gathered} l_t(w) \approx l_t(w_t) + \nabla l_t(w_t)^\top (w - w_t) = g_t^\top w + l_t(w_t) - g_t^\top w_t \\ \text{in which } g_t = \nabla l_t(w_t) \end{gathered} \tag{4}$$

The constant term $(l_t(w_t) - g_t^\top w_t)$ is omitted in Equation 5. Overall, this linear approximation is to enable the derivation of a closed-form solution at each time step, which is not possible with the original problem in Equation 2.

$$\begin{gathered} w_{t+1} = \underset{w}{\text{argmin}} \left\{ g_{1:t}^\top w + \lambda_1 \|w\|_1 + \frac{1}{2}\lambda_2 \|w\|_2^2 + \frac{1}{2}\lambda_p \sum_{s=1}^{t} \sigma_{t,s} \|w - w_s\|_2^2 \right\} \\ \text{in which } g_{1:t}^\top = \sum_{i=1}^{t} g_t^\top \text{ and } \sigma_{t,s} = \gamma^{t-s} \end{gathered} \tag{5}$$

FTRL-DP utilizes 3 different regularizers to serve 3 different purposes. The 1st regularizer is the L_1-norm used to enforce sparsity on the solution. The 2nd regularizer is the L_2-norm used to favor low variance solutions that have small weights. The 3rd regularizer is the proximal terms used to stabilize the solution by not allowing it to deviate too much from past solutions. In our work, we propose an adaptive mechanism (Section 4.3) to automatically tune the parameter γ of the proximal coefficient $\sigma_{t,s}$, which is fixed and manually tuned in the work introduced by Tan et al. (2016). The solution to the objective function of FTRL-DP is stated in Theorem 1.

Theorem 1. *The optimization problem in Equation 5 can be solved in the following closed form:*

$$w_{t+1,i} = \begin{cases} 0 & \text{if } \|z_{t,i}\|_1 \leq \lambda_1 \\ -\frac{z_{t,i} - \lambda_1 \text{sign}(z_{t,i})}{\lambda_2 + \lambda_p \frac{1-\gamma^t}{1-\gamma}} & \text{otherwise.} \end{cases} \tag{6}$$

$$\text{in which } z_t = g_{1:t} - \lambda_p \sum_{s=1}^{t} \gamma^{t-s} w_s$$

The proof of Theorem 1 is provided in Appendix A. Theorem 1 provides a formula to compute w_t at each time step. This formula has a recursive structure and can be efficiently realized by Algorithm 2.

Algorithm 2 Follow the Regularized Leader with Decaying Proximal

1: Input: $\lambda_1, \lambda_2, \lambda_p, \gamma$
2: Initialize $v_t = h_t = z_t = w_t = 0$, $r_t = 1$
3: **for** $t = 1, 2, ...$ **do**
    /* Make prediction */
4: Make prediction $w_{t,i} = \begin{cases} 0 & \text{if } \|z_{t,i}\|_1 \leq \lambda_1 \\ -\frac{z_{t,i} - \lambda_1 \text{sign}(z_{t,i})}{\lambda_2 + \lambda_p r_t} & \text{otherwise.} \end{cases}$
5: Receive x_t, y_t /* and suffer the lost $l_t(w_t)$ in Equation 3*/
    /* Update parameters */
6: $p_t = \frac{1}{1+e^{-w_t^\top x_t}}$
7: $g_t = (p_t - y_t)x_t$
8: $v_t = v_{t-1} + g_t$
9: $h_t = \gamma h_{t-1} + w_t$
10: $z_t = v_t - \gamma_p h_t$
11: $r_t = \gamma r_{t-1} + 1$

Algorithm 2 is initialized with 5 parameters (v_t, h_t, z_t, w_t, and r_t). These parameters are buffered and iteratively updated after each time step. In each time step, the algorithm firstly makes a weight prediction w_t (Line 4). It then uses the new weight w_t and the feature vector x_t (Line 5) to compute the probability p_t of class 1 (Line 6). The difference between the predicted probability and the true label is used to adjust the rest 4 buffered parameters (v_t in Line

8, h_t in Line 9, z_t in Line 10, and r_t in Line 11). It is noted that caching the intermediate computation results in v_t, h_t, z_t, and w_t allows the computation cost to remain constant in each step.

In the next section (Section 4.2), we prove that the regret of FTRL-DP can be sublinearly bound via a proper choice of the decaying rate. This result leads to the proposed adaptive formula to update the decaying rate γ.

4.2. Decaying Proximal Regularization

The effect of the exponentially decaying proximal, $\sum_{s=1}^{t} \gamma^{t-s} \|w - w_s\|_2^2$ is to bias the current solution towards the most recent past solutions. This refinement is the novelty of our work compared to the FTRL-Proximal algorithm proposed by McMahan et al. (2013). McMahan et al. proposed the following per-coordinate learning schedule to regulate the proximal terms:

$$\sigma_{s,i} = \Big(\sqrt{\sum_{j=0}^{s} (g_{j,i}^2)} - \sqrt{\sum_{j=0}^{s-1} (g_{j,i}^2)} \Big) = \frac{g_{s,i}^2}{\sqrt{\sum_{j=0}^{s} (g_{j,i}^2)} + \sqrt{\sum_{j=0}^{s-1} (g_{j,i}^2)}}$$

The drawback of this schedule is that $\sigma_{s,i}$ tends to decrease with time, given roughly the same gradient $g_{s,i}$, since the denominator of $\sigma_{s,i}$ monotonically increases. Consequently, the current solution is based towards the least recent solutions, instead of the most recent ones. This consequence is unfavorable as it enhances the effect of concept drift rather than counterbalances it.

The decaying rate γ in Equation 5 controls the stability-plasticity balance of FTRL-DP. Stability refers to the ability to perform robustly in noisy environments. Plasticity refers to the ability of the algorithm to quickly adapt to drifting concepts. We can see that these two requirements are irreconcilable as they require the decaying rate to change in two opposite directions. A larger value of decay rate γ means that the model is more robust to noise but less responsive to drifting concepts. On the contrary, for smaller γ, the model can quickly respond to concept drift at the cost of sensitivity to noise.

In practice, we need to manually search for the optimal decay rate to suit the specific application. This process is time-consuming. In Theorem 2, we prove a result that sheds some light on how to choose a good decaying rate.

Theorem 2. *Suppose that* $\|w_t\|_2 \leq R$ *and* $\|g_t\|_2 \leq G$. *With* $\lambda_1 = \lambda_2 = 0$ *and* $\lambda_p = 1$, *we have the following regret bound for FTRL-DP:*

$$Regret(w^*) \leq 2R^2 \frac{1-\gamma^T}{1-\gamma} + \frac{G^2}{2} \frac{1-\gamma}{\gamma^T} \sum_{t=1}^{T} \frac{\gamma^t}{1-\gamma^t} \tag{7}$$

We prove that a proper choice of γ *can lead to a sublinear growth of the expression on the right-hand side of Inequality 7. Specifically, if we choose* $\gamma = 1 - \frac{\ln T}{2T}$, *we have the following sublinear regret bound:*

$$Regret(w^*) \leq 4R^2 \frac{T}{\ln T} + \frac{G^2}{2} \frac{(1+\ln T)}{(1-\frac{\ln T}{2T})^T} \tag{8}$$

The proof to Theorem 2 is provided in Appendix B. Theorem 2 means that if we are allowed to change the decaying rate of FTRL-DP at each time step, the specific choice of $\gamma = 1 - \frac{\ln T}{2T}$ will ensure that the average regret of FTRL-DP (dividing both sides of Inequality 8 by T) diminishes with time.

Therefore, in the next section (Section 4.3), we propose an update rule to adjust the decaying rate of FTRL-DP based on this theorem ($\gamma = 1 - \frac{\ln T}{2T}$). In addition, we also consider the effect of drifting speed and baseline noise level on the decaying rate.

4.3. Adaptive Decaying Rate

We propose Algorithm 3 as an adaptive mechanism to adjust the decaying rate. In Line 6 of Algorithm 3, the decaying rate increases following the rule derived in Theorem 2. This update rule to ensure sublinear regret bound is only meaningful in stationary settings. The rest of the algorithm is to address nonstationary settings by resetting the update rule on detected drifts.

