Adv Data Anal Classif (2009) 3:243–261
DOI 10.1007/s11634-009-0051-x

REGULAR ARTICLE

Temporally adaptive estimation of logistic classifiers on data streams

Christoforos Anagnostopoulos · Dimitris K. Tasoulis · Niall M. Adams · David J. Hand

Received: 12 May 2009 / Revised: 21 October 2009 / Accepted: 26 October 2009 /
Published online: 14 November 2009
© Springer-Verlag 2009

Abstract Modern technology has allowed real-time data collection in a variety of domains, ranging from environmental monitoring to healthcare. Consequently, there is a growing need for algorithms capable of performing inferential tasks in an online manner, continuously revising their estimates to reflect the current status of the underlying process. In particular, we are interested in constructing online and temporally adaptive classifiers capable of handling the possibly drifting decision boundaries arising in streaming environments. We first make a quadratic approximation to the log-likelihood that yields a recursive algorithm for fitting logistic regression online. We then suggest a novel way of equipping this framework with self-tuning forgetting factors. The resulting scheme is capable of tracking changes in the underlying probability distribution, adapting the decision boundary appropriately and hence maintaining high classification accuracy in dynamic or unstable environments. We demonstrate the scheme's effectiveness in both real and simulated streaming environments.

Keywords Logistic regression · Adaptive forgetting · Data streams · Classification

Mathematics Subject Classification (2000) 68T05 · 65C60

C. Anagnostopoulos (✉) · D. J. Hand
The Institute for Mathematical Sciences, Imperial College London,
53, Prince's Gate, London SW7 2PG, UK
e-mail: canagnos@imperial.ac.uk

D. K. Tasoulis · N. M. Adams · D. J. Hand
Department of Mathematics, Imperial College London,
South Kensington Campus, London SW7 2AZ, UK
e-mail: d.tasoulis@imperial.ac.uk

N. M. Adams
e-mail: n.adams@imperial.ac.uk

D. J. Hand
e-mail: d.j.hand@imperial.ac.uk

1 Introduction

Technological advances have led to the emergence of an increasing number of applications requiring analysis of streaming data, consisting of multiple indefinitely long and time-evolving sequences. Such applications include astronomic observations, telecommunications, environmental monitoring, retail banking, healthcare, and network monitoring. Streaming data analysis poses novel challenges to machine learning algorithms, prominent among which is the need for *online* inference, to allow for timely results without the need to store an ever-increasing data history. Moreover, in most streaming applications, the data sequences arise from highly dynamic, often unstable real-life processes, rendering untenable the standard assumption that current and future data come from the same distribution. In response, learning algorithms should be temporally adaptive in order to adapt to both smooth changes, often referred to as *drift*, as well as *abrupt changes* in the data-generating process (Gama et al. 2004).

In this paper we are concerned with temporally adaptive supervised classification for streaming data, which arises in a wealth of applications, such as target recognition (Copsey 2005), transaction fraud detection (Street and Kim 2001), spam filtering (Fdez-Riverola et al. 2007), user preference tracking (Crabtree and Soltysiak 1998), telecommunications (Black and Hickey 2004), and sensor networks (Copsey and Webb 2004). Techniques for streaming classification can also prove useful for applications where off-line estimation is tractable, but temporal adaptivity is essential to deal with non-stationarity (Hand 2006). Conversely, in certain domains where stationarity assumptions are tenable, incremental processing may be the only tractable solution for certain very large data sets (Street and Kim 2001).

A number of standard classification approaches have online formulations, including decision tree approaches (Carbonara and Borrowman 1998), support vector machines (Fung and Mangasarian 2002) and linear discriminant analysis (Zheng 2006). Certain among these also feature temporal adaptivity to deal with changing environments (Kuncheva 2004). Note that there are various modes of change that can be usefully distinguished and not all classifiers are equally affected by each type of change. A distinction of relevance to this discussion is that between the *diagnostic* and *sampling* paradigms for classification. In the former, the population is not modelled explicitly—instead, only the distribution of the class label conditional on the feature vector, $P(c \mid x)$, is estimated. As a result, such approaches may be relatively unaffected by changes in the population distribution or the relative class frequencies (Kuncheva 2004). In contrast, such changes would affect sampling approaches, where the full joint distribution $P(x, c)$ is modelled instead. Another important consideration when discussing types of change is to distinguish between *jumps* (or abrupt changes) and *drift* (or gradual change). Most approaches are tailored to one of these types of change. Indeed, one of the main strengths of our proposed approach is that it can handle either. Finally, measuring performance in the context of streaming classification is in itself challenging. For simulated environments, Alaiz-Rodrigez and Japkowicz (2008) recommend an appealing measure which we employ in our experiments.

