Pattern Recognition 44 (2011) 78–96

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

λ-Perceptron: An adaptive classifier for data streams

N.G. Pavlidis [a,*], D.K. Tasoulis [b], N.M. Adams [b], D.J. Hand [a,b]

[a] Institute for Mathematical Sciences, Imperial College London, 53 Prince's Gate, SW7 2PG, United Kingdom
[b] Department of Mathematics, Imperial College London, 180 Queen's Gate, SW7 2AZ, United Kingdom

ARTICLE INFO

Article history:
Received 17 June 2009
Received in revised form
7 May 2010
Accepted 29 July 2010

Keywords:
Streaming data
Classification
Population drift
Online learning
Forgetting

ABSTRACT

Streaming data introduce challenges mainly due to changing data distributions (population drift). To accommodate population drift we develop a novel linear adaptive online classification method motivated by ideas from adaptive filtering. Our approach allows the impact of past data on parameter estimates to be gradually removed, a process termed forgetting, yielding completely online adaptive algorithms. Extensive experimental results show that this approach adjusts the forgetting mechanism to maintain performance. Moreover, it might be possible to exploit the information in the evolution of the forgetting mechanism to obtain information about the type and speed of the underlying population drift process.

1. Introduction

Recently a new class of data-intensive tasks has become widely recognised: tasks which involve data that arrives in multiple, rapid, time-varying *data streams* [1]. Streaming data applications arise in finance, network monitoring, security, telecommunications data management, web applications, manufacturing, sensor management, and many others. A data stream is an ordered sequence of data that can be read only once or a small number of times using limited computing and storage capabilities [2]. This has posed new challenges for data analysis since it introduces several limitations. First, data are not typically stored after being processed, due to the practical bounds on memory utilisation. Even if they are stored, the availability of direct random access to any of them is unlikely [1]. Second, streaming data applications are characterised by time-varying population distributions. In other words, the underlying data generating mechanism is constantly evolving [1,3,4]. Due to this characteristic, any data analysis procedure must have the capacity to automatically recognise or adapt to change as it happens.

In this paper we are concerned with classification tasks. A straightforward way to design a classification method for data streams would be to use any off-the-shelf classifier, and pre-estimate its parameters using a subset of the available data. Subsequently, the classifier can be used to make inferences about future samples. This, however, will only be valid under the assumption that the population distribution is static (i.e. remains unchanged over time). If this assumption is violated, this approach is no longer justified and the performance of classifiers so constructed can be unreliable [5,6].

The problem of changing population distributions has been termed *population drift* and has been recognised as an issue in numerous areas (many of which have data stream characteristics), including high-frequency finance [7], credit scoring [6], spam filtering [8], user preference tracking [9], telecommunications [10], and sensor networks [11]. In this work, we consider two real-world applications. The first is the automated identification of tumours in a video sequence. The classification of this imaging data is important as it can contribute to the early diagnosis of cancer. Datasets from video sequences are typically subject to large textural variations across and within video frames, requiring adaptive methods to deal with the changing data distribution. The second application we consider is the prediction of the direction of change in high frequency exchange rate series. High frequency foreign exchange rate data are known to be subject to population drift [7]. Predicting the direction of change can be exploited for profitable trading.

In discussing population drift it is useful to distinguish between abrupt and gradual change. In an *abruptly changing* environment the population distribution changes at distinct time points, called *change points*. Between two consecutive change points the population distribution is static. In contrast, in a *gradually drifting* environment the population distribution changes at each time-step.

To handle abrupt change we can use a static classifier whose parameters are reset after the detection of each change point. Abrupt change detection is a non-trivial problem in most real

* Corresponding author. Tel.: +44 20 75940996; fax: +44 20 75940923.
E-mail addresses: n.pavlidis@imperial.ac.uk (N.G. Pavlidis), d.tasoulis@imperial.ac.uk (D.K. Tasoulis), n.adams@imperial.ac.uk (N.M. Adams), d.j.hand@imperial.ac.uk (D.J. Hand).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.07.026

applications and the effectiveness of this approach relies on the timely identification of each change point. Change detection methods monitor either the estimated parameters, or a measure of classification performance. An example of the former approach is [12], where a Kalman filter like approach is used to monitor the parameters of a credit scoring classifier, providing a charting mechanism to identify changes. Methods like CUSUM from sequential analysis [13] monitor classification performance to identify abrupt changes.

The problem of classification in abruptly changing environments has also been studied in the context of *prediction with expert advice* [14]. In this setting classification is performed based on the advice of a set of classifiers (experts). Hebster and Warmuth [15] provide regret bounds versus the best expert which is allowed to change a pre-specified number of times. This formulation has been generalised in [16–19]. The focus of this paper is on the development of individual classifiers suitable for streaming data. Methods that handle population drift through the selection or combination of a set of classifiers, such as prediction with expert advice and ensemble methods, are beyond the scope of this work.

A natural way to handle gradual population drift is to estimate parameters using a subset of the previously observed examples. The simplest such approach is to use a window of predefined length containing the most recent examples. More sophisticated approaches attempt to adapt the window size and/or the examples stored in the window according to the speed of drift [20,21]. Instead of using a subset of previous examples, online methods adapt to changing population distributions using information only from the current example to update the classifier. A general overview of a number of online classification methods is presented in the next section.

In this work, motivated by ideas from adaptive filtering [22,23], we develop a novel adaptive online estimation method for perceptrons with sigmoidal activation functions. A perceptron with a logistic activation function, $\varphi(x)=e^x/(1+e^x)$, is similar to the logistic regression model but parameters are estimated through the optimisation of a least squares criterion rather than a likelihood function. In the rest of the paper we refer to perceptrons with sigmoidal activation function as *sigmoid perceptrons*. Our approach aims to accommodate any type of population drift without storing past data. To achieve this, we define a cumulative error function that gradually removes the impact of past data on current parameter estimates, without relying on a model for population drift. The underlying assumption in our formulation is that more recent examples are more relevant to the current problem. Although this assumption can be violated in practise it enables the development of methods that can cope with different types of population drift without completely disregarding all past information.

Updating the parameters is achieved through gradient descent on the error function. The rate at which previous information is *forgotten* is controlled by a critical parameter that determines the responsiveness of the classifier to recently observed examples. Since in data streams the type and speed of population drift can change, the optimal value of this *forgetting factor* can also be time-varying. We propose to adjust the value of the forgetting factor in a data-driven manner by deriving the gradient of the error function with respect to this parameter and performing gradient descent at each time-step.

We should note that the accurate computation of these gradients requires storing and iterating over all the data, which is infeasible in streaming data applications. To this end, we employ an online algorithm which is a modification of stochastic gradient descent [24], that allows us to update all the required parameters, using data as it becomes available and not storing them. Extensive experimental results show that this approach has the ability to adjust the degree of forgetting to maintain performance. Moreover, it might be possible to exploit the information in the evolution of the time-varying forgetting factor to obtain information about the type and the speed of the underlying population drift process.

The paper is organised as follows: the next section presents related work on online classification. In Section 3, we introduce the data stream framework, by proposing an appropriate formulation of the cumulative error and discuss in detail the effect of population drift. Next in Sections 3.1 and 3.2 we propose adaptive methodologies for the parameters of our model, and Section 3.3 presents the proposed algorithmic scheme. Subsequently, Section 4 presents the experimental analysis of the proposed schemes in different settings and compares them against other approaches in the literature, using simulated and publicly available data. Additionally, real datasets are used to demonstrate the applicability of the proposed method. The paper ends with a section of concluding remarks and discussion.

2. Related work

In this section we present an overview of a number of online classification algorithms. Online learning is the most common approach to handle population drift and learning from very large and even redundant datasets. In online learning, model parameters are updated at each time-step using information only from the current example. Therefore, adaptation in time-varying environments is enabled by completely disregarding all past information. An exhaustive review of the numerous online classification algorithms is beyond the scope of this work. We discuss recently proposed methods that are employed in the empirical evaluation of the proposed approach.

The perceptron [25] is perhaps the simplest online binary linear classifier. It employs a classification function of the form $\hat{y}(t)=\mathrm{sign}(\beta(t)^\top z(t))$ where $\beta(t)$ is the parameter vector, $x(t)\in\mathbb{R}^d$ is the feature vector observed at time t, and $z(t)=[1,x_1(t),\ldots,x_d(t)]^\top$ is the augmented feature vector. Note that in the description of perceptron-based algorithms the class labels are $y(t)=\{-1,+1\}$. The parameter vector is updated only in the case of misclassification, using gradient descent, $\beta(t+1)=\beta(t)+\rho y(t)z(t)$, where the step-size ρ is also known as the learning rate.

The passive-aggressive (PA) algorithm is a modification of the perceptron algorithm that requires predictions, $\hat{y}(t)$, to be made with high confidence [26]. The degree of confidence in a prediction is measured by the magnitude $|w(t)^\top z(t)|$. To this end a hinge-loss function is defined that penalises both wrong predictions and predictions with low confidence. The update of $w(t)$ in the PA algorithm arises as the closed form solution to a constrained optimisation problem [26]. The perceptron and the PA algorithm are designed for problems in which the classes are linearly separable and ignore the possibility of noise-corrupted feature vectors or labels. To cope with noise two variants of the PA algorithm, called PA I and PA II, were proposed in [26]. Both algorithms employ an aggressiveness parameter, C. Higher values of C render the update step more aggressive, whereas, low values of C are appropriate in the presence of noise, or when the classes overlap. The analysis of PA I and PA II in [26] shows that for any sequence of examples, these algorithms cannot do much worse than the best fixed predictor chosen in hindsight.

Online kernel-based classification algorithms observe examples sequentially and store selected examples in their internal memory. The classification function is then defined by a kernel-dependent combination of the stored examples. A limitation of this approach is that an additional example needs to be stored after each prediction mistake. Thus, the number of examples

grows unboundedly unless the data sequence is perfectly predictable. Online kernel-based methods with a storage budget were first proposed in [27,28]. The Forgetron family of classification algorithms [29] constitutes the first online kernel-based classifiers restricted to a fixed budget of stored examples for which a rigorous mistake bound has been proven. The name Forgetron is used because these methods are variations of the kernel-based version of the perceptron algorithm which forget stored examples as necessary.

In [30] two kernel-based variants of the perceptron algorithm are proposed for time-varying environments. Shifting performance bounds are provided for these algorithms that ensure good performance on any data sequence that is well predicted by a sequence of classifiers whose parameters may change with time under certain constraints. The first algorithm, called the shifting perceptron algorithm (SPA), utilises a decaying factor that determines the rate of weight decay. When a mistake is made SPA scales down the old weight vector, to diminish the importance of earlier updates, before incorporating the new feature vector, as in the perceptron update rule. The decaying factor is not connected to the loss function, or the magnitude of the error, but only to the number of misclassifications. The number of examples stored by SPA is not bounded, which can lead to prohibitive computational difficulties when a non-linear kernel is used. In a streaming data application the decaying factor will tend to zero rendering the algorithm equivalent to the standard perceptron. This can be avoided by setting the decaying factor to a constant which yields an exponential decaying scheme similar to that of Forgetron and the proposed λ-perceptron algorithms. No mechanism is provided to adjust the value of the decaying factor to account for changes in the data-generating mechanism.

The randomised budget perceptron (RBP) algorithm [30] combines shifting with a budget of stored examples. As in the perceptron algorithm, each example on which the algorithm makes a mistake is stored. To avoid exceeding the budget, before adding a new example the algorithm discards randomly one of the stored examples. No scaling down of the contribution of past examples is involved. RBP is shown to strike the optimal trade-off between the largest norm of a classifier in the comparison sequence and the available budget [30]. The least recent budget perceptron (LBP) is a variant of RBP which removes the oldest example whenever the budget is exceeded. LBP can be regarded as an aggressive variant of Forgetron as it does not scale down the contribution of stored examples. The size of the budget is critical to the performance of RBP and LBP, as shown in [30], and no mechanism is proposed to adapt this parameter. Such a mechanism would be beneficial in a streaming environment. As in the case of SPA the rate of adaptation is not connected to the magnitude of the error but to the number of misclassifications.

Two budget variants of the second-order perceptron [31], are considered in [30]. These are the randomised budget second-order perceptron (RBSOP) and the least recent second-order perceptron (LBSOP). In RBSOP the example to be discarded is chosen at random among the current set of stored examples, while in LBSOP the oldest example is always removed.

In [32] the view that the learning rate of an online classifier, e.g. ρ in the perceptron algorithm, can be viewed as a 'forgetting mechanism' is advocated. A large learning rate magnifies the impact of the current example on the update of the estimated parameters, and vice versa. Based on this view a framework to handle population drift is proposed that adapts the learning rate of online classifiers as a function of a running estimate of the classification error rate. The assumption behind this framework is that an increasing error rate signifies the onset of a change in the environment. Adaptive learning rate versions of the perceptron and the Winnow [33] classifiers are proposed in [32]. Moreover, an adaptive and online version of the standard linear discriminant classifier (OLDC) is developed [32].

Sigmoid perceptrons constitute the building blocks for more complex non-linear classifiers like multilayer perceptrons. The parameters of sigmoid perceptrons are estimated through iterative minimisation of an error function, the most frequent choice of which is the sum of squared errors. The algorithm of choice for online training of sigmoid perceptrons is stochastic gradient descent [34]. A central issue in stochastic gradient descent is how to adaptively set the learning rate to achieve rapid convergence without compromising the ability to adapt to population drift. The idea of utilising the gradient of the error function with respect to parameters, e.g. the learning rate, of the update rule for the parameter vector is at the core of several stochastic adaptation methods [34,35]. Stochastic meta-descent (SMD) utilises this gradient information to adapt the local learning rates (i.e. a separate step-size for each element of the parameter vector) [34], through exponentiated gradient (meta-)descent. This approach is similar to the approach we develop in the next section in that a gradient descent scheme is employed to adaptively tune parameters of the update rule that is applied to the parameter vector. The two approaches differ in that we do not employ information from the current time-step only, and we adapt the forgetting factor instead of the learning rate.

In the context of prediction with expert advice, discounted regret has been proposed as an alternative measure which relies on the assumption that losses in the past are less significant than recently suffered losses. Similar to the forgetting framework we develop, this measure assigns a weight to the regret at each time-step that is a decreasing function of its distance in the past. In [14] results are provided which show that when the discount factors decrease sufficiently slowly, it is possible to make the average discounted regret vanish when the number of time-steps is large.

3. Framework

We borrow ideas from adaptive filter theory to develop an adaptive online sigmoid perceptron algorithm. Our approach is based on the definition of a criterion that enables the classifier to adapt to changes in the environment, without completely disregarding all previous knowledge. Typical criteria used to estimate the parameters of a classifier assign equal weight to each example irrespective of the time it is observed. A widely used criterion to estimate the parameters of sigmoid perceptrons is the sum of squared errors:

$$\mathcal{E}(\beta,t)=\sum_{i=1}^{t} e(\beta,i)=\frac{1}{2}\sum_{i=1}^{t}(y(i)-\varphi(\beta^{\top}z(i)))^2, \tag{1}$$

where $y(t)\in\{0,1\}$ and $\varphi(\beta^{\top}z(i))=e^{\beta^{\top}z}/(1+e^{\beta^{\top}z})$.

In the presence of population drift a reasonable assumption, applicable to any criterion, is that the impact of each example on the parameter estimates should be related to the time of observation. More recent examples are expected to be more informative about the characteristics of the current populations than examples in the distant past. This line of reasoning leads naturally to the introduction of weights in the definition of the optimisation criterion. At present we adopt the cumulative error function formulation of the recursive least squares (RLS) adaptive filter [22,23,36], although this idea can be also applied to general likelihood functions. As in the RLS filter we introduce an exponential weighting factor that is a function of time in the cumulative error function:

$$\mathcal{E}(\beta,t)=\sum_{i=1}^{t}\lambda^{t-i}e(\beta,i)=e(\beta,t)+\lambda\mathcal{E}(\beta,t-1). \tag{2}$$

In Eq. (2) the weighting factor, $\lambda \in [0,1]$, is the so-called *forgetting factor*. The forgetting factor discounts the impact of past examples in order to enable the classifier to adapt in response to population drift [23]. As λ tends to unity past and present examples become equally weighted. In contrast, lower values of λ increase the impact of recent examples on the estimated parameters. We refer to sigmoid perceptrons whose parameters are estimated through Eq. (2) as *λ-perceptrons*.