In nonstationary settings, the optimal decaying rate depends on two factors: the drifting speed and the noise level. The relationship between the drifting

Algorithm 3 Adaptive Decaying Rate

Input: p_t, y_t
Initialize $p_{min} = s_{min} = 1$, $count = error = count_{warn} = error_{warn} = 0$
for $t = 1, 2, ...$ **do**
    $count = count + 1$
    $error = error + \mathbb{1}_{\|p_t - y_t\|_1 > 0.5}$
    $\gamma = 1 - \frac{\ln(count)}{2 \times count}$
    if $count > 30$ **then**
        $p_i = \frac{error}{count}$
        $s_i = \sqrt{\frac{p_i(1-p_i)}{count}}$
        $sum = p_i + s_i$
        $\gamma = 0.99\gamma + 0.01 p_i$
        if $sum < p_{min} + s_{min}$ **then**
            $p_{min} = p_i$
            $s_{min} = s_i$
        if $sum < p_{min} + 2s_{min}$ **then** /* Trigger warning stage */
            $count_{warn} = 0$
            $error_{warn} = 0$
        else
            $count_{warn} = count_{warn} + 1$
            $error_{warn} = error_{warn} + \mathbb{1}_{\|p_t - y_t\|_1 > 0.5}$
            if $sum > p_{min} + 3s_{min}$ **then** /* Trigger critical stage */
                $count = count_{warn}$
                $error = error_{warn}$
                $p_{min} = 1$
                $s_{min} = 1$

speed and the decaying rate is reversely proportional. Namely, if the drifting speed is high, the decaying rate should be low so that the algorithm quickly forgets past solutions. On the other hand, the baseline noise level and the decaying rate are proportionally related. In a noisy environment, the decaying rate should be high for the learner to stay conservative to prediction errors. This analysis highlights the opposite effect of the drifting speed and the baseline noise level on the decaying rate, which is usually referred to as the stability-plasticity dilemma encountered in adaptive learning systems (Grossberg, 2013).

4.3.1. Drifting Speed

The basic idea is to reset the update rule (thereby decreasing the decaying rate) when the learner starts to degrade. To this end, we keep track of the

current error rate, p_i, which is computed in Line 8 of Algorithm 3. We then consider each prediction as a Bernoulli trial with mean p_i and standard deviation $s_i = \sqrt{\frac{p_i(1-p_i)}{count}}$ (Line 9).

According to statistical learning theory (Shalev-Shwartz & Ben-David, 2014), in stationary settings, the variance of the error rate must decrease when the number of examples increases. However, if the error rate deviates significantly from the current rate, a major drift may have happened. In this case, we are supposed to decrease the decay rate to speed up the forgetting process. This idea was first introduced by Gama et al. (2004).

To be more sensitive to low drifting speed, the pair (p_{min}, s_{min}) with minimum sum $p_{min} + s_{min}$ (Line 12 to Line 14) is used as the reference distribution, instead of the most recent pair (p_i, s_i). At any point, if $p_i + s_i > p_{min} + 2s_{min}$, a concept drift is anticipated and the warning stage is triggered (Line 15). The factor of 2 is chosen to ensure 95% confidence of drift. If $p_i + s_i > p_{min} + 3s_{min}$, a drift is detected and the learning is reset (Line 21 to Line 25). The factor of 3 is chosen to ensure 99% confidence of drift.

4.3.2. Noise Level

The basic idea is to increase the decaying rate when the noise level increases and vice versa. To realize this idea, we establish a simple linear relationship between the decaying rate and error rate p_i (Line 11). The contribution amount of the error rate to the decaying rate ($0.01 = 1\%$) is empirically determined and found to work well across all datasets.

4.4. Space and Time Complexity of FTRL-ADP

Both Algorithm 2 and Algorithm 3 have constant space and time complexities. Algorithm 2 is a recursive procedure to compute the latest solution weight by leveraging on the proceeding one. This is the result of the linearization trick to avoid running gradient descent at each time step in solving Equation 2. On the other hand, Algorithm 3 is a recursive procedure to estimate the current error rate. Combining Algorithm 2 and Algorithm 3 results in Follow the Reg-

Dataset	Number of Examples	Number of Expanded Features	Duration	Drift Type
Stationary	20k	2	N/A	Stationary
Dynamic	70k	2	N/A	Mixed Drift
SEA (Street & Kim, 2001)	40k	3	N/A	Abrupt Drift
Hyperplane (Fan, 2004)	40k	10	N/A	Gradual Drift
Noisy Stationary	20k	2	N/A	Stationary
Noisy Dynamic	70k	2	N/A	Mixed Drift
Noisy SEA (Street & Kim, 2001)	40k	3	N/A	Abrupt Drift
Noisy Hyperplane (Fan, 2004)	40k	10	N/A	Gradual Drift
Spam Dataset (Katakis et al., 2010)	9,324	39,916	Unknown	Unknown
URL Dataset (Ma et al., 2009)	616,000	3,231,960	30 days	Unknown
Airline Dataset (Wickham, 2011)	539,383	613	20 years	Unknown
Weather Dataset (Ditzler & Polikar, 2013)	18,159	8	50 years	Unknown
Electricity Dataset (Harries, 1999)	27,549	60	2.5 years	Unknown
Intrusion Dataset (Lippmann et al., 2000)	494,021	122	7 weeks	Unknown

Table 1: 12 Evaluated Datasets.

ularized Leader with Adaptive Decaying Rate algorithm, which therefore also enjoy constant space and time complexities.

5. Experiment

We use 8 synthetic and 6 real-world datasets to evaluate the performance of FTRL-ADP. Table 2 summarizes the characteristics of the datasets. These datasets are available at https://git.io/v51Qv.

In the experiments with 8 synthetic datasets, we evaluate 3 different versions of FTRL-ADP (FTRL-P with fixed $\gamma = 1$, FTRL-DP with fixed $\gamma < 1$, and FTRL-ADP with adaptive γ) against TDAP (Tan et al., 2016) and FTRL-Proximal (McMahan et al., 2013). All these algorithms are derived from the FTRL framework.

In the experiments with 6 real-world datasets, we compare FTRL-ADP to 6 other online algorithms: Confident Weighted Learning (Dredze et al., 2008), PA I, PA II (Crammer et al., 2006), ALMA (Gentile, 2001), TDAP (Tan et al., 2016), and FTRL-Proximal (McMahan et al., 2013). These algorithms are single learners and comparable to FTRL-ADP.

We compare the performances of the algorithms using average cumulative error (or prequential error in (Gama et al., 2013)). Specifically, we consider the evolution of the average of the total error from the first learning example to the current learning example as the performance metric. We use 0.5 as the threshold to binarize the probabilistic predictions when applicable.

5.1. Synthetic Experiments

The objective of this experiment is to study how the optimal decaying rate depends on the drifting speed and the noise level. To this end, we closely control the amount of noise added to the datasets and the underlying drifting level. We aim to analyze the experimental results to highlight the improvements of FTRL-ADP over existing algorithms in the FTRL family.

For a fair comparison, we use the same amount of regularizations ($L_1 = 1$, $L_2 = 1$ and $L_p = 1$) in the 5 algorithms, namely: FTRL-P ($\gamma = 1$), FTRL-DP ($\gamma < 1$), FTRL-ADP (adaptive γ), TDAP, and FTRL-Proximal. The decaying rates of FTRL-DP and TDAP are set to the same value of 0.9.

5.1.1. Datasets

Other than abrupt and gradual, concept drifts can also be classified as real and virtual (Gama et al., 2014). In a classification problem, the objective is to determine the posterior probability $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$. In this context, real drift refers to the shift in $P(y|x)$. Virtual drift refers to the change in the data generating process $P(x)$. Virtual drift may or may not co-occur with real drift.

We use two publicly available synthetic datasets to evaluate the case of real drift: SEA dataset (Street & Kim, 2001) and Hyperplane dataset (Fan, 2004). The SEA dataset contains abrupt real drifts. The drifts in the Hyperplane dataset is also real but the drifting speed is gradual. To study more complex drifting dynamics, we generate a new dataset that contains both abrupt and gradual virtual drifts. In addition, we also generate a stationary dataset to serve as the baseline for evaluation.

SEA dataset. This dataset simulates abrupt real drifts. It involves three features, in which only the first two are correlated with the labels. The generated points are separated into two classes using the threshold f on the sum of the first two features. For a given point (f_1, f_2, f_3), if $f_1 + f_2 < f$, it is labelled as class 0 and class 1 otherwise. Consequently, the decision line only translates in the feature space, which is different from the rotating decision line in the

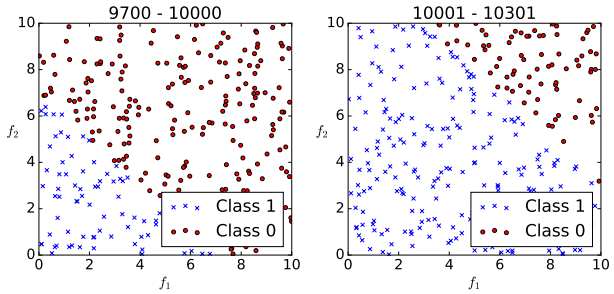

Figure 2: Visual illustration of an abrupt drift in the SEA dataset. The first two features, f_1 and f_2, among the three given features are used to make the scatter plot. The figure shows the decision line shifting abruptly at data point 1000.

