Learning from Time-Changing Data with Adaptive Windowing *

Albert Bifet Ricard Gavaldà
Universitat Politècnica de Catalunya
{abifet,gavalda}@lsi.upc.edu

Abstract

We present a new approach for dealing with distribution change and concept drift when learning from data sequences that may vary with time. We use sliding windows whose size, instead of being fixed *a priori*, is recomputed online according to the rate of change observed from the data in the window itself. This delivers the user or programmer from having to guess a time-scale for change. Contrary to many related works, we provide rigorous guarantees of performance, as bounds on the rates of false positives and false negatives.

Using ideas from data stream algorithmics, we develop a time- and memory-efficient version of this algorithm, called `ADWIN2`. We show how to combine `ADWIN2` with the Naïve Bayes (NB) predictor, in two ways: one, using it to monitor the error rate of the current model and declare when revision is necessary and, two, putting it inside the NB predictor to maintain up-to-date estimations of conditional probabilities in the data. We test our approach using synthetic and real data streams and compare them to both fixed-size and variable-size window strategies with good results.

Keywords: Data Streams, Time-Changing Data, Concept and Distribution Drift, Naïve Bayes

1 Introduction

Dealing with data whose nature changes over time is one of the core problems in data mining and machine learning. To mine or learn such data, one needs strategies for the following three tasks, at least: 1) detecting when change occurs 2) deciding which examples to keep and which ones to forget (or, more in general, keeping updated sufficient statistics), and 3) revising the current model(s) when significant change has been detected.

Most strategies use variations of the *sliding window* idea: a window is maintained that keeps the most recently read examples, and from which older examples are dropped according to some set of rules. For this three tasks, the content of the window can be used.

In this paper, we present a new algorithm (`ADWIN`, for ADaptive WINdowing) for maintaining a window of variable size containing bits or real numbers. The algorithm automatically grows the window when no change is apparent, and shrinks it when data changes. Unlike many related works, we provide rigorous guarantees of its performance, in the form of bounds on the rates of false positives and false negatives. In fact, it is possible to show that for some change structures, `ADWIN` automatically adjusts its window size to the optimum balance point between reaction time and small variance. Since `ADWIN` keeps bits or real numbers, it can be put to work together with a learning algorithm in the first way, that is, to monitor the error rate of the current model.

The first version of `ADWIN` is inefficient in time and memory. Using ideas from data-stream algorithmics, we provide another version, `ADWIN2`, working in low memory and time. In particular, `ADWIN2` keeps a window of length W with $O(\log W)$ memory and update time, while keeping essentially the same performance guarantees as `ADWIN` (in fact, it does slightly better in experiments). Because of this low time and memory requirements, it is thus possible to use `ADWIN2` in the second way: a learning algorithm can create many instances of `ADWIN2` to maintain updated the statistics (counts, averages, entropies, ...) from which it builds the model.

We compare `ADWIN2` with a number of fixed-size windows and show, as expected, that it performs about as well or only slightly worse than the best window for each rate of change, and performs far better than each windows of any fixed-size W when the change of rate is very different from W.

NOTE: Several discussions, technical details, comparison to related work, and results of experiments can be found in an extended version, available from the authors' homepages.

2 Maintaining Updated Windows of Varying Length

In this section we describe our algorithms for dynamically adjusting the length of a data window, make a formal claim about its performance, and derive an efficient variation.

*Partially supported by the 6th Framework Program of EU through the integrated project DELIS (#001907), by the EU PASCAL Network of Excellence, IST-2002-506778, and by the DGICYT MOISES-BAR project, TIN2005-08832-C03-03. Home pages: `http://www.lsi.upc.edu/~{abifet, gavalda}`

2.1 Setting The inputs to the algorithms are a confidence value $\delta \in (0,1)$ and a (possibly infinite) sequence of real values $x_1, x_2, x_3, \ldots, x_t, \ldots$ The value of x_t is available only at time t. Each x_t is generated according to some distribution D_t, independently for every t. We denote with μ_t the expected value when it is drawn according to D_t. We assume that x_t is always in $[0,1]$; by an easy rescaling, we can handle any case in which we know an interval $[a,b]$ such that $a \le x_t \le b$. Nothing else is known about the sequence of distributions D_t; in particular, μ_t is unknown for all t.