The focus of the current work is logistic regression (McCullagh and Nelder 1989). An online algorithm for logistic regression based on a Bayesian state-space approach was described in Penny and Roberts (1999). This algorithm is capable of dealing with specific types of smooth change, in accordance with the state-space model structure that subsumes the estimation scheme. Similar ideas were employed in Whittaker et al. (2008). The state-space approach can be advantageous in contexts where expert insight can determine the type of dynamics underlying the data generating process. However, for a wide range of real systems it may be practically expedient to assume that the data generating process changes in ways too complex to model completely. Such systems may be still tracked successfully so long as obsolete information is discounted at the right rate. Forgetting may be implemented either via window-based and/or resetting algorithms (Aggarwal et al. 2004), or via smooth discounting of past information.

An important assumption underlying most streaming classifiers is that at each time-step, the class label arrives shortly after the feature vector, so that a predicted class label is of use in the meantime, but the true class label can still be accessed in time to update the classifier prior to the arrival of the next feature vector. This way of formulating supervised inference in the streaming context is inspired from the notion of *filtering* in signal processing and adaptive control (Haykin 1996), and is easily amenable to theoretical and experimental analysis. Moreover, there are several applications that can be accurately described in this manner. One application of interest is short-term forecasting of high-frequency financial or meteorological quantities (e.g., 'up' vs 'down' on financial indices, 'east' vs 'west' of wind direction change). We give one such example as part of our experimental results, using a dataset of electricity demand measurements. Nevertheless, in a range of real-world applications in need of adaptive classifier technology, class labels tend to arrive significantly later, and occasionally not at all (e.g., credit risk assessment (Yang 2007)) and credit card fraud detection (Hand et al. 2008). Relaxing the assumption of regular class arrival can be partially achieved via existing methodology, such as batch updates, semi-supervised methods (Chapelle and Zien 2005), or delayed information fusion methods (Tasoulis et al. 2007). An in-depth investigation of the effect of such methods on adaptive classifier performance lies beyond the scope of this paper. We take the view that ensuring accurate adaptive classification in an idealised sampling scenario is an essential first step towards studying the more realistic, irregular class arrival problem.

In this work, we propose a principled way of estimating logistic regression online in a temporally adaptive manner using forgetting factors. In a novel contribution, we employ a recently proposed online estimation scheme for the logistic classifier (Balakrishnan and Madigan 2008) in conjunction with two different adaptation methods for tuning the forgetting factors in data-dependent, online manners. Our tuning methods draw from the properties of the Taylor approximation itself, as well as from ideas on adaptive filter theory (Haykin 1996).

In the following Section we overview the quadratic approximation scheme that underlies our approach. In Sect. 3, we describe how it may be equipped with self-tuning forgetting factors. We then proceed in Sect. 4 to test the algorithm against both real and simulated datasets, establishing its ability to adapt to changing environments while maintaining high classification accuracy.

2 Framework

Classification is a supervised problem, where the training dataset consists of instances of a p-dimensional feature vector x and an associated class label c, where $c \in \{0, 1\}$. Logistic regression models the binary class label c as a function of x as follows:

$$\mathbb{E}(c \mid x) = P(c = 1 \mid x) = \Phi\big(\beta^T x\big) \tag{1}$$

where β is a p-dimensional regression parameter vector and Φ a link function satisfying $0 \leq \Phi(z) \leq 1$ for all z, e.g.,

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp\left\{-\frac{x^2}{2}\right\} \mathrm{d}x \tag{probit}$$

$$\Phi(z) = \frac{e^z}{1 + e^z} \tag{logistic}$$

In this paper we only consider logistic links although the generalisation to probit links is direct. For a given dataset of feature vectors $(x_i)_{i=1:t}$ and corresponding class labels $(c_i)_{i=1:t}$, we can consider the maximum likelihood estimator of β:

$$\hat{\beta}_t = \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{t} \log \big(c_i \Phi\big(\beta^T x_i\big)\big) + (1 - c_i)\big(1 - \Phi\big(\beta^T x_i\big)\big) \tag{2}$$

This optimisation problem is convex, but does not attain a closed-form solution, so that iterative methods must be employed (e.g., *Iteratively Weighted Least Squares*, (McCullagh and Nelder 1989)). This implies in particular that updating the maximum likelihood estimates upon the arrival of novel information cannot be performed in a closed-form, exact, recursive manner. Therefore, some approximative scheme must be employed. In this paper, we discuss a recent suggestion by Balakrishnan and Madigan (2008) based on a Taylor approximation to the log-likelihood. We outline the off-line version here and in the next section amend it to allow for online estimation.

