On the Inter-relationships among Drift rate, Forgetting rate, Bias/variance profile and Error

**Nayyar A. Zaidi · Geoffrey I. Webb ·
Francois Petitjean · Germain Forestier**

Abstract We propose two general and falsifiable hypotheses about expectations on generalization error when learning in the context of concept drift. One posits that as drift rate increases, the forgetting rate that minimizes generalization error will also increase and vice versa. The other posits that as a learner's forgetting rate increases, the bias/variance profile that minimizes generalization error will have lower variance and vice versa. These hypotheses lead to the concept of the *sweet path*, a path through the 3-d space of alternative drift rates, forgetting rates and bias/variance profiles on which generalization error will be minimized, such that slow drift is coupled with low forgetting and low bias, while rapid drift is coupled with fast forgetting and low variance. We present experiments that support the existence of such a sweet path. We also demonstrate that simple learners that select appropriate forgetting rates and bias/variance profiles are highly competitive with the state-of-the-art in incremental learners for concept drift on real-world drift problems.

1 Introduction

The world is dynamic, in a constant state of flux. Traditional learning systems that learn static models from historical data are unable to adjust to *concept drift* — changes in the distributions from which data are drawn. A growing body of experimental machine learning research investigates incremental learners that seek to adjust models as appropriate when confronted with concept drift (Gaber et al, 2005; Gama and Rodrigues, 2009; Aggarwal, 2009; Žliobaite, 2010; Bifet et al, 2011; Nguyen et al, 2014; Brzezinski and Stefanowski, 2014; Krempl et al, 2014; Gama et al, 2014; Ditzler et al, 2015). This paper seeks to inform this line of research by identifying relationships between types of

Nayyar A. Zaidi, Geoffrey I. Webb, Francois Petitjean, Germain Forestier
Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia
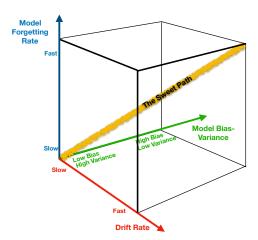E-mail: {firstname.lastname}@monash.edu

Fig. 1: An illustration of the hypothesized *sweet path*, a four way interaction among drift rate, forgetting rate, learner's bias and variance and expected error.

concept drift and the properties of the learners that will best handle those forms of drift. Specifically, we propose and investigate two hypotheses —

1. *The drift-rate/forgetting-rate nexus.* As the rate of concept drift increases, model accuracy will in general be maximized by increasing forgetting rates, and conversely, as the drift rate decreases, model accuracy will in general be maximized by decreasing forgetting rates. Here *increasing forgetting rates* means focusing on more recent evidence by reducing window sizes or increasing decay rates. *Decreasing forgetting rates* means focusing on longer term evidence by increasing window sizes or decreasing decay rates.
2. *The forgetting-rate/bias-variance-profile nexus.* As forgetting rates increase, model accuracy will in general be maximized by altering the bias/variance profile of a learner to decrease variance, and conversely, as forgetting rates increase, model accuracy will in general be maximized by decreasing bias.

The first of these hypotheses is intuitive. The faster the world is changing, the less relevance older information will have. In consequence, more aggressive forgetting mechanisms, specifically smaller windows or higher decay rates, will be required to exclude older examples for which the trade-off between providing additional information that is relevant to the current state-of-the-world and that which is misleading is weighted too heavily to the latter.

The second hypothesis derives from the hypothesis that when learning from smaller quantities of data, lower variance learners will maximize accuracy due to their ability to avoid overfitting, whereas when learning from larger datasets lower bias learners will maximize accuracy, due to their ability to model the details present in large data (Brain and Webb, 1999). The more past data we forget, the smaller the effective data quantity from which we learn and, therefore, with high-forgetting rate, low-variance models are more desirable and low-bias models with low-forgetting rate.

Put together these hypothesized effects imply the *sweet path* for concept drift illustrated in Figure 1, whereby the lowest error for a low drift rate will be achieved by a low bias learner with a low forgetting rate and the lowest error for a high drift rate will be achieved by a low variance learner with a high forgetting rate.

The bulk of this paper (Section 2 to Section 5) comprises detailed experiments that investigate the two hypotheses and the hypothesized sweet path. We discuss implications and directions for future research in Section 6.

2 Background

In supervised machine learning we seek to learn a model $\mathcal{M}$ that can predict the value (or probability of each value) y of a target variable Y for an example $x = \langle x_1, \ldots, x_a \rangle$ of an input variable $X = \langle X_1, \ldots, X_a \rangle$. We learn $\mathcal{M}$ from a training set $T = \{\langle x^1, y^1 \rangle, \ldots, \langle x^s, y^s \rangle\}$. In incremental learning, the training set is presented to the learner as a sequence over a period of time and the learner updates $\mathcal{M}$ in light of each new example or set of examples as it is encountered.