2.2 First algorithm ADWIN keeps a sliding window W with the most recently read x_i. Let n denote the length of W, $\hat{\mu}_W$ the (observed) average of the elements in W, and μ_W the (unknown) average of μ_t for $t \in W$. Strictly speaking, these quantities should be indexed by t, but in general t will be clear from the context.

Algorithm ADWIN is presented in Figure 1. The idea is simple: whenever two "large enough" subwindows of W exhibit "distinct enough" averages, one can conclude that the corresponding expected values are different, and the older portion of the window is dropped.

The value of ϵ_{cut} for a partition $W_0 \cdot W_1$ of W is computed as follows: Let n_0 and n_1 be the lengths of W_0 and W_1 and n be the length of W, so $n = n_0 + n_1$. Let $\hat{\mu}_{W_0}$ and $\hat{\mu}_{W_1}$ be the averages of the values in W_0 and W_1, and μ_{W_0} and μ_{W_1} their expected values. To obtain totally rigorous performance guarantees we define:

$$m = \frac{1}{1/n_0 + 1/n_1} \text{ (harmonic mean of } n_0 \text{ and } n_1),$$

$$\delta' = \frac{\delta}{n}, \text{ and } \quad \epsilon_{cut} = \sqrt{\frac{1}{2m} \cdot \ln \frac{4}{\delta'}}\ .$$

Our statistical test for different distributions in W_0 and W_1 simply checks whether the observed average in both subwindows differs by more than the threshold ϵ_{cut}. The role of δ' is to avoid problems with multiple hypothesis testing (since we will be testing n different possibilities for W_0 and W_1 and we want global error below δ). Later we will provide a more sensitive test based on the normal approximation that, although not 100% rigorous, is perfectly valid in practice.

Now we state our main technical result about the performance of ADWIN:

THEOREM 2.1. *At every time step we have*

1. (False positive rate bound). *If μ_t remains constant within W, the probability that* ADWIN *shrinks the window at this step is at most δ.*
2. (False negative rate bound). *Suppose that for* some *partition of W in two parts W_0W_1 (where W_1 contains the most recent items) we have $|\mu_{W_0} - \mu_{W_1}| > 2\epsilon_{cut}$. Then with probability $1-\delta$* ADWIN *shrinks W to W_1, or shorter.*