The sum of the exponential weighting factors is referred to as the *effective window width* [36]. The effective window width at time-step t for constant λ is $(1-\lambda^t)/(1-\lambda)$. For variable λ, the weight at time-step t assigned to $e(\beta,i)$ is $\prod_{j=i+1}^{t}\lambda(j)$. The sum of these weights can be recursively updated according to:

$$n(t)=1+\sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t}\lambda(j)\right)=1+\lambda(t)n(t-1),\quad t=2,3\ldots\quad n(1)=1. \tag{3}$$

A motivating example for the introduction of exponential weighting with respect to time in the definition of the cumulative error is provided in Fig. 1. We consider two, two dimensional binary classification problems, in which the distribution of the feature vectors for both classes $y \in \{0,1\}$ is Gaussian:

$$(x(t)|y)\sim\mathcal{N}(\mu_y(t),I),\quad P(y=0)=P(y=1)=\tfrac{1}{2},\ y\in\{0,1\}, \tag{4}$$

where I is the identity matrix. In the first problem, $\mu_0(t)=[3,3]$, and $\mu_1(t)=[-3,-3]$. In the second problem the class labels are reversed, i.e. $\mu_0(t)=[-3,-3]$ and $\mu_1(t)=[3,3]$. By construction, the optimal parameter values in the second problem are opposite in sign from the optimal values for the first problem.

A dataset that exhibits abrupt change is constructed by drawing examples from the first classification problem during the first 15 time-steps and from the second problem during the final 15 time-steps. An ideal cumulative error function for this task would have the same minimisers as the error function for the first problem during the first 15 examples, and the same minimisers as the error function for the second problem during the final 15 examples. A classifier estimated using this error function would adapt to the abrupt change instantly under the assumption that the global minimiser of the error function is correctly identified at each time-step.

Fig. 1 shows three dimensional mesh plots of the cumulative error function of Eq. (2), after the presentation of all 30 examples for two values of λ. In the case of no forgetting, depicted in Fig. 1(a), the global minimiser of the cumulative error function is at the origin. Introducing forgetting with $\lambda=0.8$, Fig. 1(b), yields a cumulative error function whose minimisers are in the same region as those of the cumulative error function without forgetting that corresponds to the second classification problem. Therefore, exponentially weighting the previous errors enables the classifier to adapt to population drift, by affecting the location of the global minimisers of the cumulative error function. The same conclusion can be drawn by comparing the average error rate achieved by classifiers whose parameters are estimated using the cumulative error function with forgetting, Eq. (2), for different values of λ. Note that this average error rate is measured at each time-step by computing the classification error over an independent set of 100 examples randomly generated according to the current class definition, and then averaging over the 30 time-steps. The average error rate for $\lambda=1$, which amounts to no forgetting, is 0.375, for $\lambda=0.95$ it reduces to 0.274, and is further reduced to 0.195 and 0.157 for $\lambda=0.9$ and 0.8, respectively.

3.1. Adaptation of parameter vector

Using the cumulative error function of Eq. (2), the jth element, $\beta_j(t)$, of the parameter vector β can be updated through gradient descent:

$$\beta_j(t+1)=\beta_j(t)-\eta\frac{\partial\mathcal{E}(\beta(t),t)}{\partial\beta_j(t)}, \tag{5}$$

where η is a positive constant called the *step-size* parameter or the *learning rate*, and the partial derivative of the cumulative error $\mathcal{E}(\beta(t),t)$ is:

$$\frac{\partial\mathcal{E}(\beta(t),t)}{\partial\beta_j(t)}=\sum_{i=1}^{t}\lambda^{t-i}\frac{\partial e(\beta(t),i)}{\partial\beta_j(t)}, \tag{6}$$

with:

$$\begin{aligned}\frac{\partial e(\beta(t),i)}{\partial\beta_j(t)}&=-[y(i)-\varphi(\beta(t)^\top z(i))]\frac{\partial\varphi(\beta(t)^\top z(i))}{\partial\beta_j(t)}\\&=-z_j(i)[y(i)-\varphi(\beta(t)^\top z(i))]\varphi(\beta(t)^\top z(i))[1-\varphi(\beta(t)^\top z(i))],\end{aligned} \tag{7}$$

for $j=0,1,\ldots,d$.

Computing the partial derivative $\partial\mathcal{E}(\beta(t),t)/\partial\beta_j(t)$ requires storing and processing all previous examples, $D_t=\{(x(i),y(i))\}_{i=1}^{t}$, which is prohibitive in streaming data applications. For such applications, we propose to update the parameter vector after the presentation of each example using the exponentially weighted sum of the previous realisations of the gradient. With this

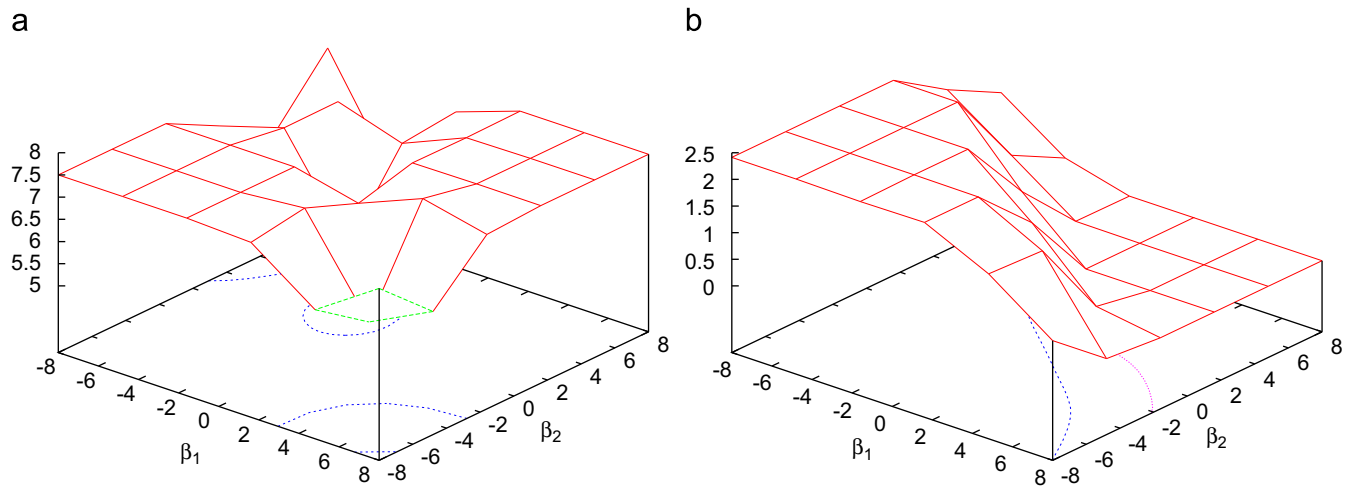

Fig. 1. Cumulative error functions for different values of the forgetting factor: (a) no forgetting $\lambda=1$ and (b) forgetting $\lambda=0.8$.

modification, Eq. (5) can be rewritten as:

$$\beta_j(t+1)=\beta_j(t)-\eta\sum_{i=1}^{t}\lambda^{t-i}\frac{\partial e(\beta(i),i)}{\partial\beta_j(i)}. \tag{8}$$

For notational convenience we define:

$$G_{\beta_j}(t)=\sum_{i=1}^{t}\lambda^{t-i}\frac{\partial e(\beta(i),i)}{\partial\beta_j(i)}=\frac{\partial e(\beta(t),t)}{\partial\beta_j(t)}+\lambda G_{\beta_j}(t-1). \tag{9}$$

In the neural network literature, the update formula of Eq. (8) is known as the online *backpropagation with momentum* algorithm [24]. The motivation for including the sum of exponentially weighted previous values of the gradient, $\eta\sum_{i=1}^{t-1}\lambda^{t-i}\partial e(\beta(i),i)/\partial\beta_j(i)$, which is known as the *momentum term*, in the parameter update equation is to accelerate the gradient descent method [23,24,37,38]. This simple modification renders the step-size with respect to each element of β adaptive. Whenever the partial derivative with respect to the jth element of the parameter vector, $\partial e(\beta(t),t)/\partial\beta_j(t)$, has the same sign on consecutive time-steps, the sum in Eq. (8) grows in magnitude, accelerating the descent in downhill directions. In contrast, when $\partial e(\beta(t),t)/\partial\beta_j(t)$ has opposite signs in consecutive time-steps the sum shrinks in magnitude, creating a stabilising effect in directions that oscillate in sign [39]. Finally, the momentum term can also prevent the learning process from terminating in a shallow local minimum on the error function [24]. In [40] a deterministic proof of convergence is derived for the online backpropagation with momentum algorithm. In more detail, it is shown that under certain natural assumptions, every accumulation point of the update formula of Eq. (8) is a stationary point of the cumulative error function of Eq. (1) for finite t [40].

3.2. Adaptation of forgetting

The choice of the forgetting factor is critical to classification performance. The appropriate choice of λ depends on the dynamic character of a data stream, which is typically unknown a priori. Moreover, setting the degree of forgetting to a constant seems inappropriate in the presence of abrupt changes, or when the speed of population drift varies over time. It is therefore desirable to have an online adaptive scheme to tune λ. In adaptive filtering, stochastic gradient descent has been proposed to adapt online the step-size of the Least Mean Squares (LMS) algorithm, and the forgetting factor of the RLS algorithm [22]. We adopt this approach to adapt λ towards values that minimise the error at time-step $(t+1)$ [22,23].

To obtain an expression for the gradient with respect to λ, we express the error as a composite function of λ and apply the chain rule:

$$\frac{\partial e(\beta(t+1),t+1)}{\partial\lambda}=\sum_{j=0}^{d}\frac{\partial e(\beta(t+1),t+1)}{\partial\beta_j(t+1)}\frac{\partial\beta_j(t+1)}{\partial\lambda}. \tag{10}$$

An expression for $\partial\beta_j(t+1)/\partial\lambda$ is obtained by differentiating Eq. (5) with respect to λ:

$$\frac{\partial\beta_j(t+1)}{\partial\lambda}=\frac{\partial\beta_j(t)}{\partial\lambda}-\eta\frac{\partial^2\mathcal{E}(\beta(t),t)}{\partial\lambda\partial\beta_j(t)}. \tag{11}$$

We obtain $\partial^2\mathcal{E}(\beta(t),t)/\partial\lambda\partial\beta_j(t)$ by differentiating Eq. (6):

$$\begin{aligned}\frac{\partial^2\mathcal{E}(\beta(t),t)}{\partial\lambda\partial\beta_j(t)}&=\sum_{i=1}^{t}\left\{(t-i)\lambda^{t-i-1}\frac{\partial e(\beta(t),i)}{\partial\beta_j(t)}+\lambda^{t-i}\frac{\partial^2 e(\beta(t),i)}{\partial\lambda\partial\beta_j(t)}\right\}\\&=\sum_{i=1}^{t-1}\left\{(t-i)\lambda^{t-i-1}\frac{\partial e(\beta(t),i)}{\partial\beta_j(t)}\right\}+\sum_{i=1}^{t}\left\{\lambda^{t-i}\frac{\partial^2 e(\beta(t),i)}{\partial\lambda\partial\beta_j(t)}\right\},\end{aligned} \tag{12}$$

adding and subtracting $\sum_{i=1}^{t-1}\lambda^{t-i-1}\partial e(\beta(t),i)/\partial\beta_j(t)$ from the right-hand side yields:

$$\begin{aligned}\frac{\partial^2\mathcal{E}(\beta(t),t)}{\partial\lambda\partial\beta_j(t)}&=\sum_{i=1}^{t-1}\left\{(t-i-1)\lambda^{t-i-1}\frac{\partial e(\beta(t),i)}{\partial\beta_j(t)}\right\}\\&\quad+\sum_{i=1}^{t-1}\left\{\lambda^{t-i-1}\frac{\partial e(\beta(t),i)}{\partial\beta_j(t)}\right\}+\sum_{i=1}^{t}\left\{\lambda^{t-i}\frac{\partial^2 e(\beta(t),i)}{\partial\lambda\partial\beta_j(t)}\right\}\\&=\sum_{i=1}^{t-1}\left\{(t-i-1)\lambda^{t-i-1}\frac{\partial e(\beta(t),i)}{\partial\beta_j(t)}\right\}+\sum_{i=1}^{t-1}\left\{\lambda^{t-i-1}\frac{\partial e(\beta(t),i)}{\partial\beta_j(t)}\right\}\\&\quad+\frac{\partial^2 e(\beta(t),t)}{\partial\lambda\partial\beta_j(t)}+\lambda\sum_{i=1}^{t-1}\left\{\lambda^{t-i-1}\frac{\partial^2 e(\beta(t),i)}{\partial\lambda\partial\beta_j(t)}\right\}\\&=\frac{\partial^2 e(\beta(t),t)}{\partial\lambda\partial\beta_j(t)}+\sum_{i=1}^{t-1}\left\{\lambda^{t-i-1}\frac{\partial e(\beta(t),i)}{\partial\beta_j(t)}\right\}\\&\quad+\lambda\left\{\sum_{i=1}^{t-1}\left\{(t-i-1)\lambda^{t-i-2}\frac{\partial e(\beta(t),i)}{\partial\beta_j(t)}\right\}\right.\\&\quad\left.+\sum_{i=1}^{t-1}\left\{\lambda^{t-i-1}\frac{\partial^2 e(\beta(t),i)}{\partial\lambda\partial\beta_j(t)}\right\}\right\}.\end{aligned} \tag{13}$$

Inspection of Eq. (13) reveals that the last two terms are actually $\partial^2\mathcal{E}(\beta(t),t-1)/\partial\lambda\partial\beta_j(t)$, and the second term is $\partial\mathcal{E}(\beta(t),t-1)/\partial\beta_j(t)$, which leads to:

$$\begin{aligned}\frac{\partial^2\mathcal{E}(\beta(t),t)}{\partial\lambda\partial\beta_j(t)}&=\frac{\partial^2 e(\beta(t),t)}{\partial\lambda\partial\beta_j(t)}+\frac{\partial\mathcal{E}(\beta(t),t-1)}{\partial\beta_j(t)}+\lambda\frac{\partial^2\mathcal{E}(\beta(t),t-1)}{\partial\lambda\partial\beta_j(t)}\\&=\sum_{i=1}^{t}\lambda^{t-i}\left(\frac{\partial^2 e(\beta(t),i)}{\partial\lambda\partial\beta_j(t)}+\frac{\partial\mathcal{E}(\beta(t),i-1)}{\partial\beta_j(t)}\right).\end{aligned} \tag{14}$$

Note that:

$$\frac{\partial^2 e(\beta(t),i)}{\partial\lambda\partial\beta_j(t)}=\sum_{k=0}^{d}\frac{\partial^2 e(\beta(t),i)}{\partial\beta_k(t)\partial\beta_j(t)}\frac{\partial\beta_k(t)}{\partial\lambda}.$$

Substituting Eqs. (11) and (14) in Eq. (10) yields a recursive formula for the computation of the derivative of the error at time $(t+1)$ with respect to the forgetting factor:

$$\begin{aligned}\frac{\partial e(\beta(t+1),t+1)}{\partial\lambda}&=\sum_{j=0}^{d}\frac{\partial e(\beta(t+1),t+1)}{\partial\beta_j(t+1)}\left\{\frac{\partial\beta_j(t)}{\partial\lambda}\right.\\&\quad\left.-\eta\left\{\frac{\partial^2 e(\beta(t),t)}{\partial\lambda\partial\beta_j(t)}+\frac{\partial\mathcal{E}(\beta(t),t-1)}{\partial\beta_j(t)}+\lambda\frac{\partial^2\mathcal{E}(\beta(t),t-1)}{\partial\lambda\partial\beta_j(t)}\right\}\right\}.\end{aligned} \tag{15}$$

The computation of $\partial\mathcal{E}(\beta(t),t-1)/\partial\beta_j(t)$ and $\partial^2\mathcal{E}(\beta(t),t-1)/\partial\lambda\partial\beta_j(t)$ requires storing and processing all the examples up to time t. Therefore, $\partial^2\mathcal{E}(\beta(t),t)/\partial\lambda\partial\beta_j(t)$ and $\partial\beta_j(t+1)/\partial\lambda$ cannot be employed by an online algorithm. To develop a completely online algorithm suitable to streaming applications we propose to use $G_\lambda\beta_j(t+1)$ and $B_j(t)$ described below which can be updated at each time-step without storing previous examples:

$$G_\lambda\beta_j(t+1)=G_\lambda\beta_j(t)-\eta B_j(t), \tag{16}$$

$$\begin{aligned}B_j(t)&=\frac{\partial^2 e(\beta(t),t)}{\partial\lambda\partial\beta_j(t)}+G_{\beta_j}(t-1)+\lambda B_j(t-1)\\&=\sum_{i=1}^{t}\lambda^{t-i}\left(\frac{\partial^2 e(\beta(i),i)}{\partial\lambda\partial\beta_j(i)}+G_{\beta_j}(i-1)\right),\end{aligned} \tag{17}$$

where $G_{\beta_j}(\cdot)$, defined in Eq. (9), is used in the update of $\beta(t)$. We have noticed empirically that the trace of $G_\lambda\beta_j(t+1)$ can diverge if η and λ are chosen too high. The stability of the algorithm is significantly improved by multiplying $B_j(t)$ in Eq. (16) by $\eta(1-\lambda)$, or $\eta/n(t)$, instead of η. This modification does not compromise the

rapid convergence of the algorithm and in Algorithm 1 we propose to multiply $B_j(t)$ by $\eta/n(t)$. Note that no scaling is involved in the update of the parameters $\beta_j(t+1)$. Extensive empirical evidence presented in Section 4 suggests that the quantity:

$$G_\lambda e(t+1) = \sum_{j=0}^{d} \frac{\partial e(\beta(t+1),t+1)}{\partial \beta_j(t+1)} G_\lambda \beta_j(t+1), \tag{18}$$

conveys significant information about the direction in which an adaptation of λ will improve performance.