Hyperplane dataset to be introduced next. A drift is simulated by changing the value of the threshold f. We use the framework of Massive Online Analysis (MOA in (Bifet et al., 2010)) to generate a dataset of 40k examples with abrupt concept changes at 10k and 20k and 30k using the thresholds 7, 13, 6, and 12 in succession. A visual representation of the first two features of the SEA dataset is shown in Figure 2. The plot on the left of Figure 2 shows the data points from 9700 to 10000 and the plot on the right the data points from 10001 to 10301. Each cross marker represents a class 1 data point and circle marker class 0 data point. We can see that the decision line shifts abruptly at data point 10000, which is the first in the sequence of three simulated drifts.

Hyperplane dataset. This dataset resembles the SEA dataset except that the decision line can also rotate in the feature space. For a given point $(f_1, f_2, ..., f_n)$, if $\sum_{i=1}^{n} f_i w_i < w_0$, it is labelled as class 0 and class 1 otherwise. The drifting dynamics is controlled by varying the weights following the law $w_i = w_i + \sigma d$, in which σ is the probability that the drifting direction is reversed and d the amount of drift. Using MOA, we generate a dataset of 40k examples with 10 features, $\sigma = 10\%$, and $d = 30\%$. A visual representation of the Hyperplane dataset is shown in Figure 3. The plot on the left of Figure 3 shows the data points from 10000 to 10300 and the plot on the right the data points from 30000 to 30300. The first 2 features of the 10-dimensional feature vectors are used to make the plots in Figure 3. The decision line shifts gradually from data point

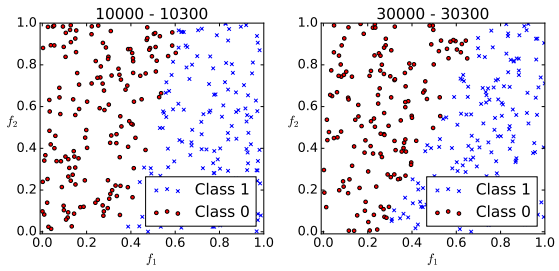

Figure 3: Visual illustration of gradual drift in the Hyperplane dataset. The first two features among the given ten features are used to make the scatter plot. The decision line shifts gradually from data point 10000 to 30300 in this case.

10000 to 30300 in this case.

The above two datasets are commonly used to evaluate the capability to handle concept drifts of online algorithms. However, they only represent the simple scenarios of a single type of drift. We propose a new dataset that simulates a scenario with more complicated drifting dynamics.

Dynamic dataset. In this dataset, we interleave abrupt drifts with gradual drift. The data points are generated by two moving Gaussian distributions to simulate real concept drifts. The level of noise is controlled by manipulating the standard variation of the Gaussian distributions. The speed of drifting varies from 0 to 1 and controls how fast the means of the Gaussians move in the feature space (0 for stationary and 1 for abrupt drift).

Using this method, we generate a dataset of 70k examples with the following drifting sequence: stationary, gradual, abrupt, gradual, abrupt, gradual, and abrupt in succession. The mean of the first Gaussian distribution, which generates the data points of class 1, oscillates back and forth between point [1,1] and point [-1,1] in the drifting sequence. Similarly, the second Gaussian distribution oscillates between [-1,-1] and [1,-1]. The speeds of the 3 gradual drifts in the sequence are 0.1, 0.2, and 0.3, respectively in this order.

The Dynamic dataset contains two features and is visualized in Figure 4. In Figure 4, the plot on the left visualizes the data points from 10000 to 10300, the plot at the middle the data points from 14000 to 14300, and the plot on the

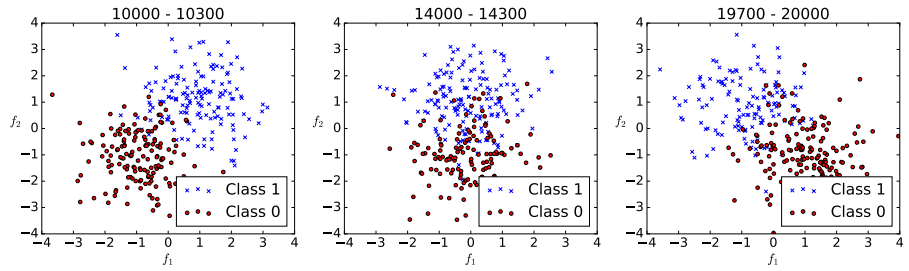

Figure 4: Visual illustration of a gradual drift in the Dynamic dataset. The plot on the left, at the middle, and on the right correspondingly visualize the data points from 10000 to 10300, from 14000 to 14300, and from 19700 to 20000. The Gaussian distribution of class 1 moves gradually from right to left and vice versa for the Gaussian distribition that generates the data points of class 0.

right the data points from 19700 to 20000. We can see that the positions of the Gaussian distributions are moving gradually from right to left for class 1 and vice versa for class 0.

Stationary dataset. We also use the same method as in the Dynamic dataset to generate a stationary dataset of 20k examples. However, to ensure stationarity, the positions of the two Gaussian means are not moving in this dataset. We achieve this by setting the drifting speed to zero.

For each of the 4 datasets above, we generate two versions: the first version with a low level of noise and the second version with a high level of noise. The amount of noise in the SEA and the Hyperplane datasets is controlled by the manipulating the probability that the given label of a data point is switched from the true class 0 to the wrong class 1 and vice versa. In the MOA package, we set this probability to 15% in both noisy SEA and noisy Hyperplane datasets and 0% in their low noise versions. On the other hand, the amount of noise in the Dynamic and the Stationary datasets is naturally controlled by varying the standard deviations of the Gaussian distributions, used to generate the data points. We set the standard deviations of the Gaussian distributions to 1.5 in both noisy Dynamic and noisy Stationary datasets and 0.8 in their low noise versions. In summary, we generate 8 synthetic datasets, which is the 8 combinations of 4 different drift conditions (no drift, gradual drift, abrupt drift,

and mixed drift) and 2 noise conditions (low level of noise and high level of noise).

5.1.2. Discussion

We use the same parameters to generate each dataset 30 times. Figure 5 shows the evolution of the prediction errors (average score of 30 iterations) of all algorithms on all synthetic datasets. Table 2 summarizes the errors using means and standard deviations. The bold numbers indicate the best performers.

We can see that FTRL-P and FTRL-Proximal outperform FTRL-ADP, FTRL-DP, and TDAP in the two Stationary datasets (Figure 5). The gaps in the prediction errors are more announced in the noisy Stationary dataset. This result validates our claim that high level of noise requires the learner to be conservative and less responsive to prediction errors. In this case, the stability of FTRL-P and FTRL-Proximal, owing to the omission of the decaying factor (or $\gamma = 1$), explains their best performance.

At the same time, we can see that FTRL-DP and TDAP become too paranoid with prediction errors as it tries to correctly classify every example by quickly forgetting old weights ($\gamma < 1$). Overall, they make the most number of mistakes due to this short-sightedness. On the contrary, FTRL-ADP performs comparably to FTRL-P and FTRL-Proximal while avoiding the shorted-sightedness of FTRL-DP of TDAP, due to its adaptive mechanism (Friedman test (Demšar, 2006) with 5% significance level). This mechanism increases the decaying rate of FTRL-ADP to asymptotically reach 1 ($\gamma = 1 - \frac{\ln T}{2T}$) to best perform in the stationary case. There are no concept drifts detected in the stationary case according to Figure 6.

The addition of noise degrades the performance of all learners in all cases. The error rates of FTRL-P, FTRL-Proximal, FTRL-DP, TDAP, and FTRL-ADP increase by 15.55%, 15.46%, 19.16%, 21.22%, and 16.46% in the noisy Stationary dataset. FTRL-ADP are more robust than FTRL-P and FTRL-DP but less robust than FTRL-P and FTRL-Proximal. This result can be explained by the conservativeness when $\gamma = 1$ and the responsiveness when $\gamma < 1$.