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
1  Initialize Window W
2  for each t > 0
3      do W ← W ∪ {x_t} (i.e., add x_t to the head of W)
4         repeat Drop elements from the tail of W
5           until |μ̂_{W_0} − μ̂_{W_1}| ≥ ε_cut holds
6              for every split of W into W = W_0 · W_1
7         output μ̂_W
```

Figure 1: Algorithm ADWIN.

In practice, the definition of ϵ_{cut} as above is too conservative. Indeed, it is based on the Hoeffding bound, which is valid for all distributions but greatly overestimates the probability of large deviations for distributions of small variance; in fact, it is equivalent to assuming always the worst-case variance $\sigma^2 = 1/4$. In practice, one can observe that $\mu_{W_0} - \mu_{W_1}$ tends to a normal distribution for large window sizes, and use

$$\epsilon_{cut} = \sqrt{\frac{2}{m} \cdot \sigma_W^2 \cdot \ln \frac{2}{\delta'}} \ + \ \frac{2}{3\,m} \ln \frac{2}{\delta'}, \tag{2.1}$$

where σ_W^2 is the observed variance of the elements in window W. Thus, the term with the square root is essentially equivalent to setting ϵ_{cut} to k times the standard deviation, for k depending on the desired confidence δ, as is done in [4]. Setting $\delta' = \delta/(\ln n)$ is enough in this context to protect from the multiple hypothesis testing problem.

Let us consider how ADWIN behaves in two special cases: sudden (but infrequent) changes, and slow gradual changes. Suppose that for a long time μ_t has remained fixed at a value μ, and that it suddenly jumps to a value $\mu' = \mu + \epsilon$. By part (2) of Theorem 2.1 and Equation 2.1, one can derive that the window will start shrinking after $O(\mu \ln(1/\delta)/\epsilon^2)$ steps, and in fact will be shrunk to the point where only $O(\mu \ln(1/\delta)/\epsilon^2)$ examples prior to the change are left. From then on, if no further changes occur, no more examples will be dropped so the window will expand unboundedly.

In case of a gradual change with slope α following a long stationary period at μ, the average of W_1 after n_1 steps is $\mu + \alpha n_1/2$; we have $\epsilon (= \alpha n_1/2) \ge O(\sqrt{\mu \ln(1/\delta)/n_1})$ iff $n_1 = O(\mu \ln(1/\delta)/\alpha^2)^{1/3}$. So n_1 steps after the change the window will start shrinking, and will remain at approximately size n_1 from then on. A dependence on α of the form $O(\alpha^{-2/3})$ may seem

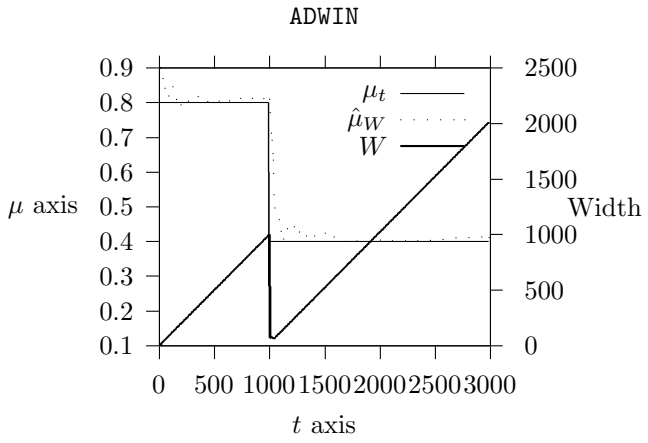

Figure 2: Output of algorithm ADWIN with abrupt change.

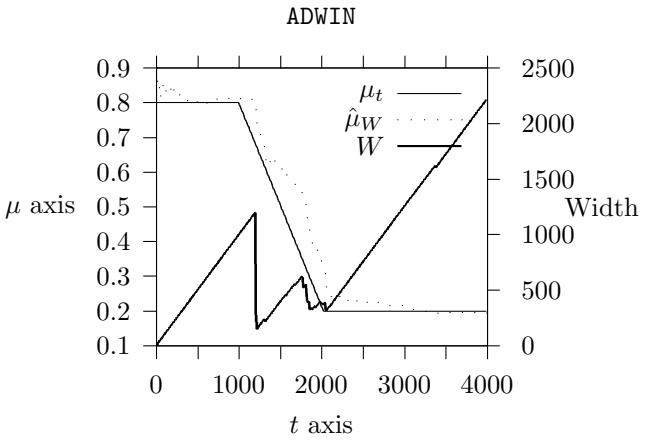

Figure 3: Output of algorithm ADWIN with slow gradual changes.

odd at first, but one can show that this window length is actually optimal in this setting: it minimizes the sum of variance error (due to short window) and error due to out-of-date data (due to long windows in the presence of change). Thus, in this setting, ADWIN provably adjusts automatically the window setting to its optimal value, up to multiplicative constants.

Figures 2 and 3 illustrate these behaviors. In Figure 2, a sudden change from $\mu_{t-1} = 0.8$ to $\mu_t = 0.4$ occurs, at $t = 1000$. In Figure 3, μ_t gradually descends from 0.8 to 0.2 in the range $t \in [1000..2000]$. In this case, ADWIN cuts the window sharply at t around 1200, keeps the window length bounded (with some random fluctuations) while the slope lasts, and starts growing it linearly again after that.

2.3 Improving time and memory requirements

Our first version of ADWIN is computationally expensive, because it checks exhaustively all "large enough" subwindows of the current window for possible cuts. Furthermore, the contents of the window is kept explicitly, with the corresponding memory cost as the window grows. To reduce these costs we present a new version ADWIN2 using ideas developed in data stream algorithmics [1, 7, 2, 3] to find a good cutpoint quickly. We next provide a sketch of how these data structures work.

Our data structure is a variation of exponential histograms [3], a data structure that maintains an approximation of the number of 1's in a sliding window of length W with logarithmic memory and update time. We adapt this data structure in a way that can provide this approximation simultaneously for about $O(\log W)$ subwindows whose lengths follow a geometric law, *with no memory overhead* with respect to keeping the count for a single window. That is, our data structure will be able to give the number of 1s among the most recently $t-1$, $t-\lfloor c \rfloor$, $t-\lfloor c^2 \rfloor$,..., $t-\lfloor c^i \rfloor$, ... read bits, with the same amount of memory required to keep an approximation for the whole W. Note that keeping exact counts for a fixed-window size is provably impossible in sublinear memory. We go around this problem by shrinking or enlarging the window strategically so that what would otherwise be an approximate count happens to be exact.