3.3. Proposed algorithm

The online algorithm we propose, the *adaptive* λ-perceptron, summarised in Algorithm 1, is a two-step procedure for adapting the parameter vector, β, and the forgetting factor, λ, after the presentation of each example. An efficient $\mathcal{O}(d)$ algorithm permits the computation of the product of the Hessian matrix with a vector without having to compute or store the Hessian [41]. Therefore, the storage and time complexity of the proposed algorithm are both $\mathcal{O}(d)$.

Algorithm 1. Adaptive λ-Perceptron.

Require: $\eta > 0, \alpha > 0, \lambda(1) \in [0,1), G_\lambda^- < 0, G_\lambda^+ > 0, \lambda^+ \in (\lambda^-, 1)$, $\lambda^- \in [0, \lambda^+)$
Set: $\beta_j(1) = 0, G_{\beta_j}(0) = 0, G_\lambda \beta_j(1) = 0, \forall j = 0, \ldots, d$, $B(0) = 0$, $n(0) = 0$
  for $t = 1,2,\ldots$ **do**
    // **Classify**
    Predict class label of $x(t)$
    Receive true class label $y(t)$

    $n(t) = 1 + \lambda(t) n(t-1)$

    // **Update** $\beta_j(t)$
    for $j = 0,1,\ldots,d$ **do**
      $G_{\beta_j}(t) = \frac{\partial e(\beta(t),t)}{\partial \beta_j(t)} + \lambda G_{\beta_j}(t-1)$
      $\beta_j(t+1) = \beta_j(t) - \eta G_{\beta_j}(t)$
    end for

    // **Update** $\lambda(t)$
    $G_\lambda e(t) = \sum_{j=0}^{d} \frac{\partial e(\beta(t),t)}{\partial \beta_j(t)} G_\lambda \beta_j(t)$
    if $G_\lambda e(t) > G_\lambda^+$ or $G_\lambda e(t) < G_\lambda^-$ **then**
      $\lambda(t+1) = [\lambda(t) - \alpha \operatorname{sign}(G_\lambda e(t))]_{\lambda^-}^{\lambda^+}$
    else
      $\lambda(t+1) = \lambda(t)$
    end if

    //**Compute** $G_\lambda \beta_j(t+1)$
    for $j = 0,1,\ldots,d$ **do**
      $B_j(t) = \frac{\partial^2 e(\beta(t),t)}{\partial \lambda \partial \beta_j(t)} + G_{\beta_j}(t-1) + \lambda B_j(t-1)$
      $G_\lambda \beta_j(t+1) = G_\lambda \beta_j(t) - \frac{\eta}{n(t)} B_j(t)$,
    end for
  end for

We propose to adapt $\lambda(t)$ using information only from the sign of $G_\lambda e(t)$ and not its magnitude:

$$\lambda(t+1) = [\lambda(t) - \alpha \operatorname{sign}(G_\lambda e(t))]_{\lambda^-}^{\lambda^+}. \tag{19}$$

Eq. (19) is a heuristic rule to update $\lambda(t)$ online that relies on the sign of $G_\lambda e(t)$ and employs truncation values. Sign-based gradient descent schemes have been shown to be eminently suitable in applications with imprecise or noise-corrupted function and gradient values [42,43], and have frequently been used to adapt step-size parameters in neural network training [38,44,45]. It will be shown in the following section that $G_\lambda e(t)$ exhibits noisy behaviour. The bracket followed by λ^- and λ^+ in Eq. (19) indicates truncation. Truncation is also employed in the variable step-size LMS algorithm, and the RLS algorithm with adaptive forgetting [23]. In the adaptive λ-perceptron algorithm the upper truncation value, λ^+, is particularly important. Values of λ close to unity cause a steady increase in the magnitude of $G_\lambda e(t)$ which renders the behaviour of the adaptive scheme unstable (i.e. wide fluctuations of $\lambda(t)$). The value of the lower threshold is not as important. In the experimental results we use $\lambda^+ = 0.88$, and $\lambda^- = 0.1$. In the adaptive λ-perceptron algorithm $\lambda(t)$ is updated if $G_\lambda e(t)$ exceeds $G_\lambda^+ > 0$, or is below $G_\lambda^- < 0$. These thresholds are imposed to further avoid oscillations of λ due to the noise in $G_\lambda e(t)$. A sensitivity analysis with respect to all the parameters of the adaptive algorithm is presented in Section 4.1.

4. Experimental results

To investigate the behaviour of the proposed method we employ it first on data generated using controlled simulation settings. In all the experiments with artificial datasets the distribution of the feature vectors for both classes $y \in \{0,1\}$ is Gaussian with:

$$(x(t)|y) \sim \mathcal{N}(\mu_y(t), \Sigma_y(t)), \quad P(y=0) = P(y=1) = \tfrac{1}{2}, \ y \in \{0,1\}, \tag{20}$$

where $\mu_y \in \mathbb{R}^d$, and $\Sigma_y \in \mathbb{R}^{d \times d}$ is a random covariance matrix. The mean vectors for both classes are initialised uniformly in $[-2,2]^d$. Note that this definition of the classes introduces model misspecification since only in the unlikely event that the two covariance matrices Σ_y are equal will the optimal decision boundary be linear. In all other cases a quadratic discrimination rule is optimal.

Gradually drifting datasets are created by having the mean vector of each conditional probability density function, $\mu_y(t)$, follow a damped random walk, while the covariance matrix, $\Sigma_y(t)$ is updated through a convex combination:

$$\mu_y(t+1) = \theta \mu_y(t) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 I) \quad \text{and} \quad \theta \in (0.9, 1),$$

$$\Sigma_y(t+1) = \frac{T-t}{T} \Sigma_y^0 + \frac{t}{T} \Sigma_y^T,$$

where T denotes the size of the dataset, Σ_y^0, Σ_y^T are two random covariance matrices, and σ^2 controls the speed of drift, with larger values corresponding to more rapid drift. By setting to random values $\mu_y(t)$, and $\Sigma_y(t)$ for $y \in \{0,1\}$ abrupt changes are generated every p time-steps after the first 250 time-steps. Artificial datasets consist of 10,000 examples, and all the reported results are averages over 100 simulations. The error rate at each time-step is computed over an independent set of 100 examples randomly generated according to the current class definitions.

4.1. Sensitivity analysis

In this subsection we perform a sensitivity analysis of the proposed method. We first explore the relationship between the learning rate of the stochastic gradient descent scheme, η, and the forgetting factor, λ. Fig. 2 illustrates the average error rate achieved by different constant forgetting λ-perceptrons on artificial datasets exhibiting gradual and abrupt drift. The figure shows that irrespective of the type of drift the optimal constant

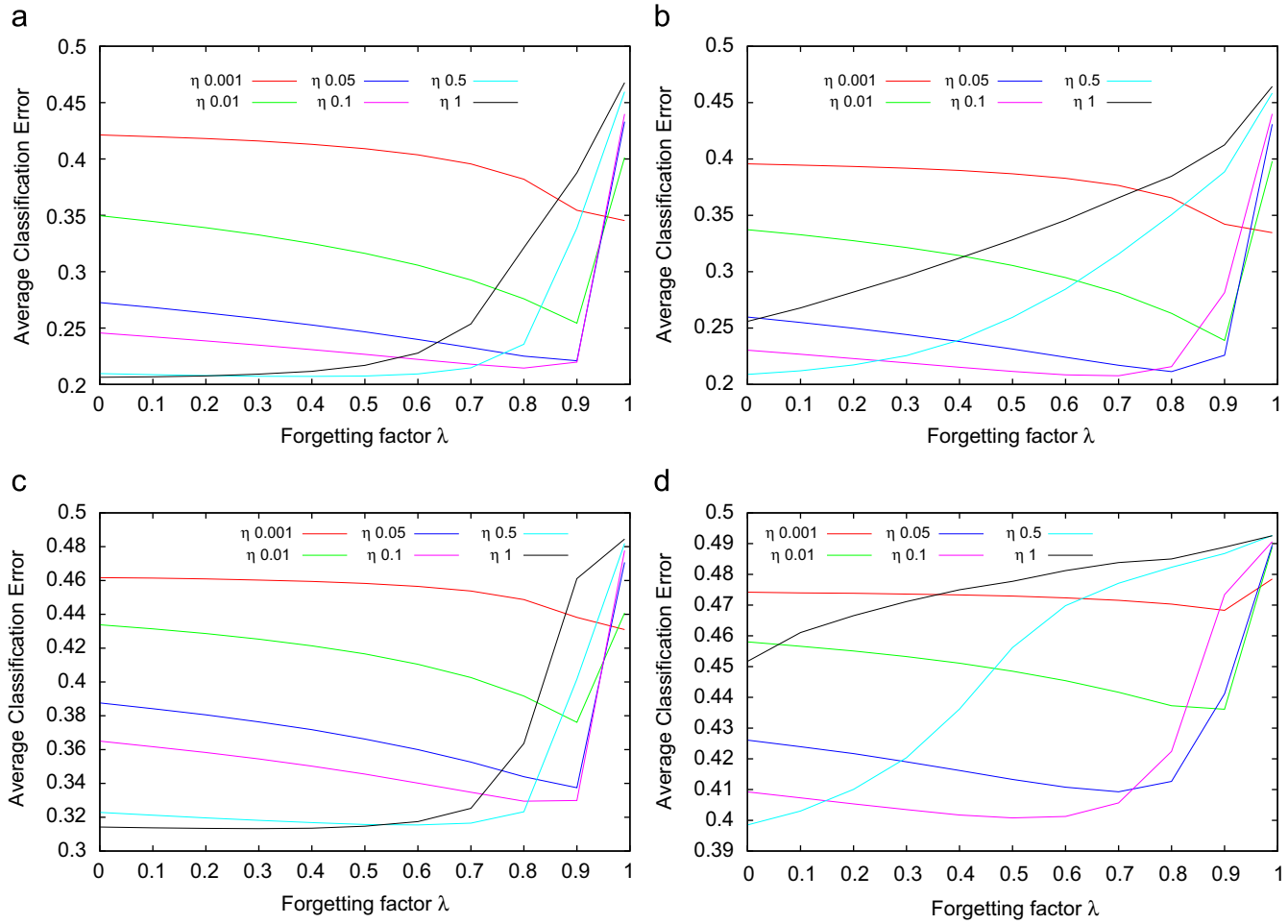

Fig. 2. Average error rate of constant forgetting λ-perceptron s for different values of the step-size η: (a) abrupt changes $d=2$; (b) abrupt changes $d=20$; (c) gradual drift $d=2$; and (d) gradual drift $d=20$.

value of λ decreases as η increases, and for $\eta \geq 0.5$ the optimal λ is zero. This finding is not unexpected. For small values of η the change in $\beta(t)$ over consecutive time-steps is very slow and hence the sum of the exponentially weighted previous values of the gradient is a good approximation to the gradient of the cumulative error, $\mathcal{E}(t)$, with respect to the current parameter vector, $\beta(t)$. In contrast, as η increases the parameter vector changes rapidly in consecutive time-steps and therefore previous values of the gradient contain little, or misleading, information concerning the direction of descent at the current parameter estimates.

The evolution of the forgetting factor through the adaptive λ-perceptron algorithm for different values of η is depicted in Fig. 3, which shows that the adaptive forgetting factor behaves differently under different types of population drift. In the case of abrupt change, there is a distinct pattern in the evolution of λ after each change point, whereas in the case of gradual drift, λ appears to fluctuate randomly around a mean value that is related to η. We thoroughly discuss the behaviour of the adaptive forgetting factor under different types of population drift later. Fig. 3 also indicates that the adaptive forgetting scheme tends to decrease the mean value of λ as η increases for $\eta < 0.5$. For $\eta = 0.5{,}1$ this no longer holds and in these cases the behaviour of the forgetting factor also ceases to be informative about the underlying population drift process. The adaptive forgetting scheme is unable to adjust λ towards values that improve performance in these cases because as η increases the approximation of the derivative of the error at the current time-step with respect to λ,$\partial e(\beta(t),t)/\partial\lambda$, through $G_\lambda e(t)$, becomes poorer. An illustration of this phenomenon is provided in Fig. 4, where the evolution of $G_\lambda e(t)$ in an abruptly changing environment is shown for four values of η. The periodic pattern in $G_\lambda e(t)$ induced by abrupt changes every 500 time steps is becoming progressively less clear and for $\eta = 0.5$, $G_\lambda e(t)$ exhibits random oscillations of increasing magnitude.

We next investigate the sensitivity of the adaptive λ-perceptron algorithm to α, the step-size of the sign-based gradient descent scheme for λ, presented in Eq. (19). Fig. 5 depicts the average error rate achieved for different values of η and α. For brevity we depict only the results for environments that are subject to both gradual drift and abrupt changes. Fig. 5 shows that setting α around 5×10^{-3} is an appropriate choice. For slow drift, performance appears to be more sensitive to the choice of α than to the choice of η, whereas the opposite holds for rapid drift.

Finally, we investigate the sensitivity to the two thresholds G_λ^+ and G_λ^- that are employed by the adaptive forgetting scheme. Fig. 6 illustrates the average error rate achieved by the adaptive λ-perceptron algorithm on datasets that exhibit gradual drift. For smaller values of the step-size parameter the best performance is achieved by setting $|G_\lambda^+| > |G_\lambda^-|$. This setting renders the algorithm less prone to decrease the forgetting factor for small values of $G_\lambda e(t)$. For larger values of η the opposite setting, $|G_\lambda^+| < |G_\lambda^-|$, yields optimal performance. This is expected since, as Fig. 2 shows for $\eta = 0.5$, optimal performance is achieved by having $\lambda = 0$. However, in no case is it optimal to have the two thresholds equal in magnitude, or equal to zero (which corresponds to the standard sign-based gradient descent).

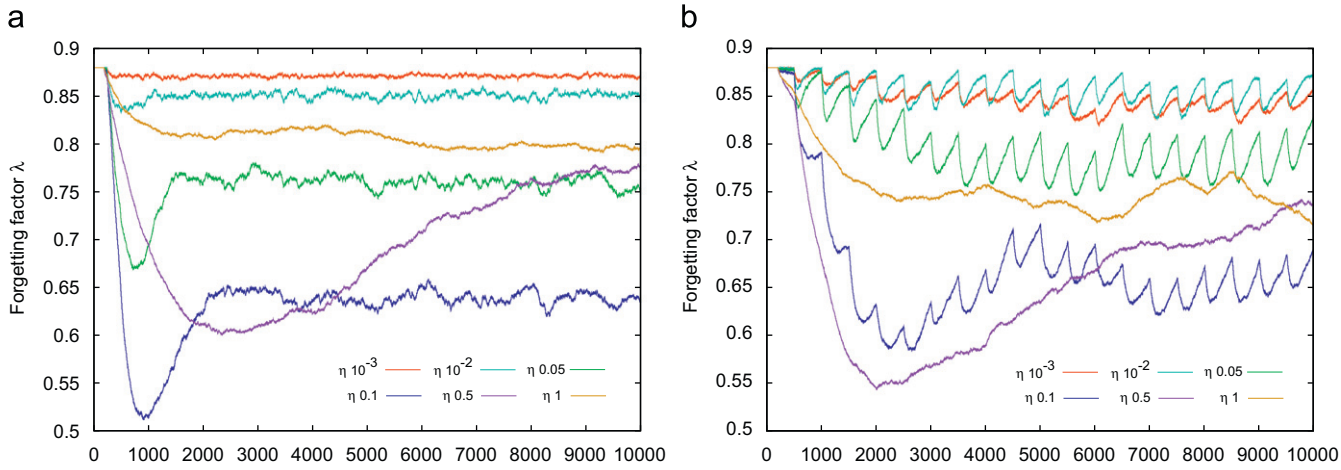

Fig. 3. Evolution of λ for different values of the step-size parameter: (a) gradual drift $d=20$, $\sigma=0.2$ and (b) abrupt changes $d=20$, $p=500$.