Consider an off-line dataset $(x_i, c_i)_{i=1}^t$. Under the logistic regression model, the contribution to the log-likelihood of each datapoint (x_i, c_i) may be seen as a function of a one-dimensional variable $\beta^T x_i$, whose form will also depend on whether $c_i = 1$ or $c_i = 0$. Hence, we may write:

$$\log \mathcal{L}\left(\beta; (x_i, c_i)_{i=1}^t\right) = \sum_{i=1}^{t} f_i\left(\beta^T x_i\right) \tag{3}$$

where the subscript i in f_i denotes the dependence on c_i:

$$f_i\left(\beta^T x_i\right) = \begin{cases} \log \Phi\big(\beta^T x_i\big), & \text{if } c_i = 1 \\ \log\left(1 - \Phi\big(\beta^T x_i\big)\right), & \text{otherwise.} \end{cases}$$

Assume now that our current estimate of the parameters is given by $\tilde{\beta}$. In Balakrishnan and Madigan (2008), the authors point out that each f_i may be replaced by the first few terms of its Taylor expansion around our current 'location' $\tilde{z}_i = \tilde{\beta}^T x_i$:

$$f_i\left(\beta^T x_i\right) \approx f_i(\tilde{z}_i) + f_i'(\tilde{z}_i)\left(\beta^T x_i - \tilde{z}_i\right) + \frac{f_i''(\tilde{z}_i)}{2}\left(\beta^T x_i - \tilde{z}_i\right)^2$$

Collecting terms, we obtain a quadratic function on $\beta^T x_i$:

$$f_i\left(\beta^T x_i\right) \approx \frac{1}{2} a(\tilde{z}_i, c_i)\left(\beta^T x_i\right)^2 - b(\tilde{z}_i, c_i)\left(\beta^T x_i\right) \tag{4}$$

where the coefficients a and b are given by:

$$a(\tilde{z}_i, c_i) = \begin{cases} \frac{\Phi''(\tilde{z}_i)}{\Phi(\tilde{z}_i)} - \left(\frac{\Phi'(\tilde{z}_i)}{\Phi(\tilde{z}_i)}\right)^2, & \text{if } c_i = 1 \\ -\frac{\Phi''(\tilde{z}_i)}{1-\Phi(\tilde{z}_i)} - \left(\frac{\Phi'(\tilde{z}_i)}{1-\Phi(\tilde{z}_i)}\right)^2, & \text{if } c_i = 0 \end{cases} \tag{5}$$

$$b(\tilde{z}_i, c_i) = \begin{cases} -\frac{\Phi'(\tilde{z}_i)}{\Phi(\tilde{z}_i)} + \tilde{z}_i a(\tilde{z}_i, c_i), & \text{if } c_i = 1 \\ \frac{\Phi'(\tilde{z}_i)}{1-\Phi(\tilde{z}_i)} + \tilde{z}_i a(\tilde{z}_i, c_i), & \text{if } c_i = 0 \end{cases} \tag{6}$$

In the case of the logistic link function, these formulae simplify significantly, by making use of the following two facts:

$$\Phi'(z) = \Phi(z)(1 - \Phi(z)), \quad \Phi''(z) = \Phi(z)(1 - \Phi(z))(1 - 2\Phi(z))$$

The simplified formulae for a and b are then given by

$$a(\tilde{z}_i) = -\Phi(\tilde{z}_i)(1 - \Phi(\tilde{z}_i)) \tag{7}$$

$$b(\tilde{z}_i, c_i) = \Phi(\tilde{z}_i) - c_i + \tilde{z}_i a(\tilde{z}_i, c_i) \tag{8}$$

where notably the dependence of a on c_i cancels out during the simplification. Summing over all datapoints we obtain

$$\log \mathcal{L}\left(\beta; (x_i, c_i)_{i=1}^t\right) \approx \frac{1}{2}\beta^T \Psi(\tilde{\beta})\beta - \beta^T \theta(\tilde{\beta}) \tag{9}$$

where

$$\Psi(\tilde{\beta}) = \sum_{i=1}^{t} a\left(\tilde{\beta}^T x_i\right) x_i x_i^T, \quad \theta(\tilde{\beta}) = \sum_{i=1}^{t} b\left(\tilde{\beta}^T x_i, c_i\right) x_i \tag{10}$$