Concept drift occurs when the distribution $P_t(X, Y)$ from which the data are drawn at time t differs from that at subsequent time u, $P_u(X, Y)$.[1]

We can measure the *magnitude* of drift from time t to u, $D(t, u)$, by a measure of distance between the probability distributions $P_t(X, Y)$ and $P_u(X, Y)$ and the *rate* of drift at time t by:

$$Rate_t = \lim_{n \to \infty} nD\left(t-0.5/n, t+0.5/n\right) \qquad (1)$$

(Webb et al, 2016). The observations in this paper hold for any distance measure that is a metric, such as the Total Variation Distance (Levin et al, 2008).

Forgetting mechanisms are a standard strategy for dealing with concept drift. The two main forgetting mechanisms are *windowing*, in which a sliding window is maintained containing only the W most recent examples; and *decay* or *weighting*, in which greater weight is placed on more recent examples and lesser weight on older ones (Gama et al, 2014).

3 Experimental setup

To explore the drift-rate/forgetting-rate/bias-variance-profile nexus, we require an incremental learner that can learn from sliding windows or with decay. Of course, we also require means of varying the learner's bias/variance profile.

For our experiments, we use the semi-naive Bayesian method AnDE (Webb et al, 2012), as it satisfies these requirements. First, the model has a tuneable

---

[1] Note that some papers (Gama et al, 2014) distinguish *real concept drift* in which $P(Y \mid X)$ changes, from virtual concept drift in which $P(X)$ changes. For the purposes of this paper we do not distinguish between these, as the distinction does not appear pertinent.

parameter n that controls the representation bias and variance. When $n = 0$ (in AnDE), one gets a naive Bayes classifier which is highly biased but has low variance. Higher values of n decrease bias at the cost of an increase in variance and lower values decrease variance at a cost of increased bias. Second, the AnDE model can be represented using counts of observed marginal frequencies, the dimensionality of each of which is controlled by n. As described below, these can readily be incrementally updated to reflect a sliding window or incremental decay without need for relearning the entire model.

The goal of Bayesian methods is to factorize the joint distribution: $\mathrm{P}(y, \mathbf{x})$. The AnDE model factorizes the joint distribution as:

$$
\hat{\mathrm{P}}_{\mathrm{AnDE}}(y, \mathbf{x}) = \begin{cases} \sum_{s \in \binom{\mathcal{A}}{n}} \delta(x_s)\hat{\mathrm{P}}(y, x_s) \prod_{i=1}^{a} \hat{\mathrm{P}}(x_i \mid y, x_s) / \sum_{s \in \binom{\mathcal{A}}{n}} \delta(x_s) \ : \sum_{s \in \binom{\mathcal{A}}{n}} \delta(x_s) > 0 \\ \\ \hat{\mathrm{P}}_{\mathrm{A(n\text{-}1)DE}}(y, \mathbf{x}) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad : \text{otherwise} \end{cases}
$$

$$(2)$$

where $\binom{\mathcal{A}}{n}$ indicates the set of all size-n subsets of $\{1, \ldots a\}$ and $\delta(x_\alpha)$ is a function that is 1 if the training data contains an object with the value x_α, otherwise 0.

3.1 Window-based Adaptation

A sliding window that supports learning only from the last W data points can be achieved simply with a queue-based data structure. At each time step t, when a new data point $\langle \mathbf{x}, y \rangle$ arrives:

- Increment relevant count statistics based on $\langle \mathbf{x}, y \rangle$
- Push $\langle \mathbf{x}, y \rangle$ onto the queue
- If queue length exceeds W
    - $\langle \tilde{\mathbf{x}}, \tilde{y} \rangle$ = De-queue.
    - Decrement relevant count statistics based on $\langle \tilde{\mathbf{x}}, \tilde{y} \rangle$

It can be seen that parameter W controls the forgetting rate. Large W means large windows, hence slow forgetting and small W means small windows, hence fast forgetting.

3.2 Decay-based Adaptation

To support incremental exponential decay, before adding the count statistics of the data point $\mathbf{x}$ at step t, all that is required is that the counts in the count table are decayed. For example if $N_{x_i,y}$ denotes the stored count of the number of times attribute i takes value x_i and class attribute takes the value y, it is decayed as:
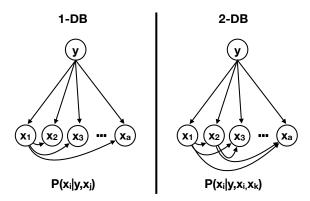
$$
N_{x_i,y} = N_{x_i,y} * \exp(-D),
$$

Fig. 2: Simple illustration of the structure of superparent k-DB classifiers. (Left) superparent 1-DB, each variable takes one more parent other than the class, (Right) superparent 2-DB, each variable takes two more parents other than the class.

where D is the decay parameter.