More precisely, to design the algorithm one chooses a parameter M, which controls both 1) the amount of memory used (it will be $O(M \log W/M)$ words, and 2) the closeness of the cutpoints checked (the basis c of the geometric series above will be about $c = 1+1/M$). Note that the choice of M does *not* reflect any assumption about the time-scale of change: Since points are checked at a geometric rate anyway, this policy is essentially scale-independent.

We summarize these main results with the following theorem.

THEOREM 2.2. *The* ADWIN2 *algorithm maintains a data structure with the following properties:*

- *It uses* $O(M \cdot \log(W/M))$ *memory words (assuming a memory word can contain numbers up to* W*).*
- *It can process the arrival of a new element in* $O(1)$ *amortized time and* $O(\log W)$ *worst-case time.*
- *It can provide the exact counts of* 1*'s for all the subwindows whose lengths are of the form* $\lfloor (1 + 1/M)^i \rfloor$*, in* $O(1)$ *time per query.*

Since ADWIN2 tries $O(\log W)$ cutpoints, the total processing time per example is $O(\log W)$ (amortized) and $O(\log^2 W)$ (worst-case). In the case of real values, we maintain buckets of two elements: *capacity* and *content*. We store at *content* the sum of the real numbers we want to summarize. We restrict *capacity* to be a power of two. We use $O(\log W)$ buckets, and check $O(\log W)$ possible cuts. The memory requirement for each bucket is $\log W + R + \log\log W$ bits per bucket, where R is number of bits used to store a real number. The difference in approximation power between ADWIN and ADWIN2 is almost negligible, so we use ADWIN2 exclusively for our experiments.

3 Experimental Validation of ADWIN2

We construct the following experiments to test the performance of our algorithms. We use, somewhat arbitrarily, $M = 5$ for all experiments.

In a first experiment, we investigate the rate of false positives of ADWIN2. This is a very important measure, specially when there is a cost associated with a reported change. To do this, we feed ADWIN2 a data stream of 100,000 bits, generated from a stationary Bernoulli distribution with parameter μ, and different confidence parameters δ.

Table 1 shows the ratio of false positives obtained. In all cases, it is below δ as predicted by the theory, and in fact much smaller for small values of μ.

Table 1: Rate of false positives

μ	$\delta = 0.05$	$\delta = 0.1$	$\delta = 0.3$
0.01	0.0000	0.0000	0.0000
0.1	0.0001	0.0002	0.0018
0.3	0.0008	0.0017	0.0100
0.5	0.0012	0.0030	0.0128

In a second set of experiments, we want to compare ADWIN2 as an estimator with estimations obtained from fixed-size window, and with fixed-size window which are flushed when change is detected. In the last case, we use a pair of windows (X, Y) of a fixed size W. Window X is used as a reference window that contains the first W elements of the stream that occurred after the last detected change. Window Y is a sliding window that contains the latest W items in the data stream. To detect change we check whether the difference of the averages of the two windows exceeds threshold ϵ_{cut}. If it does, we copy the content of window Y into reference window X, and empty the sliding window Y. This scheme is as in [6], and we refer to it as "fixed-size windows with flushing".

We build a framework with a stream of synthetic data, and estimators of each class: an estimator that uses ADWIN2, an array of estimators of fixed-size windows for different sizes, and also an array of fixed-size windows with flushing. Our synthetic data streams consist of some triangular wavelets, of different periods, some square wavelets, also of different periods, and a staircase wavelet of different values. We test the estimator's performance over a sample of 10^6 points, feeding the same synthetic data stream to each one of the estimators tested. We compute the average distance (both L_1 and L_2) from the true probability generating the data stream to the estimation. Finally, we compare these measures for the different estimators.

The general pattern for the triangular or square wavelets is as follows. For any fixed period P, the best fixed-size estimator is that whose window size is a certain fraction of P. ADWIN2 usually does sometimes does worse than this best fixed-size window, but only slightly, and often does better than even the best fixed size that we try. Additionally, it does better than any window of fixed size W when P is much larger or much smaller than W, that is, when W is a "wrong" time scale. The explanation is simple: if W is too large the estimator does not react quickly enough to change, and if W is too small the variance within the window implies a bad estimation. One can check that ADWIN2 adjusts its window length to about $P/4$ when P is small, but keeps it much smaller than P for large P, in order again to minimize the variance / time-sensitivity tradeoff.