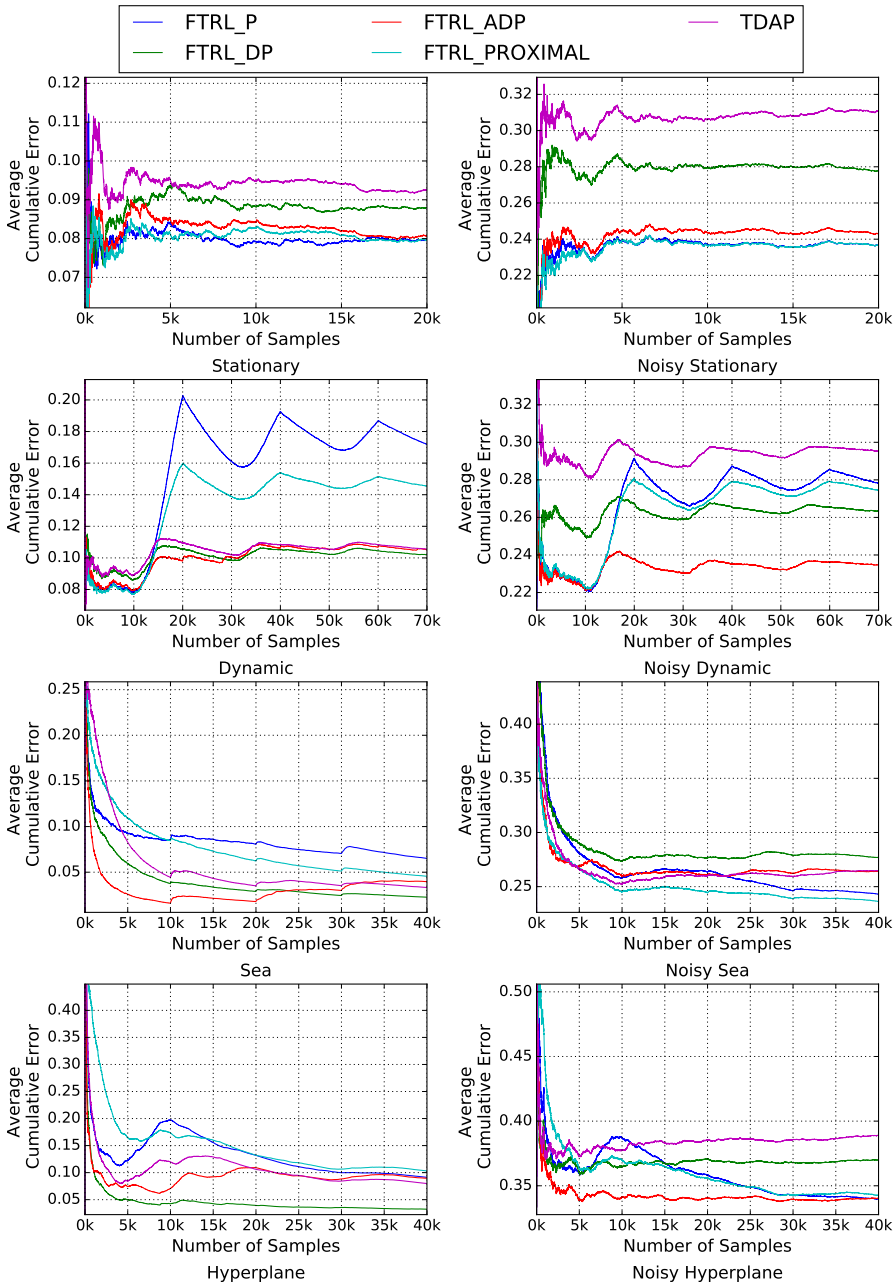

Figure 5: Error Rates of FTRL-P, FTRL-Proximal, FTRL-DP, TDAP and FTRL-ADP on 8 Synthetic Datasets.

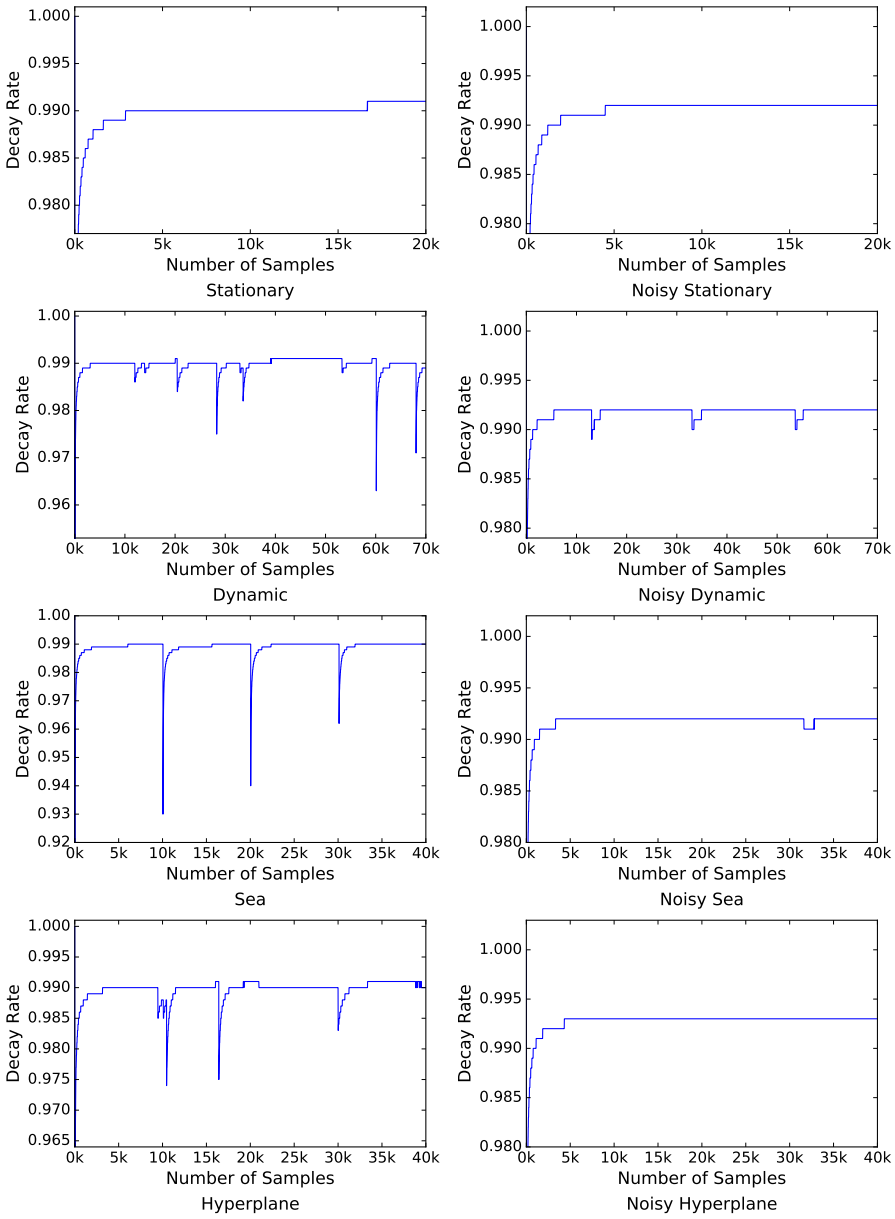

Figure 6: Evolution of Decaying Rates of FTRL-ADP on 8 Synthetic Datasets.

On the contrary, FTRL-P and FTRL-Proximal perform the worst in the Dynamic dataset. After the onsets of the abrupt drifts (localized by the plunges in the decaying rate (Figure 6)), the performances of all the learners decrease. However, the error rates of FTRL-Proximal and FTRL-P sharply increase while

Dataset	FTRL-P	FTRL-DP	FTRL-ADP	FTRL-Proximal	TDAP
Stationary	**8.05±0.38**	8.72±0.46	8.25±0.78	8.08±0.96	9.49±0.93
Dynamic	15.54±3.91	10.14±0.8	**10.02±1.06**	13.25±2.68	10.41±0.67
Sea	8.4±1.9	3.93±2.77	**3.25±2.14**	7.54±3.34	5.58±4.19
Hyperplane	13.36±4.53	**4.62±3.71**	9.35±2.74	15.15±6.48	10.72±4.23
Noisy Stationary	23.6±0.78	27.88±0.84	24.19±0.9	**23.54±0.94**	30.71±0.7
Noisy Dynamic	26.78±2.1	26.25±0.54	**23.41±0.71**	26.42±1.87	29.36±0.77
Noisy Sea	26.51±3.16	28.51±2.49	26.78±1.88	**25.1±2.27**	26.64±2.15
Noisy Hyperplane	35.98±2.01	36.96±0.71	**34.24±1.49**	36.03±2.4	38.43±0.83

Table 2: Means and Standard Deviations of Error Rates of FTRL-P, FTRL-Proximal, FTRL-DP, TDAP and FTRL-ADP on 8 Synthetic Datasets.

the error rates of TDAP, FTRL-DP, and FTRL-ADP quickly plateau. For example, after the first drift at example 1300^{th} in the Dynamic dataset, the error rates of FTRL-Proximal and FTRL-P steeply increase from around 0.08 to 0.17 while the error rates of TDAP, FTRL-DP, and FTRL-ADP settle at around 0.10. This result supports our claim that conservative learners (FTRL-P and FTRL-Proximal) do not perform well when drifting concepts are present.