Like W in window-based adaptation, it can be seen that parameter D controls the model adaptation rate. Large D means large decay, and hence fast model adaptation-rate, and, small D means small decay, and hence a slow model adaptation-rate.

4 Data Generation

To test our hypotheses, we require data streams with varying drift rates. To this end we create a framework where we can generate synthetic data for which we can systematically manipulate the rate of drift.

We represent the probability distribution using the most common formalism for doing so, a Bayesian network. Because changing the probability distribution at a node may change all of the probability distributions of all its children and their descendant nodes, in order to allow systematic manipulation of the drift rate we minimize the number of parent nodes. Specifically, we sample from superparent k-DB (Keogh and Pazzani, 1999) distributions. We show the structure of superparent 1 and 2-DB in Figure 2. In a superparent k-DB model, each attribute X_i other than the first k attributes takes $X_1, \ldots X_k$ and Y as its parents. In a superparent 1-DB structure every attribute X_i other than X_1 takes Y and X_1 as its parents and in a superparent 2-DB structure every attribute other than X_1 and X_2 takes Y, X_1 and X_2 as its parents. These structures are shown in Figure 2.

Specifying such networks is simple – the standard process is:

1. Specify the (possibly empty) set π_{X_i} of parents for each node X_i.

2. Fill the Conditional-Probability-Table (CPT) for each node. This specifies
   a probability distribution over the values of the attribute for each combi-
   nation of values of the parents.

Once the network is specified, one can use *Ancestral Sampling* (Bishop, 2006),
to generate data therefrom.

We use the following simple (heuristic) procedure to generate the network:

- All attributes including the class are binary.
- Set the initial number of attributes to 100.
- We aim for half of these attributes to have three parents, and the remaining
  half to have two parents in order that the resulting distributions do not too
  closely fit the biases of a single AnDE learner. Note, however, that as we are
  creating superparent k-DB structures, X_1 can only have Y as a parent and
  that X_2 must have X_1 and Y as parents. Therefore, in practice, we have
  1 attribute with 1 parent, 49 attributes with 2 parents and 50 attributes
  with 3 parents.
- To allow direct control over the rate of drift, we do not allow parent at-
  tributes to drift. To maximize diversity, we thus wish to minimize the
  number of parent attributes, which is why we use superparent 1-DB and
  2-DB structures instead of more general k-DB structures. To this end, for
  X_3 to X_{50}, we uniformly at random select either X_1 or X_2, together with
  Y as parents. For X_{51} to X_{100}, all of Y, X_1 and X_2 are assigned as parents.
- Once the structure is specified, we randomly initialize the CPTs. Note, we
  have to fulfill the sum-to-one constraint that $P(X_i = 0 \mid \pi_{X_i}) + P(X_i =
  1 \mid \pi_{X_i}) = 1$. To this end, for each combination of values for the parent
  attributes, we randomly select a value between 0 and 1 for $P(X_i = 0 \mid \pi_{X_i})$,
  and set $P(X_i = 1 \mid \pi_{X_i}) = 1 - P(X_i = 0 \mid \pi_{X_i})$.
- Next, we add 100 more binary attributes which have no parents and hence
  represent noise. The CPTs for these nodes are also initialized randomly as
  above. In total, we now have 200 attributes.

To sample from the distribution defined by this network, we:

- Choose the class y by uniformly at random selecting either 0 or 1.
- For $i = 1$ to 100 sample x_i from the distribution defined by $P(X_i \mid
  y, x_1, \ldots, x_{i-1})$.

Introducing Drift To introduce drift we want to change the CPTs in a
controlled manner. We need to strictly control the change because we need to
systematically increase and decrease the rate of drift. To this end we ensure
that —

- Y, X_1 and X_2 (the three nodes that are parents to the other nodes) do
  not drift, therefore, their CPTs will not be changed through-out the data
  generation process.
- Drift occurs only after every T steps.
- Drift only influences $X\%$ of the attributes.

The first constraint is necessary because changing parent probabilities will indirectly change all the child probabilities in a complex manner that is difficult to manage. The second and third constraints ensure that there is short term directionality in the drift. Simply randomly drifting every attribute 1/10th of the drift rate every step results in half the steps simply canceling out the previous step for the attribute. However, if only some attributes are drifted at each step and drift occurs only every T steps it ensures that non-trivial drift lasts for a non-trivial period of time.

Throughout the experiments, we will set T to 10 and X is set to 50. That is, half of the attributes are randomly selected after every 10-th step and are drifted.

The drifting process is controlled by a single parameter Δ. The method is designed to ensure that a higher value of Δ leads to a fast drift, and a smaller value of Δ will lead to a slow drift. When a node is drifted, Δ is either added or subtracted from each of its CPT values in a manner to ensure that the values sum to 1.0 and no value exceeds 1.0 or falls below 0.0, hence maintaining a valid probability distribution.