In a third type of experiments, we test ADWIN2 as a change detector rather than as an estimator, and compare it to Gama's method [4]. The measures of interest here are the rate of changes detected and the mean time until detection.

To do this, we feed ADWIN2 and Gama's change detector with four data streams of lengths $L = 2,000$, $10,000$, $100,000$ and $1,000,000$ bits, generated from a Bernoulli distribution of parameter μ. We keep $\mu = 0.2$ stationary during the first $L - 1,000$ time steps, and then make it increase linearly during the last $1,000$ steps. We try different slopes: 0 (no change), 10^{-4}, $2{\cdot}10^{-4}$, $3{\cdot}10^{-4}$, and $4 \cdot 10^{-4}$.

To compare the rate of false negatives on an equal foot, we adjust ADWIN2 confidence parameter δ to have the same rate of false positives as Gama's method.

Table 2 shows the results for two data streams. Rows are grouped in four parts, corresponding to the four values of L that we tested. For each value of L, we give the number of changes detected in the last $1,000$ samples (summed over all runs) and the mean and standard distribution of the time until the change is detected, in those runs where there is detection.

The first column gives the ratio of false positives.

One observation we made is that Gama's method tends to detect many more changes early on (when the window is small) and less changes as the window grows. This explains that, on the first column, even if the ratio of false positives is the same, the average time until the first false positive is produced is much smaller for Gama's method.

The last four columns describe the results when change does occur, with different slopes. ADWIN2 detects change more often, with the exception of the $L = 2,000$ experiment. As the number of samples increases, the percentage of changes detected decreases in Gama's methodology; as discussed early, this is to be expected since it takes a long time for Gama's method to overcome the weight of past examples. In contrast, ADWIN2 maintains a good rate of detected changes, largely independent of the number of the number of past samples $L-1,000$. One can observe the same phenomenon as before: even though Gama's method detects less changes, the average time until detection (when detection occurs) is smaller.

4 Experiments with Naïve Bayes

4.1 An Incremental Naïve Bayes Predictor We compare two time-change management strategies. The first one uses a static model to make predictions. This model is rebuilt every time that an external change detector module detects a change. We use Gama's detection method and ADWIN2 as change detectors. Gama's method generates a warning example some time before actually declaring change; see [4] for the details; the examples received between the warning and the change signal are used to rebuild the model. In ADWIN2, we use the examples currently stored in the window to rebuild the static model.

The second one is incremental: we simply create an instance $A_{i,j,c}$ of ADWIN2 for each count $N_{i,j,c}$, and one for each value c of C. When a labelled example is processed, add a 1 to $A_{i,j,c}$ if $x_i = v \wedge C = c$, and a 0 otherwise, and similarly for N_c. When the value of $\Pr[x_i = v_j \wedge C = c]$ is required to make a prediction, compute it using the estimate of $N_{i,j,c}$ provided by $A_{i,j,c}$. This estimate varies automatically as $\Pr[x_i = v_j \wedge C = c]$ changes in the data.

Note that different $A_{i,j,c}$ may have windows of different lengths at the same time. This will happen when the distribution is changing at different rates for different attributes and values, and there is no reason to sacrifice accuracy in all of the counts $N_{i,j,c}$, only because a few of them are changing fast. This is the intuition why this approach may give better results than one monitoring the global error of the predictor: it has more accurate information on at least some of the statistics that are used for the prediction.

Results on synthetic data are omitted in this version.

4.2 Real-world data experiments We test the performance of our Naïve Bayes predictors using the Electricity Market Dataset described by M. Harries [5] and used by Gama [4]. This dataset is a real-world dataset where we do not know when drift occurs or if there is drift, hence it is not possible to build a static model for comparison as we did before.