Here, we can see the opposite effects of drifting speed and noise level on the conservativeness of the algorithm: they require the conservativeness to change in two opposite directions. This is a manifestation of the Stability-Plasticity dilemma usually encountered in adaptive learning systems (Grossberg, 2013).

In the SEA and Hyperplane datasets, the performances of the algorithms are mixed. FTRL-DP, TDAP, and FTRL-ADP usually outperform FTRL-P and FTRL-Proximal in the cases with less noise and vice versa in the noisy cases. It is interesting to note that the FTRL-P and FTRL-Proximal are rather robust to virtual drifts (contained in the SEA and Hyperplane datasets). Their error rates do not sharply increase at the onsets of the drifts like in the Dynamic datasets, which contain real drifts. Due to this reason, although FTRL-ADP is able to detect abrupt drifts (the plunges in the decaying rate (Figure 6)), these positive detections do not help FTRL-ADP outperform FTRL-P and FTRL-Proximal significantly in the cases of SEA and Hyperplane datasets.

In addition, the performances of FTRL-DP, TDAP, and FTRL-ADP are comparable in the 2 nonstationary datasets, Dynamic and SEA, with less noise

(Friedman test with 5% significance level). When the noises are injected into these 2 datasets, FTRL-ADP becomes significantly outperforming FTRL-DP and TDAP. This result again validates our claim that high level of noise penalizes responsive learners the most. Compared to FTRL-DP and TDAP, FTRL-ADP is more robust to noise due to its decaying rate being adjusted according to the level of noise ($\gamma = 0.99\gamma + 0.01p_i$).

Therefore, we can conclude that the optimal decaying rate depends on the drifting speed and the noise level. Higher drifting speed requires the learner to be more responsive, which can be achieved by decreasing the decaying rate. On the other hand, noisy cases require the learner to be conservative by increasing the decaying rate. These two factors are considered when designing the adaptive scheme (Algorithm 3) to adjust the decaying rate (specifically, Line 21 to consider drifting speed and Line 11 to consider noise level).

5.2. Real-world Experiments

The objective of this experiment is to evaluate FTRL-ADP against 6 state-of-the-art online algorithms: Confident Weighted Learning (Dredze et al., 2008), PA I, PA II (Crammer et al., 2006), ALMA (Gentile, 2001), TDAP (Tan et al., 2016), and FTRL-Proximal (McMahan et al., 2013). We also include the evaluation of DDM-enabled Hoeffding Tree (Hulten et al., 2001) as the baseline drift-enabled classifier for comparison.

We use the same amount of L_1, L_2 and L_p regularizations in FTRL-ADP, TDAP, and FTRL-Proximal. The decaying rate γ of TDAP and parameter α of ALMA is searched in $\{0.5, 0.55, ..., 0.95\}$. The parameters C of PA-I and PA-II and η of CWL used are the best performing values in $\{10^{-3}, 10^{-2}, ..., 10^{1}\}$. For the DDM-enabled Hoeffding Tree, we use the default parameters of DDM and Hoeffding Tree provided by the MOA package.

5.2.1. Datasets

We evaluate the algorithms on the following datasets: Spam (Katakis et al., 2010), URL (Ma et al., 2009), Airline (Wickham, 2011), Weather (Ditzler &

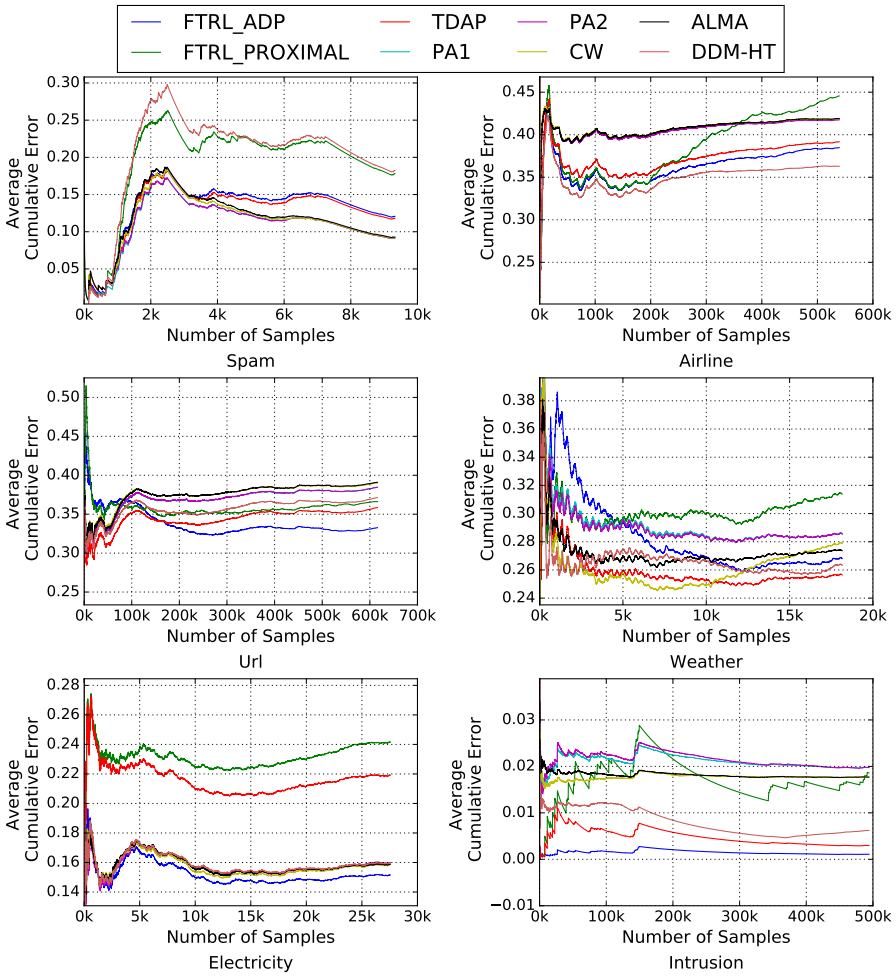

Figure 7: Error Rates of FTRL-ADP, TDAP, PA I, PA II, ALMA, FTRL-Proximal, CW, and DDM-enabled Hoeffding Tree on 6 Real-world Datasets.

Polikar, 2013), Electricity (Harries, 1999), and Intrusion (Lippmann et al., 2000). These datasets cover a wide range of applications showing the prevalence of concept drift in real-world applications.

Spam dataset. This dataset contains spam emails collected using the SpamAssassin data collection framework. The bag of words model is used to represent each spam as a vector. Concept drift contained in the dataset has been studied in detail in (Katakis et al., 2010).