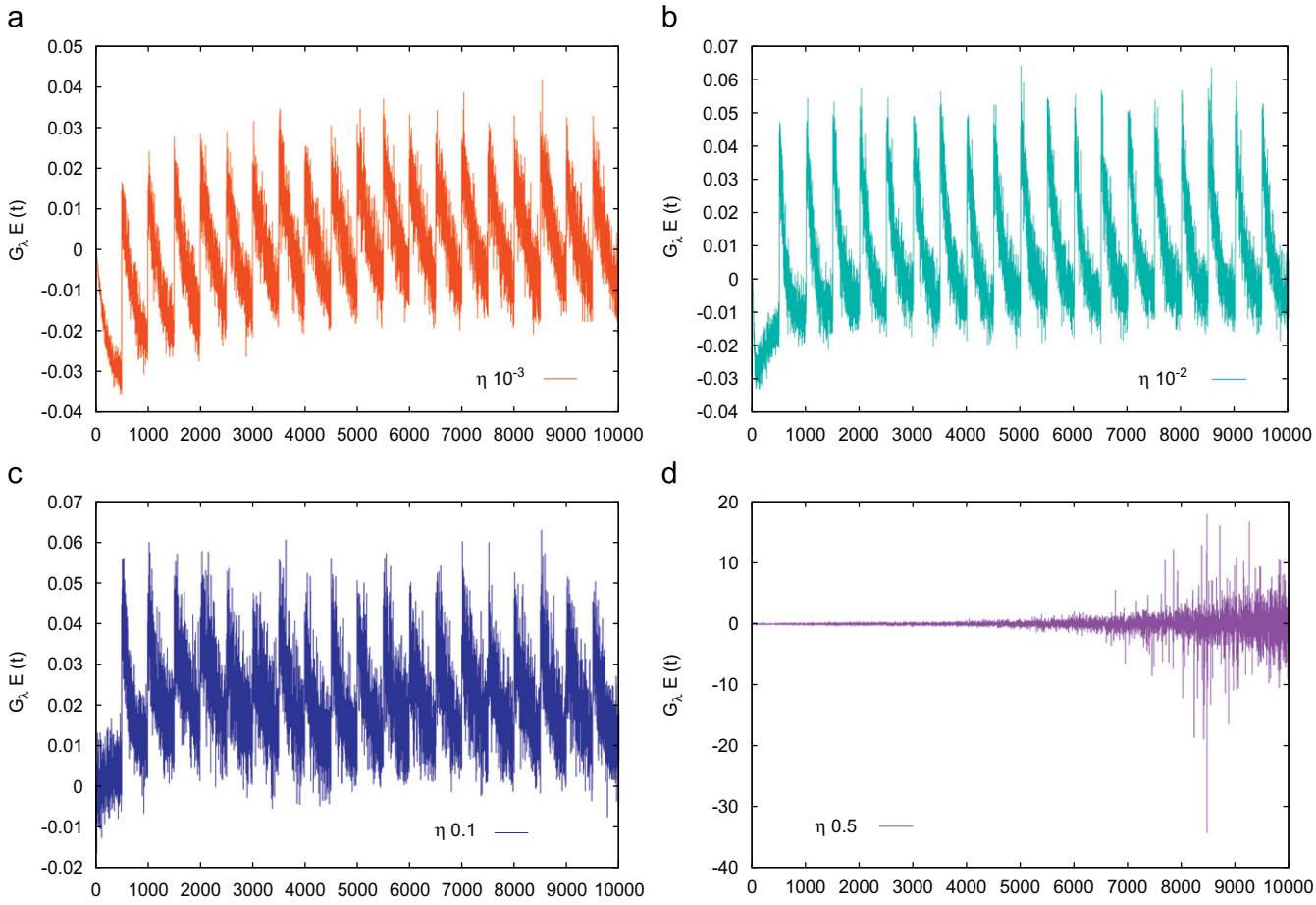

Fig. 4. Evolution of $G_\lambda e(t)$ for different values of η in abruptly changing environment with $p=500$ and $d=20$: (a) $\eta=10^{-3}$; (b) $\eta=10^{-2}$; (c) $\eta=10^{-1}$; and (d) $\eta=0.5$.

4.2. Artificial datasets

In this subsection we investigate the behaviour of the λ-perceptron algorithm on artificial datasets that exhibit abrupt and gradual population drift. We consider two values for the dimensionality of the feature vectors, $d=2$ and $d=20$. In all the experiments we compare the performance of the 17 algorithms listed in Table 1 in terms of the average error rate. The average error rate is computed as the average of the error rate over all the time-steps and all the simulations. In all the tables that report average error rates, statistically significant differences in comparison with the adaptive λ-perceptron algorithm at a 5% significance level are marked with a + or −. A + marks a statistically significant superior performance of the adaptive λ-perceptron against the other method, a − marks statistically significant inferior performance, and no sign marks indistinguishable results. We used an adaptation of Student's t-test intended for testing two samples having possibly unequal

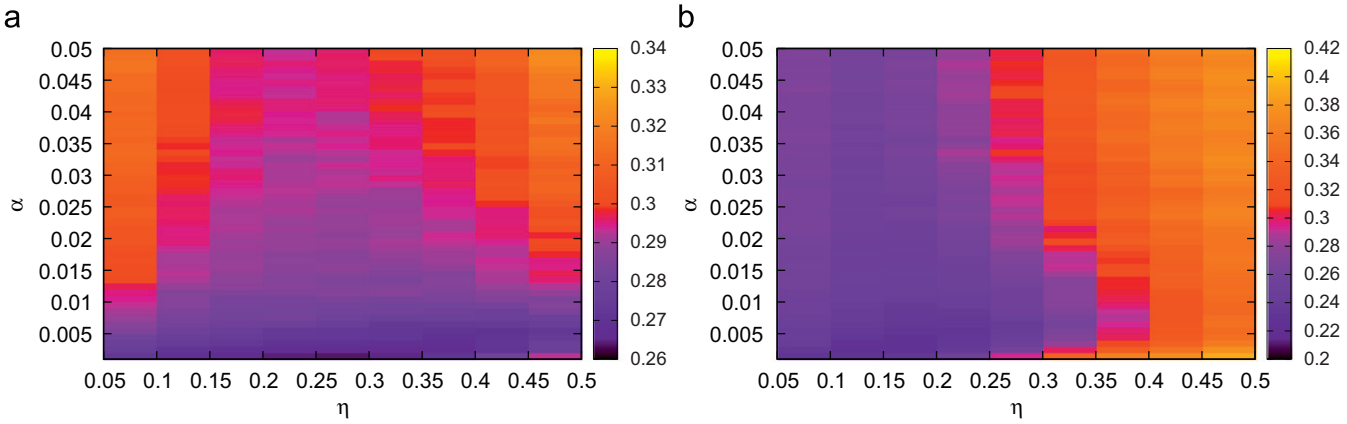

Fig. 5. Average error rate of the adaptive λ-perceptron algorithm for different values of the step-sizes η and α: (a) $\sigma=0.1$ and (b) $\sigma=0.2$.

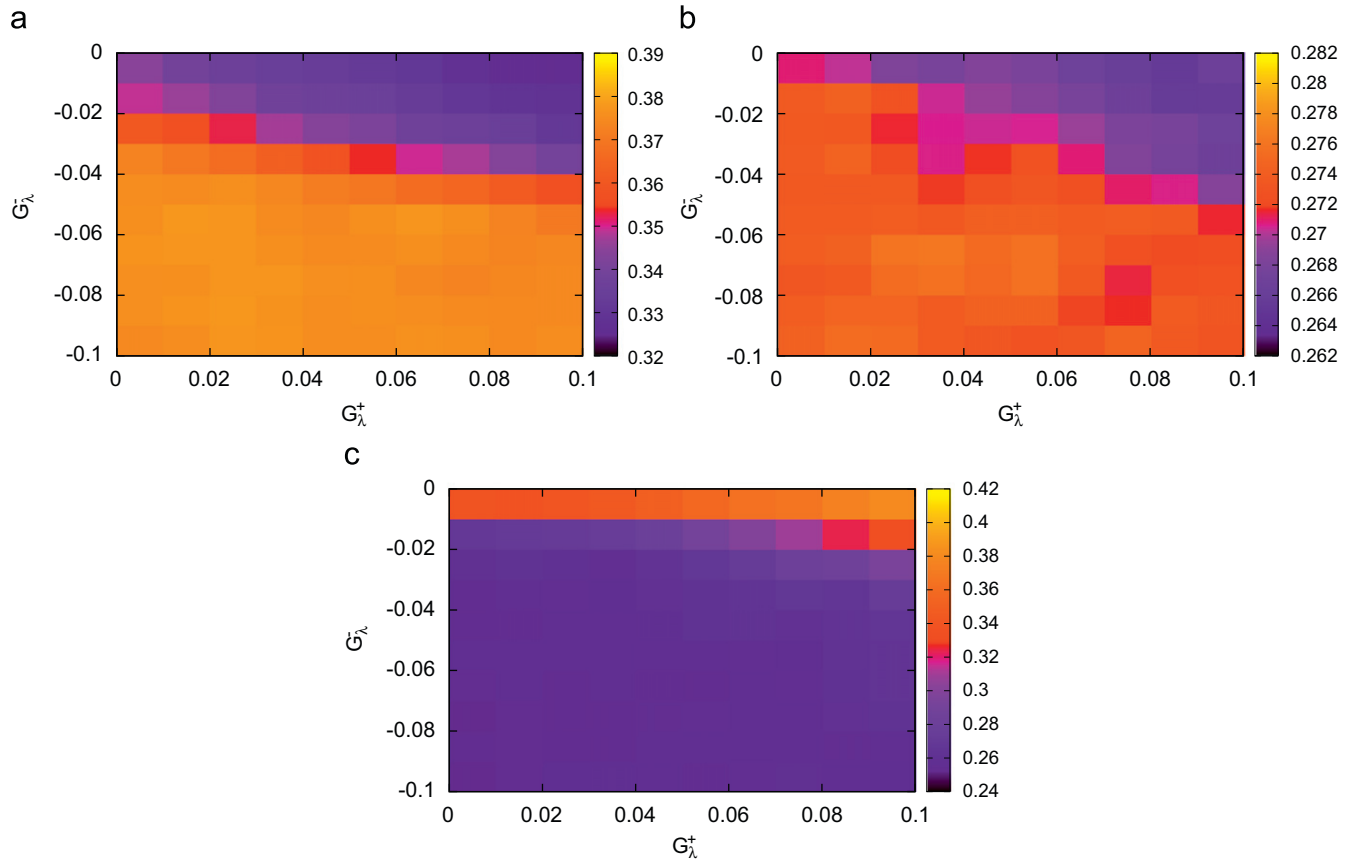

Fig. 6. Average error rate of the adaptive λ-perceptron algorithm for different values of the thresholds G_λ^+ and G_λ^-: (a) $\eta=0.01$; (b) $\eta=0.1$; and (c) $\eta=0.5$.

variances [46]. We ignore multiplicity issues arising due to the large number of tests performed. These tests are intended to give an idea of relative performance, and a statistically significant result increases our confidence that the conclusion reflects a genuine underlying reality. As a benchmark in the artificial datasets, the performance of the ideal classifier is reported. In this context the ideal classifier is Quadratic Discriminant Analysis (QDA) that employs the true $\mu_y(t)$ and $\Sigma_y(t)$, at each time-step to predict the labels of the feature vectors.

For the sliding window classifier, the examples in a window of predefined length, w, are used to update β. The parameter vector is updated every w time-steps by performing 10 iterations of the resilient propagation (RPROP) algorithm [43]. Each iteration of an offline gradient based optimisation algorithm, like RPROP, involves as many gradient computations as those required by the stochastic gradient descent with forgetting algorithms over w time-steps. The step-size for all the methods that employ such a parameter is set to 0.1. The adaptive perceptron, the adaptive Winnow and the OLDC classifiers estimate the classification error over a window to update their learning rate. Window lengths of 25, 50, 75 and 100 were considered for these methods. In all cases we report the results for the best choice of window length. For the PA I and PA II algorithms the aggressiveness parameter C is set to 10^{-3} [26]. We employ linear kernels for SPA, RBP, LBP, RBSOP and LBSOP. For SPA we consider all the choices for a constant scaling factor in [0.1,0.9] with a step-size of 0.1, and report the results for the best choice. The self-adaptive Forgetron, RBP, LBP, RBSOP, and LBSOP store a number of previous feature vectors which is

determined by the budget parameter, B. We consider the values $B=25, 100, 500, 1000$ and report the results for the best choice of this parameter in each case. The global meta-learning rate of the SMD is set to 0.1. For the adaptive λ-perceptron algorithm we set $\alpha=5\times10^{-3}$,$G_{\lambda}^{+}=0.035$, $G_{\lambda}^{-}=-0.015$, $\lambda(1)=\lambda^{+}=0.88$, and $\lambda^{-}=0.1$.

Table 1
Considered methods (with abbreviations).

1. Sliding window sigmoid perceptron (window)
2. Perceptron (Perc) [25]
3. Adaptive perceptron (ad Perc) [32]
4. Passive-aggressive (PA) [26]
5. Passive-aggressive I (PA I) [26]
6. Passive-aggressive II (PA II) [26]
7. Shifting perceptron (SPA) [30]
8. Randomised budget perceptron (RBP) [30]
9. Least recent budget perceptron (LBP) [30]
10. Randomised budget second-order perceptron (RBSOP) [30]
11. Least recent budget second-order perceptron (LBSOP) [30]
12. Self-adaptive Forgetron (Forgetron) [29]
13. Adaptive Winnow (ad Winnow) [32]
14. Online linear discriminant classifier (OLDC) [32]
15. Sigmoid perceptron using stochastic meta-descent (SMD) [34]
16. Constant forgetting λ-perceptron
17. Adaptive λ-perceptron (λ-Perc)

Table 2
Average error rate on static environments with standard deviation in parentheses.

	$d=2$	$d=20$
Ideal QDA	0.158 (0.125)	0.193 (0.050) +
Window 50	0.161 (0.119)	0.201 (0.044) +
Perc	0.197 (0.140) +	0.172 (0.047) +
Ad Perc	0.188 (0.136) +	0.164 (0.044) +
PA	0.200 (0.142) +	0.166 (0.046) +
PA I	0.150 (0.119)	0.120 (0.034) –
PA II	0.149 (0.117)	0.125 (0.035)
SPA	0.202 (0.141) +	0.174 (0.047) +
RBP	0.202 (0.141) +	0.176 (0.049) +
LBP	0.202 (0.141) +	0.176 (0.048) +
RBSOP	0.197 (0.140) +	0.178 (0.049) +
LBSOP	0.198 (0.140) +	0.178 (0.049) +
Forgetron	0.197 (0.140) +	0.247 (0.052) +
OLDC	0.147 (0.116)	0.129 (0.035)
Ad Winnow	0.206 (0.130) +	0.245 (0.031) +
SMD	0.148 (0.117)	0.122 (0.035) –
Const. λ perc	0.148 (0.118)	0.124 (0.035)
λ-Perc	0.154 (0.122)	0.132 (0.036)

4.2.1. Static environment

In the first set of experiments we investigate the performance of the proposed approach in static environments. In a static environment the population distributions are constant over time. In the context of the artificially generated datasets this is achieved by having $\mu_y(t)=\mu_y(1)$, and $\Sigma_y(t)=\Sigma_y(1)$ for $y\in\{0,1\}$. Table 2 presents the average error rate achieved on two and 20 dimensional artificial datasets by all methods. The value of λ that yields the lowest average error rate for a constant forgetting λ-perceptron is in both cases zero. This is due to the rapid convergence of the performance of the λ-perceptron algorithm relative to the length of the simulation. As Fig. 7 shows, the performance of the adaptive λ-perceptron algorithm converges in less than 200 time-steps. Once performance converges in a static environment and given that the step-size parameter η is constant, greater values of λ increase the variability of the estimated minimiser at each time-step and hence compromise performance. Fig. 8 illustrates the evolution of the forgetting factor, $\lambda(t)$, for the adaptive λ-perceptron algorithm, averaged over all the simulations. Note that for both values of dimensionality, $d=2$, 20, the adaptive forgetting scheme steadily reduces the value of λ throughout the length of the simulation.