5 Experimental Analysis

We have seen how parameter n controls the bias/variance profile of the model, how parameters W and D control the forgetting rate, and how parameter Δ controls the rate of the drift. In this section, we present experiments that systematically study the interaction among these factors.

We use $\Delta = 0.05$ to represent fast drift, $\Delta = 0.01$ for medium drift and $\Delta = 0.0005$ for slow drift. We select these from a wider range of values explored as exemplars that demonstrate clear differences in outcomes. As an example of the rates not presented, for $\Delta = 0.02$ it is less clear which out of our fast and medium forgetting rates provides lower error, as our first hypothesis predicts.

We generate data streams of 5,000 successive time steps, at each time step drawing one example randomly from the probability distribution for that step and drifting the distribution every 10 steps.

We use prequential evaluation, whereby at each time step the current model is applied to classify the next example in the data stream and then the example is used to update the model. We plot the resulting error rates, where each point in the plot is the average error over 50 successive time steps. We run each experiment 150 times for NB and A1DE and 100 times for A2DE (due to there being insufficient time to complete more runs). We present averages over all runs.

We first present results for fast drift. Figure 3 presents results using windows for forgetting and Figure 4 presents results using decay for forgetting. The average prequential error for the window size or decay rate that achieves the lowest such error is listed for each of the three classifiers (Figures 3d and 4d).
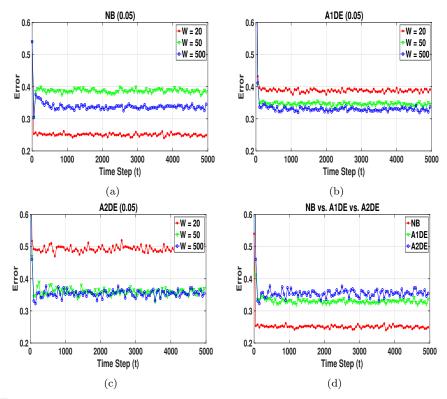
Fig. 3: Windowing with Fast Drift ($\Delta = 0.05$) – Variation in prequential loss of NB (Figure 3a), A1DE (Figure 3b) and A2DE (Figure 3c) with window sizes of 20, 50 and 500. Figure 3d: Comparison of NB (error = 0.253), A1DE (error = 0.337) and A2DE (error = 0.361) with best window size.
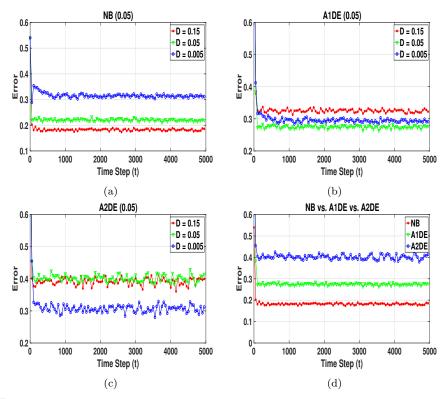
Fig. 4: Decay with Fast Drift ($\Delta = 0.05$) – Variation in prequential loss of NB (Figure 4a), A1DE (Figure 4b) and A2DE (Figure 4c) with decay rates of 0.15, 0.05 and 0.005. Figure 4d: Comparison of NB (error = 0.186), A1DE (error = 0.283) and A2DE (error = 0.408) with best decay rate.

Here we see that for NB, as predicted, the lowest error is achieved with fast forgetting (window size 20; decay rate 0.15).

However, contrary to our expectations, A1DE and A2DE achieve their lowest error with slower forgetting. This is because these models effectively fail in the face of such rapid drift. Recall that A1DE must estimate for every attribute X_i and attribute X_j both $P(Y, X_i)$ and $P(X_j \mid Y, X_i)$. A2DE must estimate for every attribute X_i, attribute X_j and attribute $X_k - P(Y, X_i, X_j)$ and $P(X_k \mid Y, X_i, X_j)$. The distributions for Y, X_1 and X_2 are not drifting, but all others are drifting at a rapid rate. Larger window sizes allow these classifiers to produce more accurate estimates of the unvarying probabilities, $P(Y, X_1)$, $P(Y, X_2)$, $P(X_1 \mid Y, X_2)$, $P(X_2 \mid Y, X_1)$ and $P(Y, X_1, X_2)$, whereas no window size provides accurate estimates of the remaining probabilities because either they are too small to provide accurate estimates or the distributions change too much over the duration of the window for the estimate to be accurate. Thus, the relative error is dominated by the ability to accurately estimate the invariant probabilities and the performance approximates learning from a stationary distribution when the majority of attributes are noise attributes.