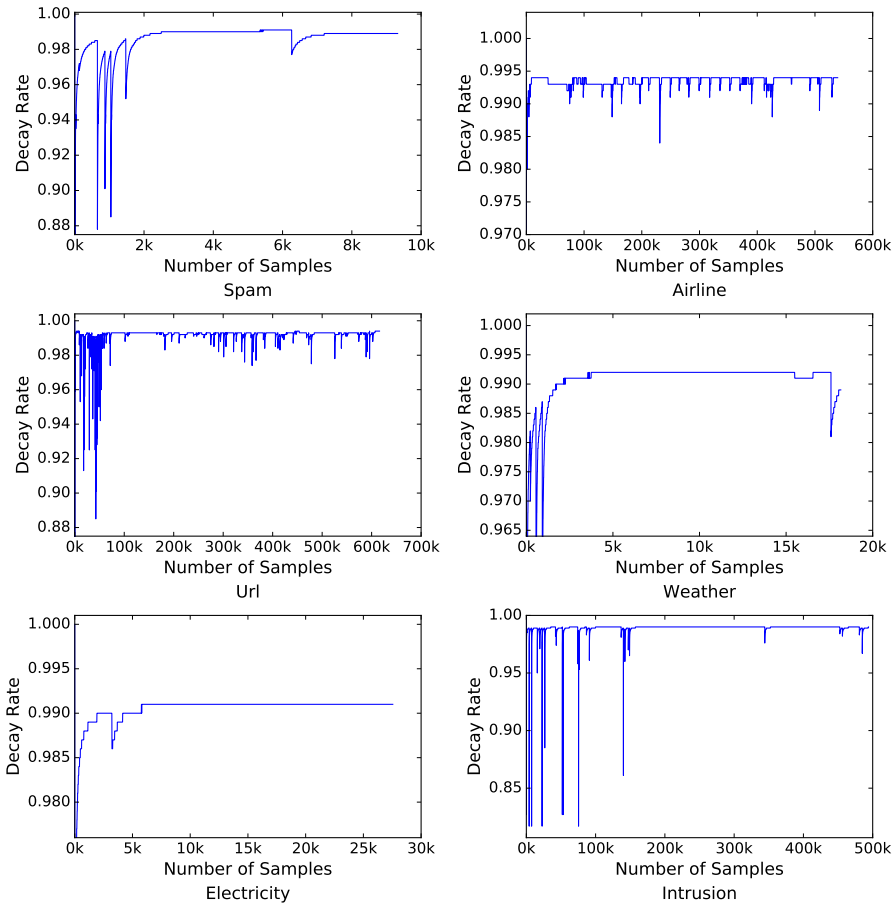

Figure 8: Evolution of Decaying Rates of FTRL-ADP on 6 Real-world Datasets.

URL dataset. This dataset is collected in an effort aimed to create a machine learning based firewall to replace the commonly used blacklists. The dataset contains anonymized and labeled URLs collected over the period of 120 days. Due to the adversary nature of the domain of computer security, concept drifts are anticipated in this dataset and have been studied in detail in (Ma et al., 2009).

Airline dataset. This dataset is motivated by the need to predict flight delays. It contains more than half a million examples collected over the period of more than 20 years. The raw dataset contains 4 categorical features (Airline, AirportFrom, AirportTo, and DayOfWeek) and 2 numerical features (Time and Length). We

Dataset	FTRL-ADP	FTRL-Proximal	TDAP	PAI	PAII	CW	ALMA	DDM-HT
Spam	13.24	19.28	14.71	**11.06**	11.09	11.5	11.79	20.28
Airline	36.3	38.78	37.3	40.73	40.7	40.87	40.84	**35.1**
Url	**33.91**	35.84	34.37	36.94	36.97	37.56	37.56	35.74
Weather	28.32	30.13	**26.0**	29.08	28.97	26.25	27.24	26.71
Electricity	**15.23**	23.16	21.63	15.82	15.76	15.71	15.84	15.88
Intrusion	**0.15**	1.70	0.48	2.11	2.14	1.77	1.82	0.80

Table 3: Error Rates of FTRL-ADP, TDAP, PA I, PA II, ALMA, FTRL-Proximal, CW, and DDM-enabled Hoeffding Tree on 6 Real-world Datasets.

use one hot encoding to convert categorical features to numerical features for the evaluation.

Weather dataset. This dataset is motivated by the need to predict rain in weather forecasting. It is a collection of weather records over more than 50 years curated by the National Oceanic and Atmospheric Administration, USA.

Electricity dataset. This dataset records the changes in electricity consumption in New South Wales (Australia). The duration of the dataset is from 7 May 1996 to 5 December 1998. A record contains 6 attributes: day of week, time of day, demand in New South Wales, price in Victoria, demand in Victoria, and electricity transfer between the states. The label is either UP for the increase in the consumption of electricity or DOWN vice versa.

Intrusion dataset. This dataset records the network connections made at the gateway of a simulated military network environment. The network traffic contains various type of attacks in the midst of normal traffic. The dataset contains 41 features including both categorical features, such as the protocol type, and numerical features, such as the number of failed logins. The label of a connection is either attack or normal.

5.2.2. Discussion

Like the synthetic experiment, the cumulative prediction errors are used as the performance metric. Figure 7 shows the error rates associated with the best parameters incurred by all algorithms on all datasets. Table 3 shows the final

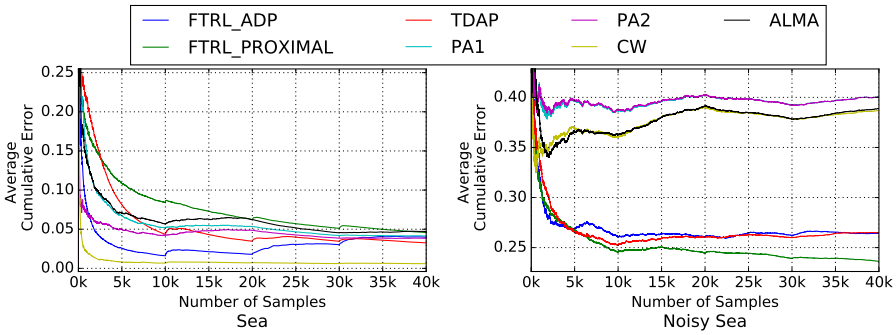

Figure 9: Error Rates of FTRL-ADP, TDAP, PA I, PA II, ALMA, FTRL-Proximal and CW on the two SEA Datasets.

error rates. The bold numbers in Table 3 indicate the best performers. Figure 8 shows the evolution of the decaying rate reported by FTRL-ADP.

From Figure 7, we can see that FTRL-ADP and TDAP tend to be more robust to concept drifts than FTRL-Proximal. For example, in the case of the Spam dataset, the prediction error of the FTRL-Proximal consistently rises after the onset of the drift (occurs at around example 1000^{th} based on the evolution of decaying rate in Figure 8). The error rates of FTRL-ADP and TDAP also increase but plateau quickly. We can notice the same observation in the Airline dataset and the Intrusion dataset. The performance of FTRL-Proximal significantly increases from example $200k^{\text{th}}$ (Airline dataset) and $10k^{\text{th}}$ (Intrusion dataset) while the performance of TDAP and FTRL-ADP also increase but at a much lower rate.

TDAP and FTRL-ADP perform comparably well in real-world datasets. This is the result of fine tuning the decaying rate of TDAP by running the experiment multiple times to select the best decaying rate. On the contrary, the decaying rate of FTRL-ADP is adaptive and doesn't require tuning.

The baseline classifier DDM-HT performs the worst in the Spam dataset. However, it is the best classifier in the Airline dataset. In comparison to FTRL-ADP, DDM-HT outperforms FTRL-ADP in the Airline and the Weather datasets. However, FTRL-ADP outperforms DDM-HT in the rest 4 datasets.

PA I, PA II, CW, and ALMA outperform other algorithms in the Spam

dataset. They are the most adaptive to concept drift in this case with better performance after the drift. However, these 4 learners are often over-sensitive to noise; this characteristic may explain their poor performance in the other 3 datasets. We can demonstrate this characteristic in a simulated scenario. For example, Figure 9 shows the cumulative error rates of all algorithms on the two SEA datasets. We can see that CW performs almost perfectly (99.35% accuracy) and significantly outperforms other algorithms in the clean SEA dataset. However, its performance degrades substantially when the noise is injected.

6. Conclusion and Future Work

In this paper, we have proposed an adaptive online algorithm to address the problem of concept drift, called FTRL-ADP. We have introduced two novelties in the adaptive mechanism of FTRL-ADP. First, we derive a formula to adaptively adjust the decaying rate based on sublinear regret analysis. Second, we use the concept drift detector DDM to speed up the adaptation to drifting concepts. The adaptive mechanism eliminates the need for manually tuning the decaying rate required by TDAP. Experiments with 14 datasets and 6 other online algorithms show that FTRL-ADP is most advantageous in noisy environments with real concept drifts.

We outline two directions for the future work. In the first direction, we plan to investigate other possibilities for the adaptive decaying mechanism. For example, we are aware of a rich literature on adaptive forgetting factor for the original non-adaptive Reclusive Least Squared (RLS) algorithm. The forgetting factor in RLS plays the same role as the decaying rate in FTRL-ADP. In the second direction, we aim to develop a mathematical analysis to prove the convergence property of FTRL-ADP in the case of concept drift. We are now only able to visually demonstrate the tracking ability of the algorithm, which is a good starting point for a rigorous analysis.

Acknowledgment

This work was supported by the Research Scholarship generously provided by the Nanyang Technological University and the Singapore Government.

References

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., & Morales-Bueno, R. (2006). Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams* (pp. 77–86). volume 6.

Barros, R. S. M., Cabral, D. R., Gonçalves Jr, P. M., & Santos, S. G. T. C. (2017). RDDM: Reactive drift detection method. *Expert Systems with Applications*, *90*, 344–355.