The convergence behaviour of different constant forgetting λ-perceptrons is illustrated in Fig. 9 for two values of $\eta=10^{-1}$ and 10^{-2}. The figure shows that for both values of η a high value of λ increases the speed of convergence. As discussed in Section 4.1, for smaller values of η higher values of λ yield

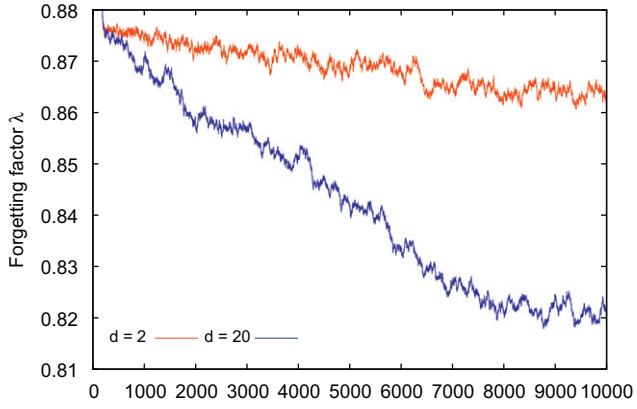

Fig. 8. Evolution of the forgetting factor in static environments.

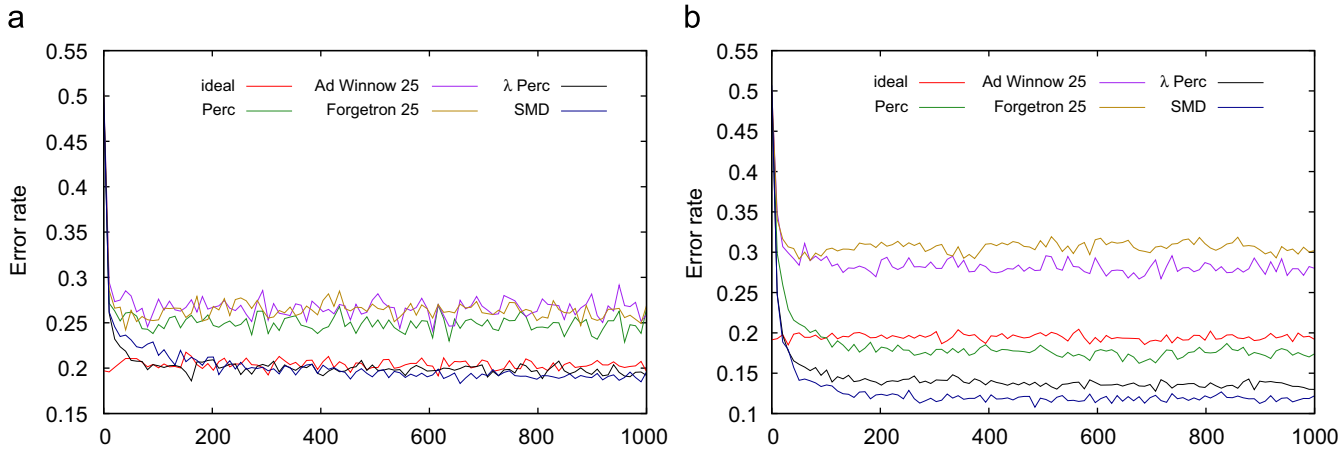

Fig. 7. Classification error rate on static environments: (a) $d=2$ and (b) $d=20$.

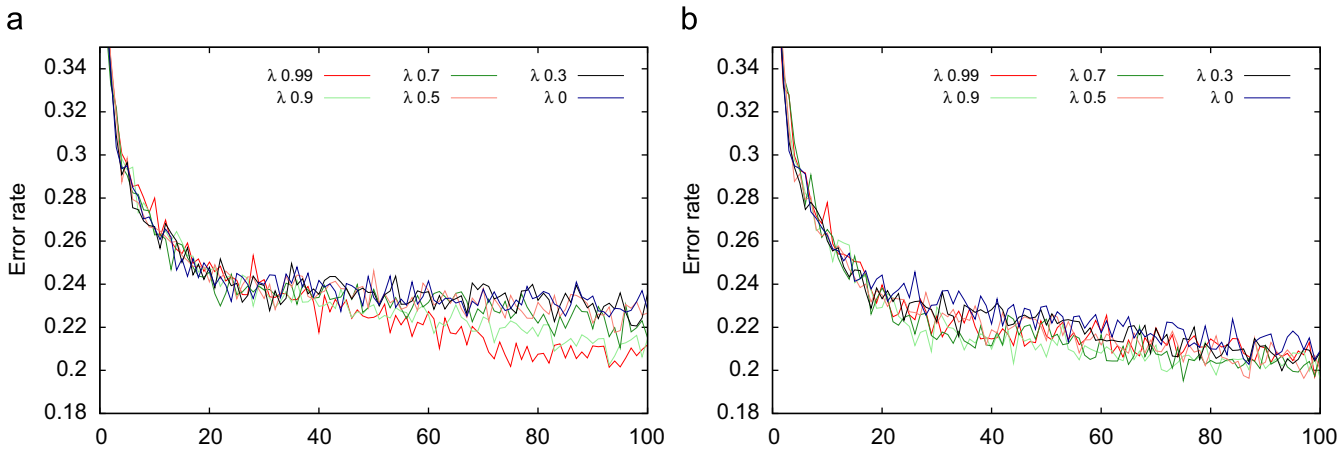

Fig. 9. Algorithm convergence on static problems for different constant forgetting λ-perceptrons: (a) $\eta=0.01$ and (b) $\eta=0.1$.

Table 3
Average error rate on gradually drifting environments.

	$d=2$		$d=20$	
	$\sigma=0.1$	$\sigma=0.2$	$\sigma=0.1$	$\sigma=0.2$
Ideal QDA	0.264 (0.015) –	0.134 (0.012) –	0.193 (0.005) –	0.093 (0.003) –
Window	0.356 (0.014) +	0.290 (0.020) +	0.447 (0.005) +	0.468 (0.005) +
Perc	0.340 (0.014) +	0.218 (0.016) –	0.355 (0.004) –	0.348 (0.004) –
Ad Perc	0.334 (0.013)	0.216 (0.015) –	0.358 (0.004) –	0.354 (0.004) –
PA	0.343 (0.015) +	0.214 (0.016) –	0.348 (0.004) –	0.340 (0.004) –
PA I	0.446 (0.017) +	0.386 (0.020) +	0.465 (0.005) +	0.466 (0.005) +
PA II	0.376 (0.014) +	0.293 (0.017) +	0.413 (0.004) –	0.414 (0.004) –
SPA	0.345 (0.014) +	0.220 (0.016) –	0.356 (0.004) –	0.349 (0.004) –
RBP	0.349 (0.014) +	0.226 (0.017) –	0.376 (0.004) –	0.343 (0.004) –
LBP	0.348 (0.014) +	0.224 (0.016) –	0.365 (0.003) –	0.326 (0.004) –
RBSOP	0.345 (0.014) +	0.225 (0.016) –	0.377 (0.004) –	0.374 (0.004) –
LBSOP	0.344 (0.014) +	0.223 (0.016) –	0.381 (0.004) –	0.380 (0.004) –
Forgetron	0.353 (0.013) +	0.229 (0.016) –	0.420 (0.003) +	0.411 (0.004) –
OLDC	0.384 (0.012) +	0.336 (0.017) +	0.423 (0.004) +	0.430 (0.004) +
Ad Winnow	0.344 (0.013) +	0.256 (0.014) –	0.414 (0.004) –	0.402 (0.003) –
SMD	0.348 (0.015) +	0.260 (0.016) –	0.425 (0.004) +	0.426 (0.004) +
	$\lambda=0.9$	$\lambda=0.8$	$\lambda=0.5$	$\lambda=0.3$
Const. λ perc	0.334 (0.014)	0.256 (0.015) –	0.400 (0.004) –	0.407 (0.004) –
λ-Perc	0.337 (0.015)	0.273 (0.016)	0.417 (0.004)	0.423 (0.004)

lower average error rate. Fig. 9 shows that for $\eta=10^{-2}$ the most rapid convergence is exhibited for $\lambda=0.99$. However, even for this step-size the value of λ that yields the lowest average error rate over the entire length of the simulation is substantially smaller, namely $\lambda=0.7$.

4.2.2. Gradual drift

The average error rate achieved by all the considered methods is reported in Table 3, for gradually drifting environments with dimensionality $d=2$, 20 and $\sigma=0.1,0.2$, corresponding to slow and rapid drift. Fig. 10 depicts the evolution of the classification error over the final 1000 time-steps. For clarity of presentation, the performance of six methods is shown in the figure. In the two dimensional case with slow drift, $\sigma=0.1$, the adaptive λ-perceptron is among the best performing methods. Increasing σ to 0.2 has the effect of increasing the distance between the mean vectors of the conditional probability density functions, $P(x(t)|y)$, $y=\{0,1\}$, and thereby reducing the extent of class overlap. This is reflected in the performance improvement achieved by all methods compared to the case with $\sigma=0.1$. Fig. 11 illustrates a trace plot of the parameters of an optimal sigmoid perceptron and the parameter estimates through the adaptive λ-perceptron algorithm in a single simulation. The optimal parameters of a sigmoid perceptron are estimated offline at each time-step using a dataset of 300 examples from the current populations. As the extent of class overlap decreases and the classes become more linearly separable, the perceptron, and its variants (adaptive perceptron, PA, SPA, RBP, LBP, RBSOP, LBSOP) compare more favourably to other methods. The extent of class overlap is much smaller in the higher dimensional datasets, $d=20$, with drift. For this reason, the best performance in these datasets is achieved by the perceptron and its aforementioned variants. In the higher dimensional datasets, the adaptive λ-perceptron algorithm achieves a statistically significant superior performance against SMD, OLDC, PA I and the window classifier.

The evolution of $\lambda(t)$ through the adaptive λ-perceptron algorithm, depicted in Fig. 12, indicates that after a transient period the adaptive scheme sets $\lambda(t)$ to a lower average value for a higher dimensionality of the feature vector, d, and a larger speed of drift, σ^2. The dimensionality appears to have a much larger

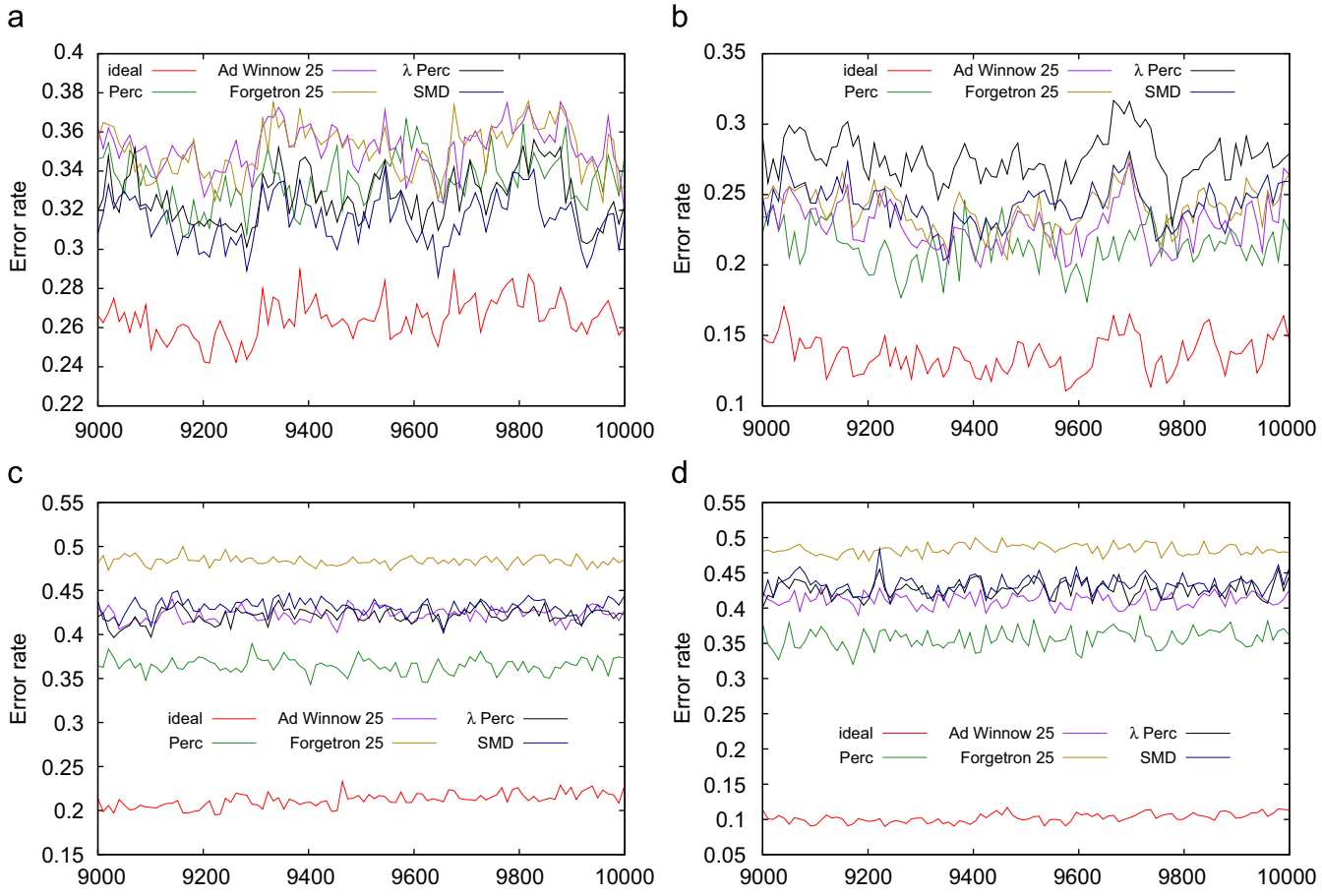

Fig. 10. Classification error at each time-step on gradually drifting environments with constant speed of drift: (a) $d=2$, $\sigma=0.1$; (b) $d=2$, $\sigma=0.2$; (c) $d=20$, $\sigma=0.1$; and (d) $d=20$, $\sigma=0.2$.

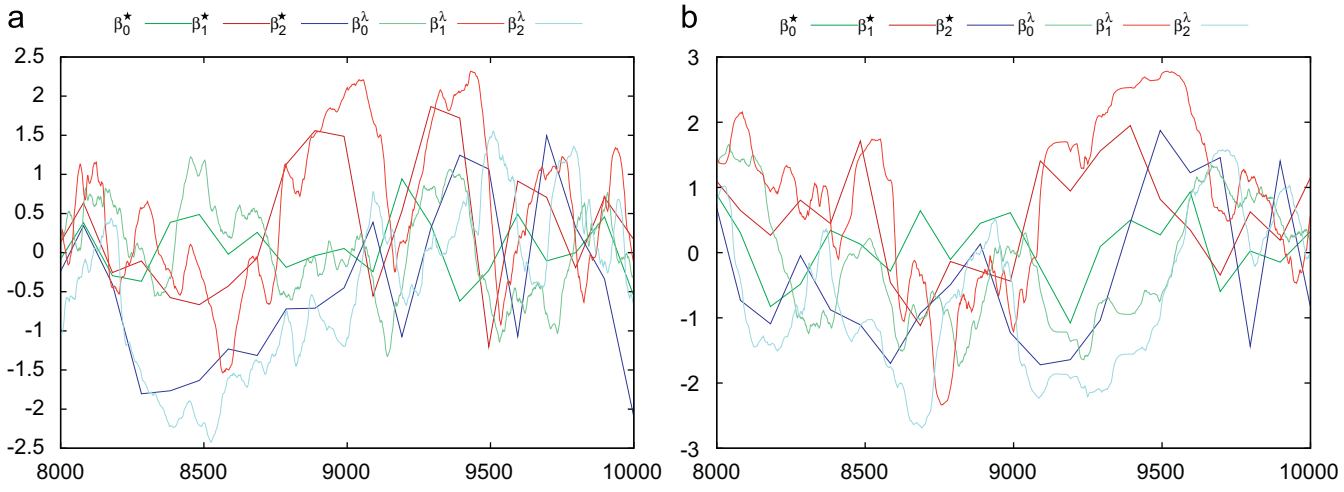

Fig. 11. Trace plots of the evolution of the parameters of an optimal sigmoid perceptron, $\beta^\star$, and the parameters estimated through the adaptive λ-perceptron algorithm, β^λ, under gradual drift: (a) $d=2$, $\sigma=0.1$ and (b) $d=2$, $\sigma=0.2$.

impact on the average value of $\lambda(t)$ than σ^2. Note that the same holds for the optimal constant value of λ reported in Table 3.