At an intermediate drift rate ($\Delta < 0.25$), the intermediate bias learner A1DE starts to outperform NB. Figures 5 and 6 show prequential 0-1 Loss with different window sizes and decay rates with a drift delta of size 0.01. It is apparent that for this intermediate drift rate the intermediate window size (50) and intermediate decay rate (0.05) attain the lowest error and that the learner with intermediate bias (A1DE) minimizes overall error. Figures 5d and 6d, shows the comparison of NB, A1DE and A2DE with their best respective window size and decay rate, and it can be seen that A1DE results in best performance.

Figures 7 and 8 show prequential error with a slow drift rate, $\Delta = 0.0005$, with varying window sizes and decay rates. Figure 7d, compares the performance of the three models with their respective best window size and Figure 8d with their best decay rate. In both cases it is apparent that A2DE achieves the lowest error.

Thus, in all three scenarios of fast, intermediate and slow drift we find the relationship predicted by the sweet path between drift rate, forgetting rate and bias/variance profile.

5.1 Exploiting the insights of the sweet path for designing practical learners

We have demonstrated that a generalizable and falsifiable hypothesis is consistent with experimental outcomes. This raises the question of how the resulting insights might be used to create new effective learners that can respond to concept drift. Unfortunately, doing so appears to be far from trivial. If we allow that drift rates may vary over time, the sweet path suggests that we need learners that can adapt to changes in drift rates with corresponding adaptation in their forgetting rates and bias/variance profiles. We are yet to devise a practical algorithmic solution to the complex problem.
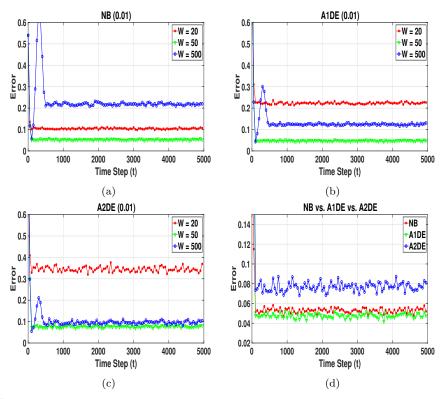
Fig. 5: Windowing with Medium Drift ($\Delta = 0.01$) – Variation in prequential loss of NB (Figure 5a), A1DE (Figure 5b) and A2DE (Figure 5c) with window sizes of 20, 50 and 500. Figure 5d: Comparison of NB (error = 0.0531), A1DE (error = 0.047) and A2DE (error = 0.083) with best window size.
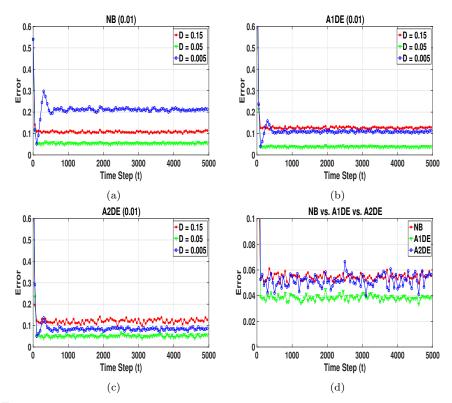
Fig. 6: Decay with Medium Drift ($\Delta = 0.01$) – Variation in prequential loss of NB (Figure 6a), A1DE (Figure 6b) and A2DE (Figure 6c) with varying decay rates of 0.15, 0.05 and 0.005. Figure 6d: Comparison of NB (error = 0.0599), A1DE (error = 0.0498) and A2DE (error = 0.0629) with best decay rates.
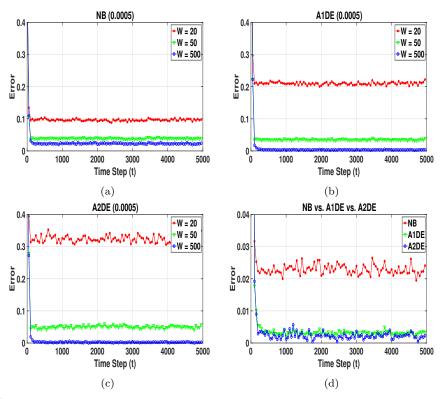
Fig. 7: Windowing with Slow Drift ($\Delta = 0.0005$) – Variation in prequential loss of NB (Figure 7a), A1DE (Figure 7b) and A2DE (Figure 7c) with window sizes of 20, 50 and 500. Figure 7d: Comparison of NB (error = 0.0289), A1DE (error = 0.0156) and A2DE (error = 0.0151) with best window size.
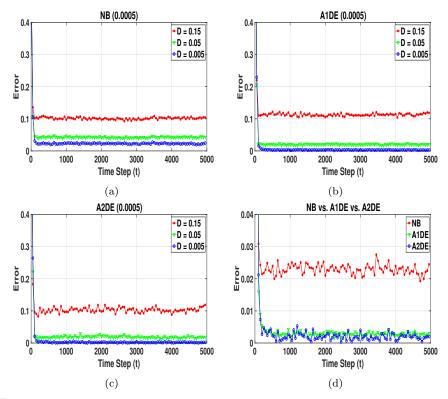
Fig. 8: Decay with Slow Drift ($\Delta = 0.0005$) – Variation in 0-1 Loss of NB (Figure 8a), A1DE (Figure 8b) and A2DE (Figure 8c) with varying decay rates of 0.15, 0.05 and 0.005. Figure 8d: Comparison of NB (error = 0.0290), A1DE (error = 0.0153) and A2DE (error = 0.0148) with best decay size.