Barros, R. S. M., Santos, S. G. T. C., & Gonçalves Jr, P. M. (2016). A boosting-like online learning ensemble. In *2016 IEEE International Joint Conference on Neural Networks (IJCNN)* (pp. 1871–1878).

Bekerman, D., Shapira, B., Rokach, L., & Bar, A. (2015). Unknown malware detection using network traffic classification. In *2015 IEEE Conference on Communications and Network Security (CNS)* (pp. 134–142).

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). Moa: Massive online analysis. *Journal of Machine Learning Research*, *11*, 1601–1604.

Boyd, J. P. (2014). *Solving Transcendental Equations: The Chebyshev Polynomial Proxy and Other Numerical Rootfinders, Perturbation Series, and Oracles*. SIAM.

Boyd, S. P., & Vandenberghe, L. (2010). *Follow-the-Regularized-Leader and Mirror Descent*. Cambridge University Press.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, *7*, 551–585.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, *7*(Jan), 1–30.

Ditzler, G., & Polikar, R. (2011). Hellinger distance based drift detection for nonstationary environments. In *2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)* (pp. 41–48).

Ditzler, G., & Polikar, R. (2013). Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *25*(10), 2283–2301.

Ditzler, G., Roveri, M., Alippi, C., & Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, *10*(4), 12–25.

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 71–80). ACM.

Dredze, M., Crammer, K., & Pereira, F. (2008). Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning* (pp. 264–271). ACM.

Fan, W. (2004). Systematic data selection to mine concept-drifting data streams. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* KDD '04 (pp. 128–137). ACM.

Frías-Blanco, I., Campo-Ávila, J. d., Ramos-Jiménez, G., Morales-Bueno, R., Ortiz-Díaz, A., & Caballero-Mota, Y. (2015). Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, *27*(3), 810–823.

Frías-Blanco, I., Verdecia-Cabrera, A., Ortiz-Díaz, A., & Carvalho, A. (2016). Fast adaptive stacking of ensembles. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing* (pp. 929–934).

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence* (pp. 286–295). Springer International Publishing.

Gama, J., Sebastião, R., & Rodrigues, P. P. (2013). On evaluating stream learning algorithms. *Machine Learning*, *90*(3), 317–346.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, *46*(4), 44.

Gentile, C. (2001). A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, *2*, 213–242.

Gonçalves Jr, P. M., Santos, S. G. T. C., Barros, R. S. M., & Vieira, D. C. (2014). A comparative study on concept drift detectors. *Expert Systems with Applications*, *41*(18), 8144–8156.

Grossberg, S. (2013). Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural Networks*, *37*, 1–47.

Harries, M. (1999). Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of New South Wales.

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 97–106). ACM.

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2010). Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, *22*(3), 371–391.

Lee, J., & Magoulès, F. (2012). Detection of concept drift for learning from stream data. In *2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems* (pp. 241–245).

Lim, J.-s., Lee, S., & Pang, H.-S. (2013). Low complexity adaptive forgetting factor for online sequential extreme learning machine (OS-ELM) for application to nonstationary system estimations. *Neural Computing and Applications*, *22*(3), 569–576.

Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., Weber, D., Webster, S. E., Wyschogrod, D., Cunningham, R. K. et al. (2000). Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings* (pp. 12–26). IEEE volume 2.

Liu, D., Wu, Y., & Jiang, H. (2016). Fp-elm: An online sequential learning algorithm for dealing with concept drift. *Neurocomputing*, *207*, 322–334.

Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009). Identifying suspicious URLs: an application of large-scale online learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 681–688). ACM.

Masud, M., Gao, J., Khan, L., Han, J., & Thuraisingham, B. M. (2011). Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, *23*(6), 859–874.

McMahan, H. B. (2014). A survey of algorithms and analysis for adaptive online learning. *arXiv preprint arXiv:1403.3465*, .

McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D., & others (2013). Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1222–1230). ACM.

Minku, L. L., & Yao, X. (2012). DDD: A New Ensemble Approach for Dealing with Concept Drift. *IEEE Transactions on Knowledge and Data Engineering*, *24*(4), 619–633.

Nishida, K., & Yamauchi, K. (2007). Detecting concept drift using statistical testing. In *International conference on discovery science* (pp. 264–269). Springer International Publishing.

Oza, N. C., & Russell, S. (2001). *Online ensemble learning*. University of California, Berkeley.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, *41*(1), 100–115.

Pavlidis, N., Tasoulis, D., Adams, N., & Hand, D. (2011). Lambda-perception - an adaptive classifier for data streams. *Pattern Recognition*, *44*(1), 78–96.

Pears, R., Sakthithasan, S., & Koh, Y. S. (2014). Detecting concept change in dynamic data streams: A sequential approach based on reservoir sampling. *Machine Learning*, *97*(3), 259–293.

Pesaranghader, A., & Viktor, H. L. (2016). Fast hoeffding drift detection method for evolving data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 96–111). Springer.

Pocock, A., Yiapanis, P., Singer, J., Luján, M., & Brown, G. (2010). Online non-stationary boosting. In *International Workshop on Multiple Classifier Systems* (pp. 205–214). Springer International Publishing.

Pozzolo, A. D., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2015). Credit card fraud detection and concept-drift adaptation with delayed supervised information. In *2015 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8).

Sakthithasan, S., Pears, R., & Koh, Y. S. (2013). One pass concept change detection for data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 461–472). Springer.

Santos, S. G. T. C., Gonçalves Jr, P. M., Santos Silva, G. D., & Barros, R. S. M. (2014). Speeding up recovery from concept drifts. In *Joint European*

Conference on Machine Learning and Knowledge Discovery in Databases (pp. 179–194). Springer.

Saxe, J., & Berlin, K. (2015). Deep neural network based malware detection using two dimensional binary program features. In *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)* (pp. 11–20).

Sethi, T. S., Kantardzic, M., & Hu, H. (2016). A grid density based framework for classifying streaming data in the presence of concept drift. *Journal of Intelligent Information Systems*, *46*(1), 179–211.

Shalev-Shwartz, S. (2011). Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, *4*(2), 107–194.

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

Street, W. N., & Kim, Y. (2001). A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 377–382). ACM.

Tan, Y., Fan, Z., Li, G., Wang, F., Li, Z., Liu, S., Pan, Q., Xing, E. P., & Ho, Q. (2016). Scalable time-decaying adaptive prediction algorithm. In *KDD* (pp. 617–626).

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., & Mahoney, P. (2006). Stanley: The robot that won the DARPA grand challenge. *Journal of Field Robotics*, *23*(9), 661–692.

Wang, J., Zhao, P., & Hoi, S. C. (2012). Exact soft confidence-weighted learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning* (pp. 107–114). Omnipress.

Wickham, H. (2011). Asa 2009 data expo. *Journal of Computational and Graphical Statistics*, *20*(2), 281–283.

Xiao, L. (2010). Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, *11*, 2543–2596.

Appendix A. Proof of Theorem 1

PROOF. To remind the optimization objective of FTRL-DP:

$$w_{t+1} = \operatorname*{argmin}_w g_{1:t}^\top w + \lambda_1 \|w\|_1 + \frac{1}{2}\lambda_2 \|w\|_2^2 + \frac{1}{2}\lambda_p \sum_{s=1}^{t} \gamma^{t-s} \|w - w_s\|_2^2$$

$$w_{t+1} = \operatorname*{argmin}_w \big(g_{1:t}^\top - \lambda_p \sum_{s=1}^{t} \gamma^{t-s} w_s^\top\big) w + \lambda_1 \|w\|_1 + \frac{1}{2}\big(\lambda_2 + \lambda_p \frac{1-\gamma^t}{1-\gamma}\big)\|w\|_2^2 + \frac{1}{2}\lambda_p \sum_{s=1}^{t} \gamma^{t-s} \|w_s\|_2^2$$

Omitting the constant term $\frac{1}{2}\lambda_p \sum_{s=1}^{t} \gamma^{t-s} \|w_s\|_2^2$, we have:

$$w_{t+1} = \operatorname*{argmin}_w z_t^\top w + \lambda_1 \|w\|_1 + \frac{1}{2}\big(\lambda_2 + \lambda_p r_t\big)\|w\|_2^2 \tag{A.1}$$