This data was collected from the Australian New South Wales Electricity Market. The prices are not fixed and are affected by demand and supply of the market. The ELEC2 dataset contains 45312 instances dated from 7 May 1996 to 5 December 1998. Each example of the dataset refers to a period of 30 minutes, i.e. there are 48 instances for each time period of one day. Each example on the dataset has 5 fields, the day of week, the time stamp, the NSW electricity demand, the Vic electricity demand, the scheduled electricity transfer between states and the class label. The class label identifies the change of the price related to a moving average of the last 24 hours. The class level only reflect deviations of the price on a one day average and removes the impact of longer term price trends.

At each time step, we train a static model using the last 48 samples received. We compare this static model with other models, also on the last 48 samples. Table 3 shows accuracy results. In each column (a test), we show in boldface the result for ADWIN2 and for the best result.

ADWIN2 applied in the incremental time-change model does much better than all the others, with the exception of the shortest fixed-length window, which achieves 86.44% of the static performance compared to ADWIN2 's 83.62%. The reason for this anomaly is due to the nature of this particular dataset: by visual inspection, one can see that it contains a lot of short runs (length 10 to 20) of identical values, and therefore a myopic strategy (i.e., a short window) gives best results. ADWIN2 behaves accordingly and shortens its window as much as it can, but the formulas involved do not allow windows as short as 10 elements. In fact, we have tried replicating each instance in the dataset 10 times, and then ADWIN2 becomes the winner again.

We also test the prediction accuracy of these methods. We compare, as before, a static model generated at each time t to the other models, and evaluate them asking to predict the instance that will arrive at time $t+1$. The static model is computed training on the last 24 samples. The results are shown in Table 3. Generally, the incremental time-change management model

Table 2: Change detection experiments.

Slope	0	10^{-4}	$2 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$4 \cdot 10^{-4}$
$2 \cdot 10^3$ **samples,** 10^3 **trials**					
Detection time (Gama)	854 ± 462	532 ± 271	368 ± 248	275 ± 206	232 ± 178
%runs detected (Gama)	10.6	58.6	97.2	100	100
Detection time ADWIN2)	975 ± 607	629 ± 247	444 ± 210	306 ± 171	251 ± 141
%runs detected (ADWIN2)	10.6	39.1	94.6	93	95
10^5 **samples, 100 trials**					
Detection time (Gama)	12,164 ± 17,553	127 ± 254	206 ± 353	440 ± 406	658 ± 422
%runs detected (Gama)	12	4	7	11	8
Detection time ADWIN2)	47,439 ± 32,609	878 ± 102	640 ± 101	501 ± 72	398 ± 69
%runs detected (ADWIN2)	12	28	89	84	89

Table 3: Naïve Bayes, Electricity data benchmark, testing on last 48 items and on next instance.

		Testing on last 48 items Static = 91.62%		Testing on next instance Static = 94.40%	
	Width	%Dynamic	% Dynamic/Static	%Dynamic	% Dynamic/Static
Gama Change Detection		45.94%	50.14%	45.87%	48.59%
ADWIN2 Change Detection		60.29%	65.81%	46.86%	49.64%
ADWIN2 for counts		**76.61%**	**83.62%**	**72.71%**	**77.02%**
Fixed-sized Window	32	**79.13%**	**86.44%**	71.54%	75.79%
Fixed-sized Window	128	72.29%	78.97%	68.78%	72.87%
Fixed-sized Window	512	68.34%	74.65%	67.14%	71.13%
Fixed-sized Window	2048	65.02%	71.02%	64.25%	68.07%
Fixed-sized flushing Window	32	78.57%	85.83%	71.62%	75.88%
Fixed-sized flushing Window	128	73.46%	80.24%	70.12%	74.29%
Fixed-sized flushing Window	512	69.65%	76.08%	68.02%	72.07%
Fixed-sized flushing Window	2048	66.54%	72.69%	65.60%	69.50%

does much better than the static model that refreshes its NB model when change is detected.

5 Conclusions

We have described a new method for dealing with distribution change and concept drift when learning from data sequences that may vary with time. We developed an algorithm ADWIN using sliding windows whose size is recomputed online according to the rate of change observed from the data in the window itself. This delivers the user from having to choose any parameter (for example, window size), a step that most often ends up being guesswork. So, client algorithms can simply assume that ADWIN stores the currently relevant data.

We tested on both synthetic and real datasets, showed that ADWIN2 really adapts its behavior to the characteristics of the problem at hand.

References

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. 21st ACM Symposium on Principles of Database Systems*, 2002.

[2] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[3] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 14(1):27–45, 2002.

[4] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *SBIA Brazilian Symposium on Artificial Intelligence*, pages 286–295, 2004.

[5] M. Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales, 1999.

[6] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proc. 30th VLDB Conf., Toronto, Canada*, 2004.

[7] S. Muthukrishnan. Data streams: Algorithms and applications. In *Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.