We expect that this quadratic form approximates the true log-likelihood well in the neighborhood of $\tilde{\beta}$. We may therefore obtain a better estimate, $\tilde{\beta}^{\star}$, by maximising this quadratic form with respect to β. This can be done in closed form:

$$\tilde{\beta}^{\star} = \underset{\beta}{\operatorname{argmax}} \left(\frac{1}{2} \beta^T \Psi(\tilde{\beta}) \beta - \beta^T \theta(\tilde{\beta}) \right) = \Psi^{-1}(\tilde{\beta}) \theta(\tilde{\beta}) \tag{11}$$

This scheme is guaranteed to converge, as discussed in Balakrishnan and Madigan (2008).

2.1 On-line estimation

We now drop the assumption of an off-line dataset, assuming instead that the data arrive sequentially, in pairs (x_t, c_t). We now seek to maintain the accuracy of our estimated model parameters without keeping the past history in memory, by retaining a finite-dimensional summary statistic. Although the task is supervised, the standard training-test dataset distinction does not readily apply in a streaming context. We hence need to make certain regularity assumptions about the way in which the data arrive. In particular, we assume that there is a certain small lag between the arrival of the feature vector x_t and the respective response, in this case its label, c_t, during which it is useful to have an estimated response. As discussed in the introduction, in the current framework we assume that this lag never exceeds the interval between two consecutive datapoints, so that we may employ the label c_t in updating our estimates before we predict c_{t+1}. We also assume that the feature vectors arrive at regular intervals.

The iterative scheme described in (11) involves at each step the propagation of the novel parameter estimate through the equations defining Ψ and θ, which requires us to revisit the entire history of the data stream. This is not feasible in online contexts. In response, we modify our scheme, following Balakrishnan and Madigan (2008). Denote our parameter estimate at time t by $\hat{\beta}_t$. Upon arrival of (x_{t+1}, c_{t+1}), we correctly compute the contribution of the last datapoint but do not back-propagate the current parameter estimate. This produces the following recursions:

$$\hat{\Psi}_{t+1} = \hat{\Psi}_t + a\big(\hat{\beta}_t^T x_{t+1}\big) x_{t+1} x_{t+1}^T \tag{12}$$

$$\hat{\theta}_{t+1} = \hat{\theta}_t + b\big(\hat{\beta}_t^T x_{t+1}, c_{t+1}\big) x_{t+1} \tag{13}$$

$$\hat{\beta}_{t+1} = P_{t+1} \theta_{t+1}, \qquad \text{where } P_{t+1} := \Psi_{t+1}^{-1} \tag{14}$$

Note that this scheme is *not* identical to its off-line counterpart, since it relies on a further approximation:

$$\Psi(\hat{\beta}_t) = \sum_{i=1}^{t+1} a_i \big(\hat{\beta}_t^T x_i\big) x_i x_i^T \approx \Psi_{t+1} = \sum_{i=1}^{t+1} a_i \big(\hat{\beta}_{i-1}^T x_i\big) x_i x_i^T$$

and similarly for θ. However, it is proven in Balakrishnan and Madigan (2008) to converge to the correct parameter estimates. Note that in Balakrishnan and Madigan (2008) the explicit solution (14) is not available to the authors, since they append an L1-norm penalty term to (11) to allow for variable selection. In the current work, we do not consider variable selection but the extension is direct.

3 Introducing forgetting

Subject to the approximations discussed above, the scheme (12)–(14) becomes an instance of recursive least squares. It is therefore very straightforward to incorporate forgetting factors, simply by replacing equations (13) and (12) with their 'forgetful' counterparts, wherein the contribution of the history is discounted at an exponential rate of $\lambda \leq 1$:

$$\hat{\Psi}_{t+1} = \lambda \hat{\Psi}_t + a_{t+1} x_{t+1} x_{t+1}^T, \quad a_{t+1} = a\big(\hat{\beta}_t^T x_{t+1}\big) \tag{15}$$

$$\hat{\theta}_{t+1} = \lambda \hat{\theta}_t + b_{t+1} x_{t+1}, \quad b_{t+1} = b\big(\hat{\beta}_t^T x_{t+1}, c_{t+1}\big) \tag{16}$$

The parameter estimate $\hat{\beta}_{t+1}$ is then given by $\hat{\Psi}_{t+1}^{-1} \hat{\theta}_{t+1}$ as in (14). A fixed value of 1 for λ allows estimation to depend on the entire data history, whereas for $\lambda < 1$, the importance of past data decays exponentially fast.