4.2.3. *Abrupt change*

As in the case of gradual drift, we consider two values of the dimensionality of the feature vector, $d=2,20$ and two values for the interval between two consecutive change points, $p=200,500$. The average error rate of all the considered methods on abruptly changing environments is reported in Table 4. The classification error during the final 1000 time-steps of the simulations is shown in Fig. 13.

The best performing methods in terms of average error rate are the perceptron, the adaptive perceptron and PA. Fig. 13 reveals

that these methods achieve a lower average error rate because they do not learn the classification problem between consecutive change points as well as methods like the adaptive λ-perceptron and SMD. As a consequence their performance is compromised less at each change point. The relative performance of the adaptive λ-perceptron depends on the extent of class overlap and on the frequency of change points. As the time interval between consecutive change points increases the benefits from learning each classification task effectively outweigh the performance deterioration at each change point, and vice versa. This finding also shows that in the case of abrupt changes there can be aspects of performance that are not captured by the average error rate criterion.

For the lower dimensional feature space, Fig. 14 provides trace plots of the parameters of an optimal sigmoid perceptron, and the parameter estimates through the adaptive λ-perceptron algorithm.

Fig. 15 depicts the evolution of $\lambda(t)$ through the adaptive λ-perceptron algorithm. After a transient period, the evolution of λ follows closely the pattern of abrupt changes in the data. Each abrupt change is accompanied by a decline of $\lambda(t)$, signifying that the importance of previous examples in the estimation of β is reduced. Following this decline and prior to the next change point, $\lambda(t)$ increases. Also note that the average values of $\lambda(t)$ induced by the adaptive forgetting scheme are close to the optimal constant values for λ reported in Table 4.

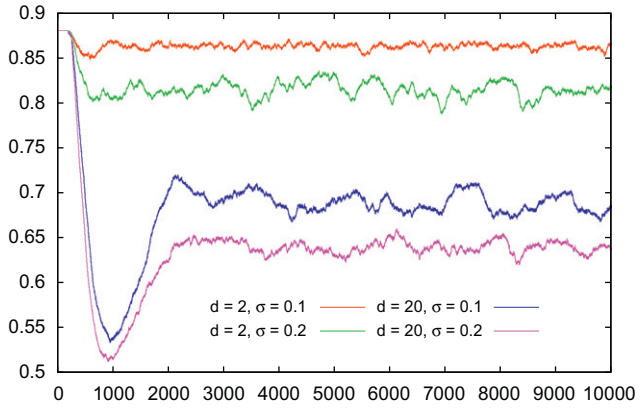

Fig. 12. Evolution of the forgetting factor in gradually drifting environments with constant speed of drift.

The evolution of $G_\lambda e(t)$ for $d=20$ is plotted in Fig. 16. The figure shows that at each abrupt change point $G_\lambda e(t)$ exhibits large positive spikes which cause the decrease in λ shown in Fig. 15. As the classifier learns how to distinguish between the two classes with the presentation of more examples from the current class definitions, $G_\lambda e(t)$ decreases. This pattern suggests that it might be possible to monitor $G_\lambda e(t)$ to detect abrupt population drift.

4.2.4. Gradual drift with abrupt changes

We finally consider the hardest case of population drift in which both types of population drift, abrupt changes and gradual drift, are present. We construct a dataset in which the class definitions are subject to gradual drift with $\sigma=0.1$ and abrupt changes every 500 time-steps. The average error rate for all the methods is reported in Table 5. For low dimensional examples the adaptive λ-perceptron algorithm outperforms most other methods. As in the previous cases, for $d=20$ the perceptron, and its variants perform substantially better than other algorithms because the extent of overlap between classes is smaller. Fig. 17 illustrates the performance of six methods during the last 1000 time-steps of the simulation. The figure shows that the impact that abrupt changes have on performance is more pronounced when $d=2$. This finding is reflected in the evolution of $G_\lambda e(t)$ depicted in Fig. 18. For $d=2$, the previously observed pattern of large positive values of $G_\lambda e(t)$ at each change point is retained despite the presence of gradual drift. On the contrary, for $d=20$, $G_\lambda e(t)$ appears to fluctuate randomly, as in the case of gradual drift. Trace plots for the lower dimensional feature space, of the parameters of an optimal sigmoid perceptron, $\beta^\star$, and the parameters estimated through the adaptive λ-perceptron algorithm, β^λ, are shown in Fig. 19.

4.3. Publicly available datasets

In this section we evaluate the performance of the proposed adaptive λ-perceptron algorithm on the following five publicly

Table 4
Average error rate on abruptly changing environments.

	$d=2$		$d=20$	
	$p=500$	$p=200$	$p=500$	$p=200$
Ideal QDA	0.173 (0.028) –	0.171 (0.017) –	0.192 (0.010) –	0.193 (0.006) –
Window	0.206 (0.024) –	0.250 (0.015) –	0.230 (0.008) –	0.272 (0.006) –
Perc	0.221 (0.030) –	0.225 (0.018) –	0.191 (0.009) –	0.215 (0.006) –
Ad Perc	0.214 (0.030) –	0.219 (0.018) –	0.187 (0.008) –	0.214 (0.006) –
PA	0.223 (0.031) –	0.226 (0.018) –	0.183 (0.009) –	0.207 (0.006) –
PA I	0.334 (0.028) +	0.395 (0.021) +	0.316 (0.012) +	0.388 (0.010) +
PA II	0.232 (0.025)	0.282 (0.017) +	0.211 (0.008) –	0.278 (0.007) –
SPA	0.226 (0.030) –	0.230 (0.018) –	0.192 (0.009) –	0.217 (0.006) –
RBP	0.227 (0.031)	0.233 (0.018) –	0.200 (0.009) –	0.230 (0.006) –
LBP	0.226 (0.031) –	0.232 (0.017) –	0.197 (0.009) –	0.229 (0.007) –
RBSOP	0.223 (0.031) –	0.231 (0.018) –	0.201 (0.009) –	0.234 (0.006) –
LBSOP	0.222 (0.030) –	0.229 (0.018) –	0.200 (0.009) –	0.235 (0.006) –
Forgetron	0.227 (0.030)	0.235 (0.017) –	0.300 (0.009) +	0.320 (0.006) +
OLDC	0.270 (0.024) +	0.323 (0.017) +	0.232 (0.011) –	0.298 (0.009) –
Ad Winnow	0.233 (0.028)	0.249 (0.017) –	0.311 (0.009) +	0.348 (0.006) +
SMD	0.227 (0.024)	0.254 (0.020) –	0.242 (0.008)	0.306 (0.007) +
	$\lambda=0.8$	$\lambda=0.8$	$\lambda=0.7$	$\lambda=0.5$
Const. λ perc	0.216 (0.023) –	0.249 (0.014) –	0.207 (0.008) –	0.266 (0.006) –
λ-Perc	0.234 (0.025)	0.268 (0.015)	0.242 (0.011)	0.304 (0.010)

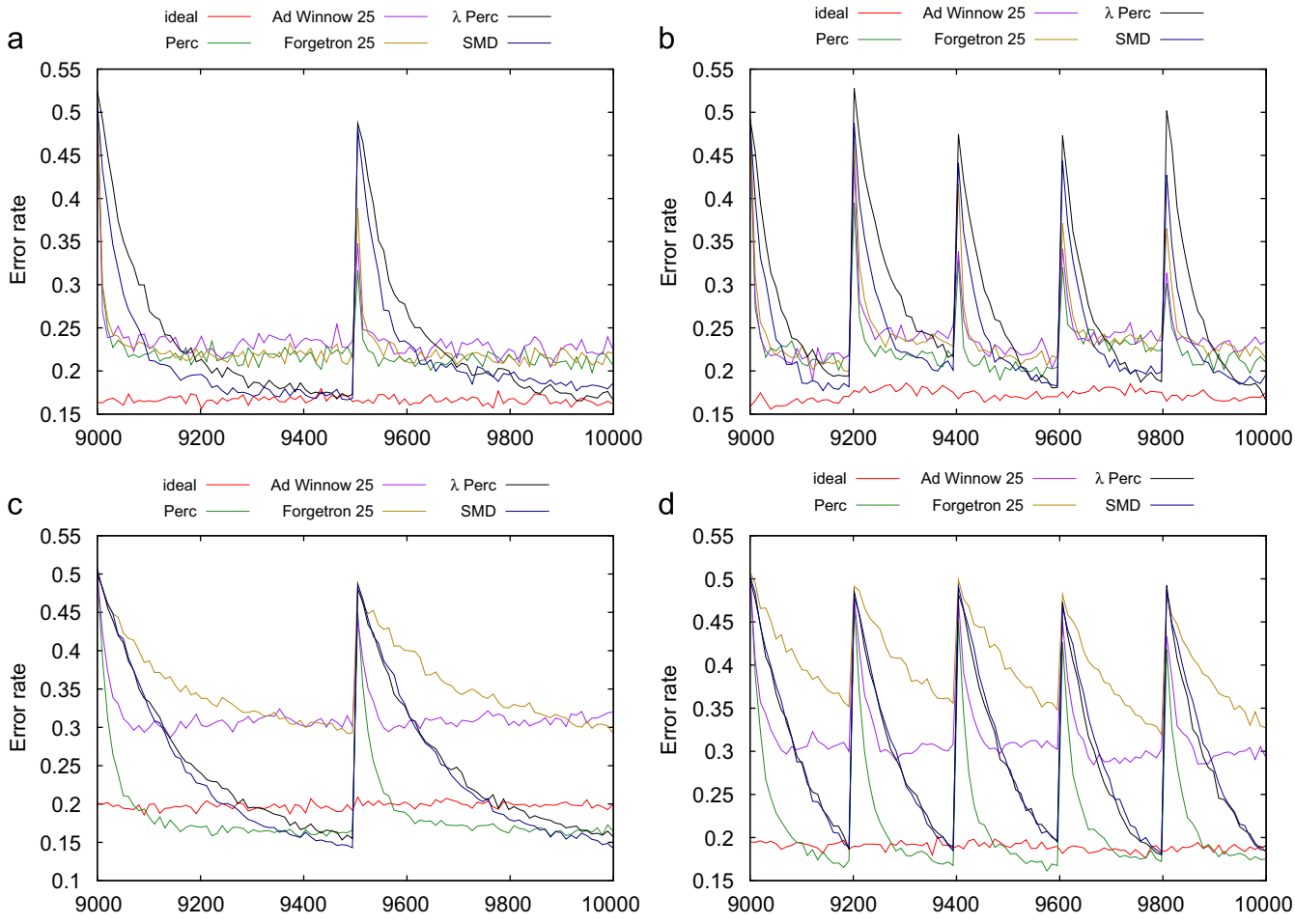

Fig. 13. Classification error rate on abruptly changing environment: (a) $d=2$, $p=500$; (b) $d=2$, $p=200$; (c) $d=20$, $p=500$; and (d) $d=20$, $p=200$.

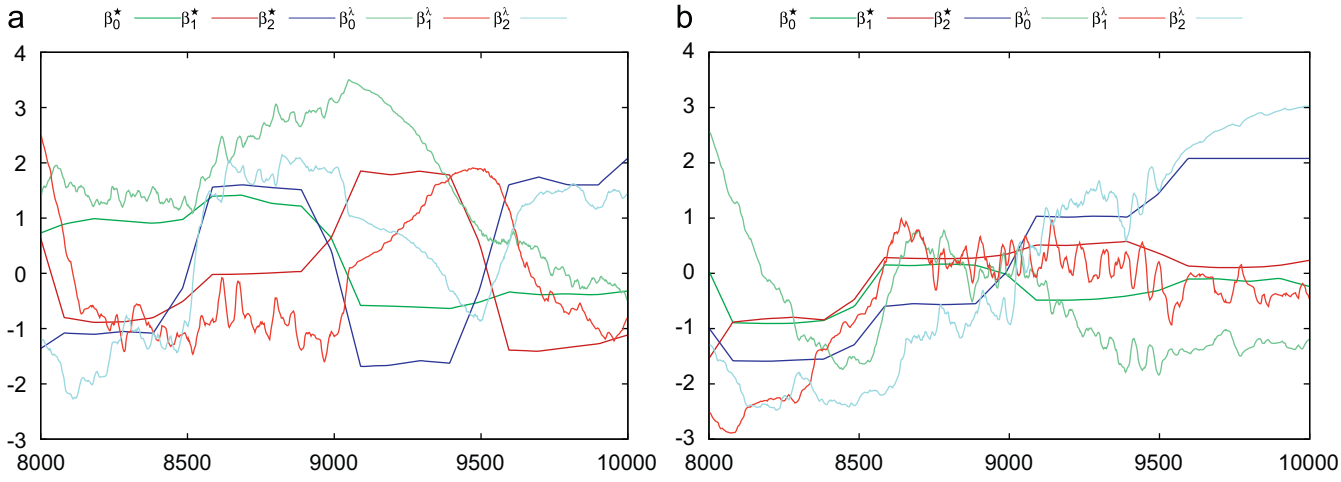

Fig. 14. Trace plots of the evolution of the parameters of an optimal sigmoid perceptron, $\beta^\star$, and the parameters estimated through the adaptive λ-perceptron algorithm, β^λ, in the case of abrupt changes: (a) $d=2$, $p=500$ and (b) $d=2$, $p=200$.

available datasets:

- *Stagger dataset* [47]: This is an artificial dataset in which classes are separable exhibiting abrupt population drift. Feature vectors contain three categorical variables each with three possible values. The dataset consists of 1200 examples and the class definitions change every 400 examples. The first and third class definitions yield linear classification boundaries, whereas the class boundary generated by the second class definition is non-linear. Classifier performance at each time-step is evaluated on a testing set of 200 examples generated using the current class definitions.
- *Moving hyperplane dataset* [48]: A gradually changing environment with two dimensional feature vectors and separable classes. The dataset is constructed by having a linear class

boundary that crosses the origin rotate one degree at each time-step. The total number of examples is 7200 corresponding to 20 full cycles. Feature vectors are sampled uniformly from the unit square. A testing dataset of 100 examples is used to evaluate classifier performance at each time-step.

- *Gauss dataset* [49]: A two-dimensional dataset that exhibits abrupt changes. Feature vectors are labelled according to two different but overlapping Gaussian density functions ($\mathcal{N}([0,0]^\top, I)$ and $\mathcal{N}([2,0]^\top, 4I)$). After each change point, the class labels are reversed.
- *SINE1 dataset* [49]: Two-dimensional dataset exhibiting abrupt changes, with separable classes. Each feature assumes values uniformly distributed in [0,1]. Initially, points that lie below the curve $y=\sin(x)$ are assigned to class 0, otherwise they are labelled as class 1. At each change point class labels are reversed.
- *Electricity market dataset* [50]: Dataset of prices in the Australian New South Wales (NSW) Electricity Market. The data consists of 45,312 successive measurements, taken every 30 min, spanning the period from May 1996 to December 1998. Each feature vector has five variables: day of week, time stamp, NSW electricity demand, an electricity demand measure, and scheduled electricity transfer between states. The class label is either up or down, referring to whether the current electricity price is higher or lower than the average price of the preceding 24 h. During the considered period the market was expanded by the inclusion of adjacent areas, which produced a more elaborate management of the supply.

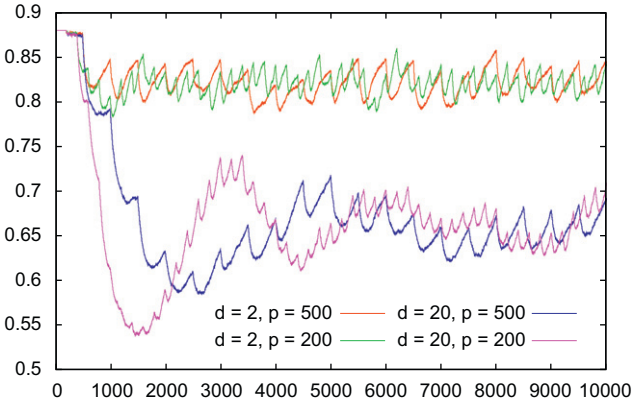

Fig. 15. Evolution of the forgetting factor λ in abruptly changing environments.