In Equation A.1, $z_t = g_{1:t}^\top - \lambda_p \sum_{s=1}^{t} \gamma^{t-s} w_s^\top$ and $r_t = \frac{1-\gamma^t}{1-\gamma}$. Each component of w contribute independently to the objective function of A.1 hence can be solve separately:

$$w_{t+1,i} = \operatorname*{argmin}_{w_i} z_{t,i} w_i + \lambda_1 \|w_i\|_1 + \frac{1}{2}\big(\lambda_2 + \lambda_p r_t\big)\|w_i\|_2^2 \tag{A.2}$$

Note that w_i in A.2 refers to the i^{th} component of w. Let $f(w_i) = z_{t,i} w_i + \lambda_1 \|w_i\|_1 + \frac{1}{2}\big(\lambda_2 + \lambda_p r_t\big)\|w_i\|_2^2$. There are two cases:

- If $\|z_{t,i}\|_1 \leq \lambda_1$, we have:

$$f(w_i) \geq -\|z_{t,i} w_i\|_1 + \lambda_1 \|w_i\|_1 + \frac{1}{2}\big(\lambda_2 + \lambda_p r_t\big)\|w_i\|_2^2$$

$$f(w_i) \geq -\lambda_1\|w_i\|_1 + \lambda_1\|w_i\|_1 + \frac{1}{2}\big(\lambda_2 + \lambda_p r_t\big)\|w_i\|_2^2 = \frac{1}{2}\big(\lambda_2 + \lambda_p r_t\big)\|w_i\|_2^2 \geq 0$$

$f(w_i)$ achieves the minimum at $w_i = 0$

- If $\|z_{t,i}\|_1 \geq \lambda_1$, $z_{t,i}$ and w_i must have opposite signs at the minimum of $f(w_i)$ as otherwise w_i can always have sign flipped to further reduce $f_i(w_i)$. Therefore, it is equivalent to solving:

$$w_{t+1,i} = \operatorname*{argmin}_{w_i} z_{t,i}w_i - \operatorname{sign}(z_{t,i})\lambda_1\|w_i\|_1 + \frac{1}{2}\big(\lambda_2 + \lambda_p r_t\big)\|w_i\|_2^2$$

  which achieves minimum at zero gradient or $w_i = -\frac{z_{t,i} - \operatorname{sign}(z_{t,i})\lambda_1}{\lambda_2 + \lambda_p r_t}$

  This concludes the proof.

Appendix B. Proof of Theorem 2

PROOF. As $r_{1:t}(w) = \frac{1}{2}\sum_{s=1}^{t}\sigma_s\|w - w_s\|_2^2$ is 1-strongly convex (Lemma 3) w.r.t. norm $\|w\|_{(t)} = \sqrt{\sigma_{1:t}}\|w\|_2$, whose dual is $\|w\|_{(t),*} = \frac{1}{\sqrt{\sigma_{1:t}}}\|w\|_2$, applying Theorem 2 in (McMahan, 2014), we have:

$$Regret(w^*) \leq \frac{1}{2\eta_T}(2R)^2 + \frac{1}{2}\sum_{t=1}^{T}\eta_t\|g_t\|^2$$

In which, $\eta_t = \frac{1}{\sigma_{1:t}} = \frac{1}{\sum_{s=1}^{t}\gamma^{T-s}} = \frac{(1-\gamma)\gamma^t}{\gamma^T(1-\gamma^t)}$, therefore:

$$Regret(w^*) \leq 2R^2\frac{1-\gamma^T}{1-\gamma} + \frac{G^2}{2}\frac{1}{\gamma^T}\sum_{t=1}^{T}\frac{(1-\gamma)\gamma^t}{(1-\gamma^t)}$$

If we choose $\gamma = 1 - \frac{\ln T}{2T}$, we have:

$$Regret(w^*) \leq 4R^2\frac{T}{\ln T} + \frac{G^2}{2}\frac{1}{\gamma^T}\sum_{t=1}^{T}\frac{\gamma^t}{\sum_{i=0}^{t-1}\gamma^i}$$

As $\frac{\gamma^t}{\sum_{i=0}^{t-1}\gamma^i} \leq \frac{1}{t}$ for $\gamma < 1$, we have:

$$Regret(w^*) \leq 4R^2\frac{T}{\ln T} + \frac{G^2}{2}\frac{1}{\gamma^T}\sum_{t=1}^{T}\frac{1}{t}$$

As $\sum_{t=1}^{T}\frac{1}{t} \leq 1 + \ln T$ (Lemma 4), we have:

$$Regret(w^*) \leq 4R^2\frac{T}{\ln T} + \frac{G^2}{2}\frac{(1+\ln T)}{(1-\frac{\ln T}{2T})^T}$$

Therefore:

$$\lim_{T\to\infty}\frac{Regret(w^*)}{T} \leq \lim_{T\to\infty} 4R^2\frac{1}{\ln T} + \frac{G^2}{2}\frac{(1+\ln T)}{\sqrt{T}}\frac{1}{\sqrt{T}(1-\frac{\ln T}{2T})^T} \tag{B.1}$$

As $\lim_{T\to\infty}\sqrt{T}(1-\frac{\ln T}{2T})^T = 1$ (Lemma 5), the right hand side of B.1 vanishes when T goes to infinity. Therefore:

$$\lim_{T\to\infty}\frac{Regret(w^*)}{T} \leq 0$$

This concludes the proof.

Lemma 3. *Prove that $r_{1:t}(w) = \frac{1}{2}\sum_{s=1}^{t}\sigma_s\|w-w_s\|_2^2$ is 1-strongly convex w.r.t. norm $\|w\|_{(t)} = \sqrt{\sigma_{1:t}}\|w\|_2$, given $\sigma_1, \sigma_2, ..., \sigma_t > 0$ and $w_1, w_2, ..., w_t$.*

Proof. With arbitrary $x, y \in \mathbb{R}^n$, we have:

$$\begin{aligned}
& r_{1:t}(y) \geq r_{1:t}(x) + \nabla r_{1:t}(x)(y-x) + \frac{1}{2}\|y-x\|_{(t)}^2 \\
\Leftrightarrow \quad & r_{1:t}(y) - r_{1:t}(x) \geq \nabla r_{1:t}(x)(y-x) + \frac{1}{2}\|y-x\|_{(t)}^2 \\
\Leftrightarrow \quad & \frac{1}{2}\sum_{s=1}^{t}\sigma_s(y+x-2w_s)^\top(y-x) \geq \sum_{s=1}^{t}\sigma_s(x-w_s)(y-x) + \frac{1}{2}\|y-x\|_{(t)}^2 \\
\Leftrightarrow \quad & \frac{1}{2}\sum_{s=1}^{t}\sigma_s(y-x)^\top(y-x) \geq \frac{1}{2}\sigma_{1:t}\|y-x\|_2^2
\end{aligned}$$

The last inequality is actually an equality. This concludes the proof.

Lemma 4. *Prove that:*

$$\sum_{t=1}^{T} \frac{1}{t} \leq 1 + \ln T \tag{B.2}$$

Proof. We have:

$$\ln T = \int_{x=1}^{T} \frac{\mathrm{d}x}{x} = \sum_{k=1}^{T-1} \int_{x=k}^{k+1} \frac{\mathrm{d}x}{x} \geq \sum_{k=1}^{T-1} \int_{x=k}^{k+1} \frac{\mathrm{d}x}{k+1} = \sum_{k=1}^{T-1} \frac{1}{k+1}$$

Therefore:

$$1 + \ln T \geq \sum_{t=1}^{T} \frac{1}{t}$$

This concludes the proof.

Lemma 5. *Prove that:*

$$\lim_{T\to\infty} \sqrt{T}\big(1 - \frac{\ln T}{2T}\big)^T = 1 \tag{B.3}$$

Proof. Change T to $\frac{1}{x}$, and take the log of the left-hand side of B.3, we have:

$$\lim_{T\to\infty} \ln\left[\sqrt{T}\big(1 - \frac{\ln T}{2T}\big)^T\right] = \lim_{x\to 0} \frac{\ln\left(1 + \frac{x\ln x}{2}\right) - \frac{1}{2}x\ln x}{x}$$

Applying L'Hôpital's rule, we have:

$$\lim_{T\to\infty} \ln\left[\sqrt{T}\big(1 - \frac{\ln T}{2T}\big)^T\right] = \lim_{x\to 0} -\frac{x\ln x + x(\ln x)^2}{2(2 + x\ln x)}$$

As $\lim_{x\to 0} x(\ln x)^n = 0$ for all integer n, we have:

$$\lim_{x\to 0} -\frac{x\ln x + x(\ln x)^2}{2(2 + x\ln x)} = 0$$

Therefore:

$$\lim_{T\to\infty} \sqrt{T}\big(1 - \frac{\ln T}{2T}\big)^T = 1$$

This concludes the proof.