	#Instances	#Attributes	#Classes
PowerSupply	29928	2	2
Airlines	539383	7	2
ElectricNorm	45312	8	2
Sensor	2219803	5	58

Table 1: Details of datasets.

To assess the practical implications of our hypotheses for designing practical learning algorithms, we compare the performance on real-world drifting data of our AnDE classifiers with differing forgetting rates and bias/variance profiles against a range of state of the art concept drift classifiers. We use 4 standard benchmark drift datasets, whose details are given in Table 1. Numeric attributes are discretized using IDAW discretization (Webb, 2014) with 5 intervals.

We compare the performance with 12 standard learning techniques:

1. AccuracyUpdatedEnsemble (Brzezinski and Stefanowski, 2014),
2. AccuracyWeightedEnsemble (Wang et al, 2003),
3. DriftDetectionMethodClassifier (Gama et al, 2004),
4. DriftDetectionMethodClassifierEDDM (Baena-Garcıa et al, 2006),
5. HoeffdingAdaptiveTree (Bifet and Gavaldà, 2009),
6. HoeffdingOptionTree (Pfahringer et al, 2007),
7. HoeffdingTree (Hulten et al, 2001),
8. LeveragingBag (Bifet et al, 2010),
9. OzaBag (Oza and Russell, 2001),
10. OzaBagAdwin (Bifet et al, 2009),
11. OzaBoost (Oza and Russell, 2001),
12. OzaBoostAdwin (Oza and Russell, 2001; Babcock et al, 2002).

It can be seen from Table 2 that `AccWeightedEns` (AccuracyWeightedEnsemble) achieved the lowest error on `PowerSupply`, `AccUpdatedEn` (AccuracyUpdateEnsemble) the lowest on `Airlines`, whereas `LeveregingBag` (LeveregingBagging) achieved the lowest error on `ElectricNorm` and `Sensor`. In the following, we will use these (best) results as benchmarks and see how adaptive AnDE with decay and window based adaptation can perform relative to these results.

We compare the performance of adaptive AnDE with window-based adaptation in Figure 9, depicting results with various window sizes. The lowest error obtained on the dataset by one of 12 standard techniques, is also plotted as a horizontal blue line for comparison. It can be seen that except for `PowerSupply`, adaptive AnDE can achieve lower error than the lowest of any of the twelve state-of-the-art techniques. For `Airlines`, A1DE achieves the lowest error of 0.3309 and for `ElectricNorm` and `Sensors`, A2DE achieves errors of: 0.1124 and 0.2250.

A comparison of adaptive AnDE with decay-based adaptation is given in Figure 10. Similar to window-based results, adaptive AnDE achieved lower error than the lowest achieved by any of the state-of-the-art techniques on all but

	AccUpdatedEns	OzaBagAdwin	DriftDetClassifier	DriftDetClassifierEDDM
PowerSupply	0.8599	0.8692	0.8634	0.8615
Airlines	**0.3335**	0.3448	0.3534	0.3511
ElectricNorm	0.2219	0.167	0.1984	0.149
Sensor	0.3102	0.2874	0.3206	0.3159
	ASHoeffdingTree	HoeffdingTree	OzaBag	HoeffdingAdaptiveTree
PowerSupply	0.864	0.864	0.8655	0.8661
Airlines	0.3552	0.3552	0.3575	0.3632
ElectricNorm	0.2007	0.2007	0.1982	0.1759
Sensor	0.7153	0.7153	0.7067	0.3718
	OzaBoost	AccWeightedEns	LeveragingBag	OzaBoostAdwin
PowerSupply	0.9583	**0.8579**	0.8717	0.9584
Airlines	0.3719	0.3751	0.3769	0.3888
ElectricNorm	0.1781	0.2471	**0.1303**	0.143
Sensor	0.9514	0.3596	**0.2395**	0.407