Table 6 reports the classification error for all the considered methods. For the STAGGER and Moving Hyperplane the reported results are averages over 100 simulations and the statistical significance of performance differences against the adaptive λ-perceptron algorithm is evaluated using paired t-tests. For specific datasets, such as GAUSS, SINE1, Electricity Market and the two real-world applications we consider in the following subsections, we evaluate statistical significance through McNemar's test [51]. The performance of the adaptive λ-perceptron is competitive with most other methods in the STAGGER and GAUSS datasets. In the Electricity Market dataset, most methods achieve statistically significant lower classification error than the adaptive λ-perceptron. In the Moving Hyperplane dataset the adaptive λ-perceptron, achieves a substantially lower average classification error than all other methods.

We employ the SINE1 dataset to evaluate the effect of noise on classification accuracy. In the original dataset classes are separable. To examine the effect of noise we add to each feature vector a random vector sampled from a zero-mean Gaussian with a covariance matrix, $\sigma^2 I$, where $\sigma^2 \in \{0,0.1,0.3,0.5\}$. The performance of the considered classifiers on these noisy datasets is reported in Table 7. In the original dataset, RBSOP and LBSOP

Table 5
Average error rate on artificial datasets exhibiting both gradual and abrupt drift.

	$d=2$, $p=500$, $\sigma=0.1$	$d=20$, $p=500$, $\sigma=0.1$
Ideal QDA	0.252 (0.013) –	0.192 (0.005) –
Window	0.362 (0.012) +	0.455 (0.005) +
Perc	0.331 (0.011)	0.359 (0.004) –
Ad Perc	0.324 (0.011) –	0.362 (0.004) –
PA	0.334 (0.011)	0.352 (0.004) –
PA I	0.445 (0.016) +	0.469 (0.005) +
PA II	0.371 (0.013) +	0.417 (0.005) –
SPA	0.336 (0.011)	0.360 (0.004) –
RBP	0.340 (0.010) +	0.378 (0.003) –
LBP	0.338 (0.011) +	0.366 (0.003) –
RBSOP	0.336 (0.011) +	0.384 (0.004) –
LBSOP	0.335 (0.011)	0.387 (0.004) –
Forgetron	0.345 (0.011) +	0.422 (0.003)
OLDC	0.382 (0.013) +	0.428 (0.005) +
Ad Winnow	0.335 (0.010)	0.416 (0.004) –
SMD	0.342 (0.013) +	0.429 (0.005) +
	$\lambda=0.8$	$\lambda=0.5$
Const. λ perc	0.329 (0.011) –	0.405 (0.005) –
λ-Perc	0.333 (0.011)	0.422 (0.005)

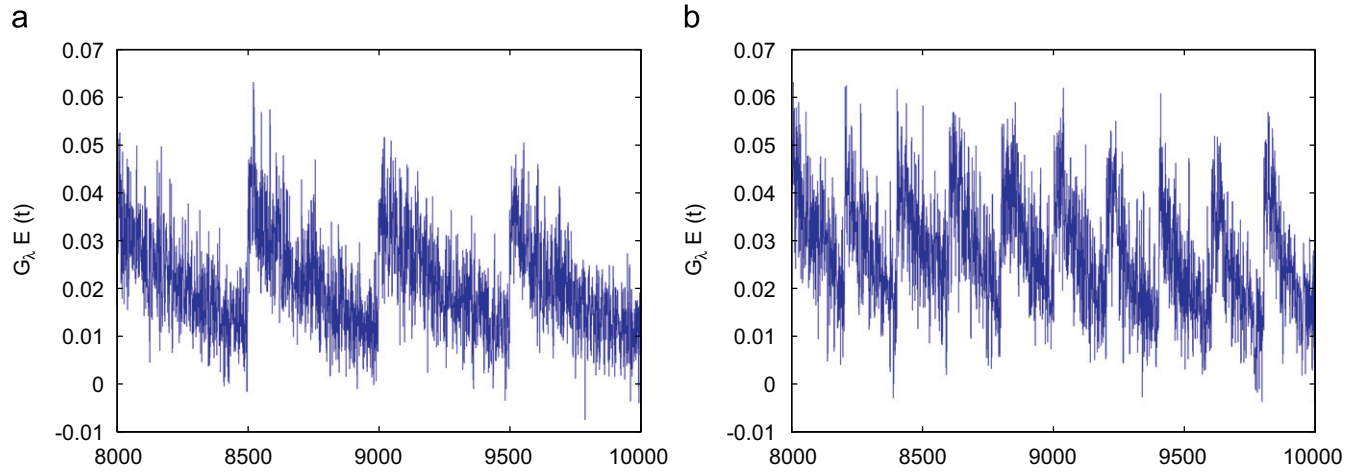

Fig. 16. Evolution of $G_\lambda e(t)$ in abruptly changing environments: (a) $d=20$, $p=500$ and (b) $d=20$, $p=200$.

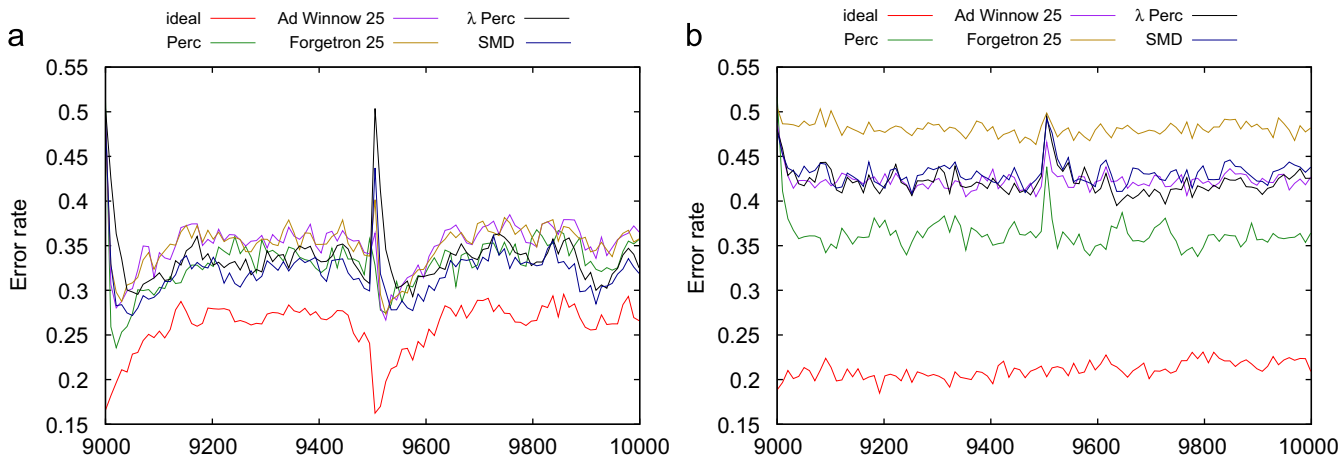

Fig. 17. Classification error on datasets that exhibit abrupt changes and gradual drift: (a) d=2 and (b) $d = 20$.

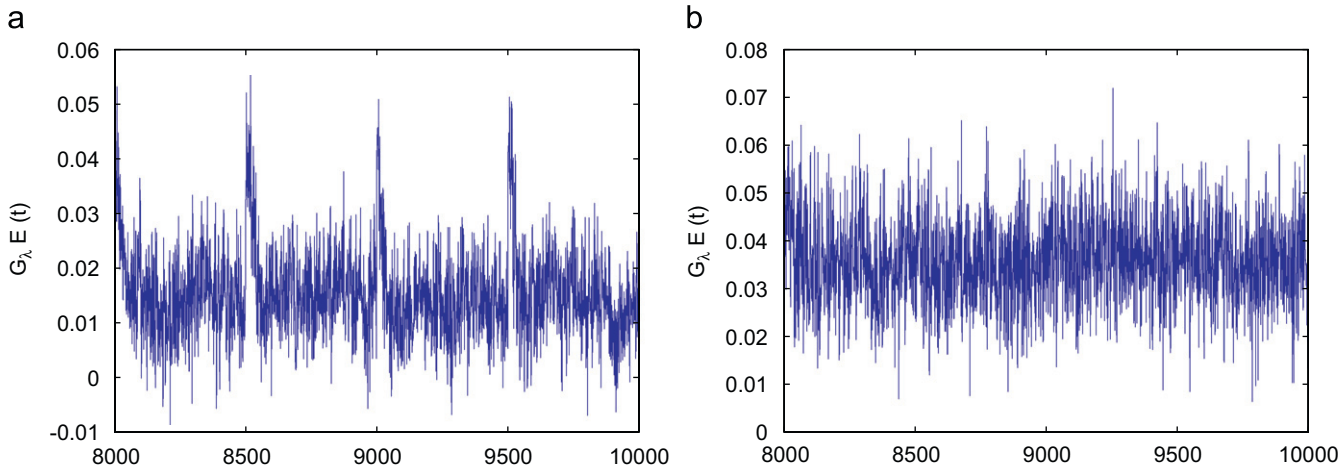

Fig. 18. Evolution of $G_\lambda e(t)$ in environments with both gradual and abrupt drift: (a) $d = 2$ and (b) $d = 20$.

exhibit the lowest classification error. The performance of the perceptron and its variants deteriorates substantially with the introduction of noise. For $\sigma \geq 0.3$ the performance of these methods is not better than assigning labels randomly. Note that as σ^2 increases the requirement of the PA algorithm to make predictions with high confidence impairs its performance relative to the other perceptron variants. For all levels of noise the sliding window classifier, and the adaptive λ-perceptron perform very well.

Overall, the experimental results suggest that the adaptive λ-perceptron algorithm is most suitable to applications that exhibit gradual drift with non-linearly separable classes. An important class of problems in which classes are not separable is when either the feature vectors, or the labels are corrupted by noise. The adaptive λ-perceptron can also handle abrupt changes when classes are separable but the perceptron and the PA algorithm achieve a higher classification accuracy under these conditions.

4.4. Automated tumour detection

We further investigate the performance of the proposed method on a dataset concerned with the automated detection of tumours in colonoscopic video sequences [52]. The accurate online classification of imaging data can contribute to the early detection of colorectal cancer precursors, and assist in the early diagnosis of colorectal cancer. In the dataset textures from normal and abnormal tissue samples were randomly chosen from four frames of the same video sequence without applying any preprocessing to the data [52]. Feature extraction was performed using the method of co-occurrence matrices [53]. This method represents the spatial distribution and the dependence of the grey levels within a local area using an image window of size 16 by 16 pixels. Each feature vector thus constructed, contains 16 elements. The class label designates whether a window contains tumour pixels (class 1), or not (class 0). The dataset consists of 17,076 feature vectors. The four frame sequence is exhibited in Fig. 20.

Table 8 reports the average classification error attained by the considered methods, while Fig. 21(a) illustrates the average classification error over a window of length 100 for six methods. The best performing methods on this dataset are the Forgetron, PA, the perceptron and the adaptive Winnow. Fig. 21(b) depicts the evolution of λ through the adaptive forgetting scheme (top) and the proportion of patterns belonging to class 1 in a data window of length 100 (bottom). The bottom part of Fig. 21(b) indicates that there are periods during which no tumour pixels are encountered. The methods that always update their parameters, i.e. the adaptive λ-perceptron, SMD and OLDC, appear to overfit the data during these periods and suffer a large classification error when tumour pixels appear.

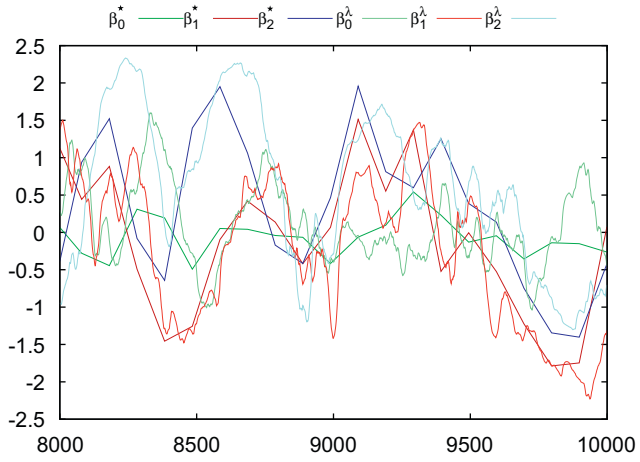

Fig. 19. Trace plots of the evolution of the parameters of an optimal sigmoid perceptron and the parameters estimated through the adaptive λ-perceptron algorithm in the case of gradual drift and abrupt changes: $d=2$, $p=500$, $\sigma=0.1$.

Table 6
Classification error on publicly available datasets.

	STAGGER	Mov. hyperplane	Gauss	Elec. market
Window	0.185 (0.012) –	0.322 (0.009) +	0.309 –	0.342 +
Perc	0.254 (0.013) +	0.333 (0.005) +	0.455 +	0.166 –
Ad Perc	0.238 (0.021) –	0.332 (0.006) +	0.435 +	0.166 –
PA	0.238 (0.009) –	0.376 (0.004) +	0.400 +	0.152 –
PA I	0.351 (0.018) +	0.498 (0.004) +	0.466 +	0.380 +
PA II	0.303 (0.015) +	0.467 (0.009) +	0.406 +	0.318 +
SPA	0.248 (0.016)	0.353 (0.005) +	0.413 +	0.166 –
RBP	0.248 (0.016)	0.358 (0.007) +	0.401 +	0.196 –
LBP	0.248 (0.016)	0.343 (0.007) +	0.405 +	0.176 –
RBSOP	0.167 (0.007) –	0.342 (0.007) +	0.461 +	0.186 –
LBSOP	0.163 (0.006) –	0.322 (0.008) +	0.467 +	0.197 –
Forgetron	0.041 (0.006) –	0.378 (0.009) +	0.201 –	0.159 –
OLDC	0.270 (0.021) +	0.419 (0.011) +	0.353 –	0.325 +
Ad Winnow	0.330 (0.033) +	0.302 (0.007) +	0.427 +	0.166 –
SMD	0.286 (0.016) +	0.403 (0.013) +	0.367	0.250 –
	$\lambda=0.9$	$\lambda=0.8$	$\lambda=0.8$	$\lambda=0.3$
Const. λ perc	0.246 (0.014)	0.220 (0.007) –	0.326 –	0.259 –
λ-Perc	0.246	0.223	0.364	0.296

Table 7
Classification error on SINE1 dataset for different levels of noise.

	$\sigma^2=0$	$\sigma^2=0.1$	$\sigma^2=0.3$	$\sigma^2=0.5$
Window	0.072 –	0.280 (0.004) –	0.360 (0.005) –	0.394 (0.006) –
Perc	0.153 –	0.415 (0.006) +	0.530 (0.006) +	0.557 (0.007) +
Ad Perc	0.128 –	0.368 (0.007) +	0.494 (0.007) +	0.538 (0.007) +
PA	0.113 –	0.459 (0.008) +	0.597 (0.007) +	0.619 (0.007) +
PA I	0.499 +	0.503 (0.004) +	0.501 (0.004) +	0.501 (0.004) +
PA II	0.508 +	0.497 (0.004) +	0.479 (0.003) +	0.470 (0.004) +
SPA	0.132 –	0.422 (0.007) +	0.545 (0.006) +	0.575 (0.007) +
RBP	0.128 –	0.396 (0.007) +	0.489 (0.007) +	0.514 (0.006) +
LBP	0.102 –	0.398 (0.006) +	0.506 (0.006) +	0.530 (0.006) +
RBSOP	0.035 –	0.350 (0.006) +	0.450 (0.005) +	0.481 (0.005) +
LBSOP	0.029 –	0.352 (0.005) +	0.453 (0.005) +	0.485 (0.006) +
Forgetron	0.113 –	0.407 (0.006) +	0.516 (0.006) +	0.536 (0.006) +
OLDC	0.160 –	0.310 (0.007) –	0.385 (0.007) +	0.415 (0.005) +
Ad Winnow	0.356 +	0.419 (0.010) +	0.471 (0.007) +	0.510 (0.009) +
SMD	0.408 +	0.398 (0.005) +	0.413 (0.005) +	0.424 (0.005) +
	$\lambda=0.6$	$\lambda=0.9$	$\lambda=0.8$	$\lambda=0.6$
Const. λ perc	0.288 –	0.294 (0.004) –	0.360 (0.004) –	0.394 (0.006) –
λ-Perc	0.342	0.317 (0.004)	0.362 (0.004)	0.400 (0.005)

The evolution of the forgetting factor by the adaptive λ-perceptron, illustrated in Fig. 21(b), indicates that the algorithm attempts to accommodate these abrupt changes in the prior probability of each class by sharply reducing the forgetting factor.

4.5. High-frequency foreign exchange

In the past the cost of gathering high frequency foreign exchange data was prohibitive and datasets were constructed by discrete sampling at much lower frequencies, e.g. daily data. The analysis of high frequency data has the potential to reveal important aspects of the operation of the foreign exchange market, as participants in these markets form decisions by observing high-frequency data. The advent of electronic technology has enabled the collection of high-frequency foreign exchange rate data. However, the analysis of this data presents the problem of dealing with a process that is time-varying [7]. Forecasting the level and the direction of change of foreign exchange rates has proven a challenging task [54].

We consider time-series of 10 min closing prices for three exchange rates against the Euro, namely the exchange rate of the Euro against the US Dollar (EUR/USD), the exchange rate of the British pound against the Euro (GBP/EUR), and the exchange rate of the Euro against the Japanese Yen (EUR/JPY). The data cover the period from 21/10/2002 until 15/5/2007 and each time-series consists of more than 165,000 observations. We use the previous ten values of the exchange rate as features to predict the direction of the exchange rate movement at the current time-step. In particular, the feature vector at each time-step t, $t>10$, is $x(t)=(p(t),\ldots,p(t-9))$, where $p(t)$ is the value of the exchange rate at time t, and the class label $y(t)$ is defined as:

$$y(t)=\begin{cases}1 & \text{if } p(t+1)\geq p(t),\\ 0 & \text{otherwise.}\end{cases}$$

The average classification error achieved by each method is reported in Table 9. The best performing methods in these datasets are PA I and PA II, the modifications of the PA algorithm that account for noise, OLDC, SMD and the adaptive λ-perceptron. In particular, for the EUR/GBP exchange rate these methods achieve an average accuracy in predicting the direction of change in excess of 60%. The performance of all the methods that do not accommodate for class overlap, like the perceptron, PA, and Forgetron, is not better than deciding randomly the direction of the exchange rate movement.

Table 8
Average classification error on colonoscopic images dataset.

Window	0.064 –
Perc	0.056 –
Ad Perc	0.059 –
PA	0.054 –
PA I	0.281 +
PA II	0.206 +
SPA	0.062 –
RBP	0.063 –
LBP	0.062 –
RBSOP	0.097 –
LBSOP	0.088 –
Forgetron	0.045 –
OLDC	0.162
Ad Winnow	0.056 –
SMD	0.192 +
Const. λ perc 0.3	0.141 –
λ-Perc	0.163

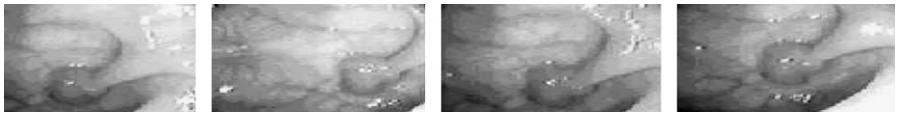

Fig. 20. The four frame sequence of the colonoscopic images.

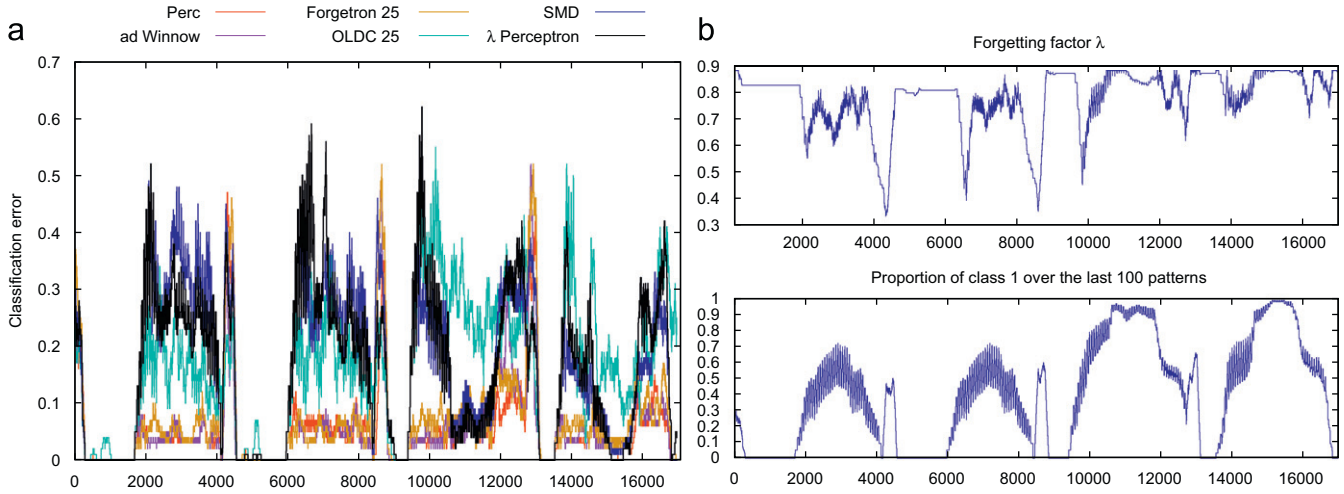

Fig. 21. Classification error on the colonoscopic images dataset (left) and evolution of λ and proportion of patterns belonging to class 1 in a data window of length 100 (right).

Table 9
Average classification error on 10 min closing prices of high-frequency foreign exchange data.

	EUR/USD	EUR/JPY	EUR/GBP
Window	0.433 −	0.448 +	0.375 +
Perc	0.515 +	0.517 +	0.497 +
Ad Perc	0.515 +	0.516 +	0.493 +
PA	0.516 +	0.518 +	0.500 +
PA I	0.430 −	0.440 −	0.370 −
PA II	0.434 −	0.442 −	0.371
SPA	0.515 +	0.516 +	0.497 +
RBP	0.504 +	0.506 +	0.484 +
LBP	0.509 +	0.509 +	0.483 +
RBSOP	0.491 +	0.496 +	0.457 +
LBSOP	0.489 +	0.494 +	0.459 +
Forgetron	0.515 +	0.517 +	0.499 +
OLDC	0.430 −	0.442	0.375 +
Ad Winnow	0.515 +	0.516 +	0.498 +
SMD	0.433 −	0.441 −	0.371 +
	$\lambda=0.3$	$\lambda=0.3$	$\lambda=0$
Const. λ perc	0.430 −	0.440 −	0.370 −
λ-Perc	0.435	0.442	0.371

5. Conclusions

The scope of this work has been to develop an adaptive online approach for perceptrons with sigmoidal activation functions suitable to streaming data applications. We develop a formulation in which adaptation is achieved by exponentially weighting, and thereby forgetting, past contributions to the current descent direction in parameter space. In this framework, a gradient descent scheme can be employed to adjust the forgetting factor in an online and data-driven manner. The storage requirements of this scheme would render it impractical for streaming data applications. Motivated by this formulation, we propose an algorithm that only involves quantities that can be updated online. The computational complexity of this algorithm scales linearly with the dimensionality of the feature vector.

The proposed approach can be viewed as a compromise between offline and online learning in the following sense. For small values of the step-size of the stochastic gradient descent scheme, previous estimates of the gradient provide accurate information concerning the gradient of the cumulative error function at the current point in parameter space. Under these conditions, introducing forgetting accelerates adaptation and moreover, the information in the evolution of the forgetting factor can be used to obtain insight about the type and speed of the underlying population drift process. As the step-size of the stochastic gradient descent scheme increases, previous estimates of the gradient become less informative about the direction of descent from the current point in the parameter space. In these cases the optimal forgetting factor is zero, and the proposed method becomes equivalent to stochastic gradient descent.

Experimental results show that the proposed approach exhibits robust behaviour under various types of population drift, and that the adaptive scheme adjusts the forgetting factor towards values that yield higher classification performance. Extensive comparison with numerous online linear classifiers on artificially constructed and real-world datasets shows that the adaptive λ-perceptron is more suitable for applications that exhibit gradual drift and when the classes are not separable. In future work we intend to extend this work to generalised linear models and multilayer neural networks for classification and regression tasks.

Acknowledgements

We wish to thank the reviewers for their valuable comments and suggestions which greatly improved this paper. This research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Networks) project and is jointly funded by a BAE Systems and EPSRC (Engineering and Physical Research Council) strategic partnership, under EPSRC

Grant EP/C548051/1. David Hand was partially supported by a Royal Society Wolfson Research Merit Award.

References

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, Models and issues in data streams, in: PODS '02: Proceedings of ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, New York, NY, USA2002, pp. 1–16.
[2] J. Gama, P. Rodrigues, Data stream processing, in: J. Gama, M. Gaber (Eds.), Learning from Data Streams: Processing Techniques in Sensor Networks, Springer2007, pp. 25–39.
[3] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 97–106.
[4] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 226–235.
[5] D.J. Hand, Classifier technology and the illusion of progress, Statistical Science 21 (1) (2006) 1–34.
[6] M.G. Kelly, D.J. Hand, N.M. Adams, The impact of changing populations on classifier performance, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 367–371.
[7] C.A.E. Goodhart, M. O'Hara, High frequency data in financial markets: issues and applications, Journal of Empirical Finance 4 (1997) 73–114.
[8] F. Fdez-Riverola, E.L. Iglesias, F. Díaz, J.R. Méndez, J.M. Corchado, Spamhunting: an instance-based reasoning system for spam labelling and filtering, Decision Support Systems 43 (3) (2007) 722–726.
[9] B. Crabtree, S.J. Soltysiak, Identifying and tracking changing interests, International Journal on Digital Libraries 2 (1) (1998) 38–53.
[10] M. Black, R. Hickey, Learning classification rules for telecom customer call data under concept drift, Soft Computing 8 (2) (2004) 102–108.
[11] L. Cohen, G. Avrahami-Bakish, M. Last, A. Kandel, O. Kipersztok, Real-time data mining of non-stationary data streams from sensor networks, Information Fusion 9 (3) (2008) 344–353.
[12] J. Whittaker, C. Whitehead, M. Somers, A dynamic scorecard for monitoring baseline performance with application to tracking a mortgage portfolio, Journal of the Operational Research Society 58 (7) (2007) 911–921.
[13] M. Basseville, I. Nikiforov, Detection of Abrupt Changes: Theory and Application, Prentice-Hall, 1993.
[14] N. Cesa-Bianchi, G. Lugosi, Prediction, Learning and Games, Cambridge University Press, 2006.
[15] M. Hebster, M.K. Warmuth, Tracking the best expert, Machine Learning 32 (2) (1998) 151–178.
[16] Y. Freund, R.E. Schapire, Y. Singer, M.K. Warmuth, Using and combining predictors that specialize, in: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1997, pp. 334–343.
[17] E. Lehrer, A wide range no-regret theorem, Games and Economic Behavior 42 (1) (2003) 101–115.
[18] A. Blum, Y. Mansour, From external to internal regret, Journal of Machine Learning Research 8 (2007) 1307–1324.
[19] E. Hazan, C. Seshadhri, Efficient learning algorithms for changing environments, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 393–400.
[20] C.C. Aggarwal, C. Charu, J. Han, J. Wang, P.S. Yu, On demand classification of data streams, in: KDD '04: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA2004, pp. 503–508.
[21] M. Black, R.J. Hickey, Maintaining the performance of a learned classifier under concept drift, Intelligent Data Analysis 3 (6) (1999) 453–474.
[22] A. Benveniste, M. Métivier, P. Priouret, Adaptive Algorithms and Stochastic Approximation, Springer-Verlag, New York, 1990.
[23] S. Haykin, Adaptive Filter Theory, third ed., Prentice-Hall International, 1996.
[24] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations of backpropagation errors, Nature 323 (1986) 533–536.
[25] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, Psychological Review 65 (6) (1958) 386–407.
[26] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, Y. Singer, Online passive–aggressive algorithms, Journal of Machine Learning Research 7 (2006) 551–585.
[27] K. Crammer, J. Kandola, Y. Singer, Online classification on a budget, Advances in Neural Information Processing Systems 16 (2004) 225–232.
[28] J. Weston, A. Bordes, L. Bottou, Online (and offline) on an even tighter budget, in: Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics, 2005, pp. 413–420.
[29] O. Dekel, S. Shalev-Shwartz, Y. Singer, The forgetron: a kernel-based perceptron on a budget, SIAM Journal on Computing 37 (5) (2008) 1342–1372.
[30] G. Cavallanti, N. Cesa-Bianchi, C. Gentile, Tracking the best hyperplane with a simple budget perceptron, Machine Learning 69 (2/3) (2007) 143–167.
[31] N. Cesa-Bianchi, A. Conconi, C. Gentile, A second-order perceptron algorithm, SIAM Journal on Computing 34 (3) (2005) 640–668.
[32] L.I. Kuncheva, C.O. Plumpton, Adaptive learning rate for online linear discriminant classifiers, in: N. da Vitoria Lobo, T. Kasparis, M. Georgiopoulos, F. Roli, J. Kwok, G.C. Anagnostopoulos, M. Loog (Eds.), Lecture Notes in Computer Science, vol. 53, 2008, pp. 510–519.
[33] N. Littlestone, Learning quickly when irrelevant attributes abound: a new linearthreshold algorithm, Machine Learning 2 (1988) 285–318.
[34] N.N. Schraudolph, Local gain adaptation in stochastic gradient descent, in: Proceedings of the International Conference on Artificial Neural Networks, 1999, pp. 569–574.
[35] L.B. Almeida, T. Langlois, J.D. Amaral, A. Plakhov, Parameter adaptation in stochastic optimization, in: D. Saad (Ed.), On-Line Learning in Neural Networks, Cambridge University Press1999, pp. 111–134.
[36] M. Niedźwiecki, Identification of Time-Varying Processes, Wiley, 2000.
[37] C. Bishop, Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 1995.
[38] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Networks 1 (4) (1988) 295–307.
[39] S. Haykin, Neural Networks: A Comprehensive Foundation, second ed., Prentice-Hall International, 1999.
[40] O. Mangasarian, M. Solodov, Serial and parallel backpropagation convergence via nonmonotone perturbed minimization, Optimization Methods and Software 4 (1994) 103–116.
[41] B.A. Pearlmutter, Fast exact multiplication by the Hessian, Neural Computation 6 (1) (1994) 147–160.
[42] C. Igel, M. Husken, Empirical evaluation of the improved Rprop learning algorithms, Neurocomputing 50 (2003) 105–123.
[43] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the RPROP algorithm, in: IEEE International Conference on Neural Networks, San Francisco, CA, 1993, pp. 586–591.
[44] F.M. Silva, dL.B. Almeida, Speeding-up backpropagation, in: R. Eckmiller (Ed.), Advanced Neural Computers, Elsevier1990, pp. 151–160.
[45] T. Tollenaere, Supersab: fast adaptive back propagation with good scaling properties, Neural Networks 3 (5) (1990) 561–573.
[46] B.L. Welch, The generalization of student's problem when several different population variances are involved, Biometrika 34 (1–2) (1947) 28–35.
[47] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine Learning 23 (1996) 60–101.
[48] A. Narasimhamurthy, L.I. Kuncheva, A framework for generating data to simulate changing environments, in: Proceedings of IASTED International Conference on Artificial Intelligence and Applications, 2007, pp. 384–389.
[49] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, R. Morales-Bueno, Early drift detection method, in: ECML PKDD 2006 Workshop on Knowledge Discovery from Data Streams, 2006.
[50] M. Harries, Splice-2 comparative evaluation: electricity pricing, Technical Report, University of South Wales, 1999.
[51] Q. McNemar, Note on the sampling error of the difference between correlated proportions or percentages, Psychometrika 12 (2) (1947) 153–157.
[52] S.A. Karkanis, G.D. Magoulas, N. Theofanous, Image recognition and neuronal networks: intelligent systems for the improvement of imaging information, Minimal Invasive Therapy & Allied Technologies 9 (3–4) (2000) 225–230.
[53] R.M. Haralick, Statistical and structural approaches to texture, Proceedings of the IEEE 67 (5) (1979) 786–804.
[54] C. Brooks, Linear and non-linear (non)-forecastability of high-frequency exchange rates, Journal of Forecasting 16 (2) (1998) 125–145.

Nicos G. Pavlidis is a Research Associate at the Institute for Mathematical Sciences, at Imperial College London. He has studied Economics at the University of Cambridge and holds a Ph.D. in Mathematics and Computer Science from the University of Patras, Greece.

Dimitris K. Tasoulis holds a Ph.D. and an M.Sc. in Mathematics and Computer Science. He is currently a Lecturer in the Mathematics Department, at Imperial College London.

Niall Adams is currently a Reader in the Department of Mathematics, Imperial College London. His research interests include pattern recognition and data mining. His application interests include consumer finance and cell biology.

David Hand is Professor of Statistics at Imperial College, London. He studied Mathematics at the University of Oxford and Statistics and Pattern Recognition at the University of Southampton. He is currently President of the Royal Statistical Society.