Table 2: Comparison of 0-1 Loss performance of 12 standard concept drift techniques on four real-world datasets: `PowerSupply, Airlines, ElectricNorm, Sensor`. The best results are highlighted in bold font.
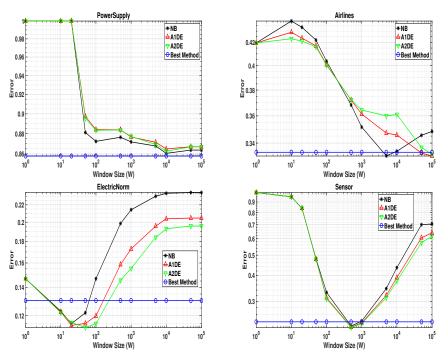

Fig. 9: Comparison of adaptive NB, A1DE and A2DE based on window-based adaptation on four real world datasets: `PowerSupply, Airlines, ElectricNorm, Sensor`. Horizontal blue line depicts the best performance out of 12 standard techniques.
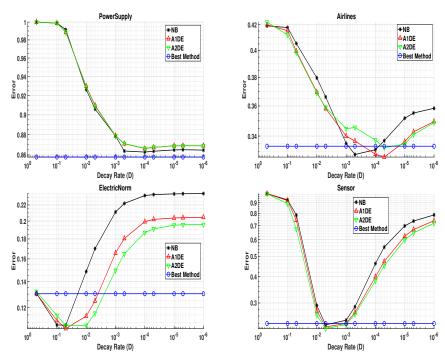
Fig. 10: Comparison of adaptive NB, A1DE and A2DE based on decay-based adaptation on four real world datasets: `PowerSupply, Airlines, ElectricNorm, Sensor`. Horizontal blue line depicts the best performance out of 12 standard techniques.

the `PowerSupply` dataset. On `Airlines` and `ElectricNorm`, A1DE achieved error of 0.3267 and 0.1061 respectively, and on `Sensor`, A2DE achieved error of 0.2268.

Note that we are not suggesting that our learners are currently suited for practical use. There is still need to find effective mechanisms to dynamically select forgetting rates and bias/variance profiles. Also, while our learners outperformed the best of the state-of-the-art, many of those learners could also have performed better if a sweep had been performed over their metaparameters.

6 Conclusions

We have proposed novel, generalizable and falsifiable hypotheses about the inter-relationships between drift rates, forgetting mechanisms, bias/variance profiles and error. Our experiments with the AnDE learners have been consistent with the predictions of the hypotheses, that there will be a *sweet path* whereby as the drift rate increases the optimal forgetting rates will also increase; and as forgetting rates increase the optimal model variance profile will decrease.

The AnDE classifiers are well suited to this study due to their efficient support of incremental learning with both windowing and decay and the range of bias/variance profiles that they provide.

Our studies with real-world data show that this framework can result in prequential accuracies that are highly competitive with the best of the state-of-the-art. Development of practical techniques for dynamically selecting and adjusting forgetting rates and bias/variance profiles as drift rates vary remains a promising avenue for future research.

Another intriguing insight that invites further investigation arises from the observation that for fast drift the lowest bias learner achieved lowest error with the slowest forgetting rate, in apparent disagreement with our hypotheses. As we discuss in Section 5, this is due to its failure to learn from the fast drifting attributes and hence its performance being dominated by the attribute subspace that is stationary (Y, X_1 and X_2). This gives support to our argument elsewhere (Webb et al, in press) that it is important to analyze drift in marginal distributions as well as at the course global level. It invites the development of learners that bring different forgetting rates and bias/variance profiles to bear on different attribute sub-spaces at different times as their rates of drift vary.

We hope that our hypotheses will provide insights into how to optimize a wide range of mechanisms for handling concept drift and will stimulate future research.

7 Acknowledgments

A Code

The code used in this work can be downloaded from repository: `https://github.com/nayyarzaidi/SweetPathVoyager`.

References

Aggarwal C (2009) Data Streams: An Overview and Scientific Applications, Springer Berlin Heidelberg, pp 377–397. DOI 10.1007/978-3-642-02788-8_14

Babcock B, Datar M, Motwani R (2002) Sampling from a moving window over streaming data. In: Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, pp 633–634

Baena-Garcıa M, del Campo-Ávila J, Fidalgo R, Bifet A, Gavalda R, Morales-Bueno R (2006) Early drift detection method. In: Fourth International Workshop on Knowledge Discovery from Data Streams, vol 6, pp 77–86

Bifet A, Gavaldà R (2009) Adaptive learning from evolving data streams. In: Advances in Intelligent Data Analysis VIII, Springer, pp 249–260

Bifet A, Holmes G, Pfahringer B, Kirkby R, Gavaldà R (2009) New ensemble methods for
    evolving data streams. In: Proceedings of the 15th ACM SIGKDD international confer-
    ence on Knowledge discovery and data mining, ACM, pp 139–148
Bifet A, Holmes G, Pfahringer B (2010) Leveraging bagging for evolving data streams. In:
    Machine Learning and Knowledge Discovery in Databases, Springer, pp 135–150
Bifet A, Gama J, Pechenizkiy M, Zliobaite I (2011) Handling concept drift: Importance,
    challenges and solutions. PAKDD-2011 Tutorial, Shenzhen, China
Bishop C (2006) Pattern Recognition and Machine Learning (Information Science and Statis-
    tics). Springer-Verlag New York, Inc., Secaucus, NJ, USA
Brain D, Webb G (1999) On the effect of data set size on bias and variance in classification
    learning. In: Richards D, Beydoun G, Hoffmann A, Compton P (eds) Proceedings of the
    Fourth Australian Knowledge Acquisition Workshop (AKAW-99), The University of New
    South Wales, Sydney, pp 117–128
Brzezinski D, Stefanowski J (2014) Reacting to different types of concept drift: The accuracy
    updated ensemble algorithm. Neural Networks and Learning Systems, IEEE Transactions
    on 25(1):81–94
Ditzler G, Roveri M, Alippi C, Polikar R (2015) Learning in nonstationary environments: A
    survey. IEEE Computational Intelligence Magazine 10(4):12–25
Gaber MM, Zaslavsky A, Krishnaswamy S (2005) Mining data streams: a review. ACM
    Sigmod Record 34(2):18–26
Gama J, Rodrigues P (2009) An Overview on Mining Data Streams, Studies in Compu-
    tational Intelligence, vol 206, Springer Berlin / Heidelberg, pp 29–45. DOI 10.1007/
    978-3-642-01091-0_2
Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. In:
    Bazzan A, Labidi S (eds) Advances in Artificial Intelligence   SBIA 2004, Lecture
    Notes in Computer Science, vol 3171, Springer Berlin Heidelberg, pp 286–295, DOI
    10.1007/978-3-540-28645-5_29
Gama J, Žliobaite I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept
    drift adaptation. ACM Computing Surveys 46(4):44:1–44:37, DOI 10.1145/2523813
Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams. In: Proceed-
    ings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery
    and Data Mining, KDD-01, ACM, pp 97–106
Keogh E, Pazzani M (1999) Learning augmented Bayesian classifiers: A comparison of
    distribution-based and classification-based approaches. In: Proceedings of the Interna-
    tional Workshop on Artificial Intelligence and Statistics, pp 225–230
Krempl G, Zliobaite I, Brzezinski D, Hullermeier E, Last M, Lemaire V, Noack T, Shaker
    A, Sievi S, Spiliopoulou M, Stefanowski J (2014) Open challenges for data stream mining
    research. In: ACM SIGKDD Explorations Newsletter, Springer, vol 16–1, pp 1–10
Levin D, Peres Y, Wilmer E (2008) Markov Chains and Mixing Times. American Mathe-
    matical Soc.
Nguyen H, Woon Y, Ng W (2014) A survey on data stream clustering and classification.
    Knowledge and Information Systems pp 1–35
Oza N, Russell S (2001) Online bagging and boosting. In: Artificial Intelligence and Statistics
    2001, Morgan Kaufmann, pp 105–112
Pfahringer B, Holmes G, Kirkby R (2007) New options for Hoeffding trees. In: Orgun M,
    Thornton J (eds) AI 2007: Advances in Artificial Intelligence, Lecture Notes in Computer
    Science, vol 4830, Springer, pp 90–99, DOI 10.1007/978-3-540-76928-6_11
Wang H, Fan W, Yu PS, Han J (2003) Mining concept-drifting data streams using ensem-
    ble classifiers. In: Proceedings of the Ninth ACM SIGKDD International conference on
    Knowledge Discovery and Data Mining, KDD-03, ACM, pp 226–235
Webb G (2014) Contrary to popular belief incremental discretization can be sound, compu-
    tationally efficient and extremely useful for streaming data. In: Proceedings of the 14th
    IEEE International Conference on Data Mining, pp 1031–1036, DOI 10.1109/ICDM.2014.
    123
Webb G, Boughton J, Zheng F, Ting K, Salem H (2012) Learning by extrapolation from
    marginal to full-multivariate probability distributions: Decreasingly naive Bayesian clas-
    sification. Machine Learning 86(2):233–272, DOI 10.1007/s10994-011-5263-6

Webb G, Hyde R, Cao H, Nguyen H, Petitjean F (2016) Characterizing concept drift. Data Mining and Knowledge Discovery 30(4):964–994, DOI 10.1007/s10618-015-0448-4

Webb G, Lee L, Goethals B, Petitjean F (in press) Analyzing concept drift and shift from sample data. Data Mining and Knowledge Discovery

Žliobaite I (2010) Learning under concept drift: an overview. CoRR abs/1010.4784, URL http://arxiv.org/abs/1010.4784, 1010.4784