Observe that formula (14) involves a matrix inversion per timestep, an unstable and time-consuming operation. We can avoid this inversion by observing that $\hat{\Psi}_t$ is at each timestep perturbed by a rank-1 additive term. We may therefore apply the Matrix Inversion Formula (Harville 1997) to replace (14) by (17)–(20) below. The derivation is standard (Haykin 1996).

$$\hat{P}_{t+1} = \frac{1}{\lambda} P_t - \frac{1}{\lambda} a_{t+1} k_{t+1} x_{t+1}^T P_t \tag{17}$$

$$\hat{k}_{t+1} = \frac{P_t x_{t+1}}{\lambda + a_{t+1} x_{t+1}^T P_t x_{t+1}} \tag{18}$$

$$\hat{\xi}_{t+1} = b_{t+1} - a_{t+1} \beta_t^T x_{t+1} \tag{19}$$

$$\hat{\beta}_{t+1} = \hat{\beta}_t + k_{t+1} \hat{\xi}_{t+1} \tag{20}$$

3.1 Adaptive forgetting

Setting the value of the forgetting factor is a critical and non-trivial exercise. In the presence of drift, the optimal setting of the forgetting factor depends on the 'speed' of the drift, a quantity that is only estimable under strict, often unrealistic, assumptions on the type of drift (Ljung and Gunnarsson 1990). On the other hand, in the presence of abrupt changes, the optimal behaviour is to reset (Salgado et al. 1988), which cannot be approximated in a satisfactory manner via fixed forgetting. Data-dependent, hyperparameter-free methods for adapting the forgetting factor online are therefore essential in this context.

In this work, we adapt the stochastic gradient descent method for RLS filters suggested in Haykin (1996). The driving observation in that work is that the quality of fit of the current parameter estimates to the novel datapoint can be seen as a function of the current forgetting factor, λ_t. The goodness of fit function is clasically given by the one-step-ahead log-likelihood (Haykin 1996), which in this case is approximated by the following quadratic form:

$$\log \mathcal{L}\big(\hat{\beta}_t; x_{t+1}, c_{t+1}\big) \approx \frac{1}{2} a_{t+1} \big(\hat{\beta}_t^T x_{t+1}\big)^2 - b_{t+1} \hat{\beta}_t^T x_{t+1}, \text{ by (4) and (15)–(16)}$$

Note that in the RLS case, this equation is identical to a quadratic form without the need for approximation. We may then observe that this goodness-of-fit function depends on the forgetting factor, insofar as the latter is employed to determine the estimates $\hat{\beta}$. It is therefore possible to take the *gradient* of this quantity with respect to λ_t, and use it to guide its adaptation. This may be computed recursively, by pushing the partial derivative through the equations (17)–(20). This is an elementary calculation, described in Haykin (1996) for the RLS case and readily applicable in our case, too, since the update equations are of the same form. The resulting stochastic gradient descent algorithm proposes a way to select values for λ_t online by taking, at each timestep, a step in the direction of this gradient:

$$\lambda_{t+1} = \lambda_t + \eta \psi_t^T x_{t+1} \hat{\xi}_{t+1}, \tag{21}$$

$$S_{t+1} = \lambda_t^{-1}(I - k_{t+1}x_{t+1}^T) S_t (I - x_{t+1}k_{t+1}^T) + \lambda_t^{-1}(k_{t+1}k_{t+1}^T - P_{t+1}) \tag{22}$$

$$\psi_{t+1} = (I - k_{t+1}x_{t+1}^T)\psi_t + S_{t+1}x_{t+1}\hat{\xi}_{t+1} \tag{23}$$

where η is a step-size and $\psi_t^T x_{t+1}\hat{\xi}_{t+1}$ represents the direction of steepest descent in $\log \mathcal{L}((\hat{\beta}_t; x_{t+1}, c_{t+1})$.

Since λ_t is in effect a *learning rate*, this methodology can be considered in the context of adaptive learning rates for online learning (Saad 1998). Adaptive learning rate methods were mostly proposed in static, online contexts, to speed up convergence (Darken et al. 1992; Williams and Zipser 1989; Sutton 1992; Schraudolph 1999) but soon thereafter their adaptive properties were noticed and brought to bear on the problem of drifting optima (Benveniste et al. 1990). The method we employ here was suggested in Benveniste et al. (1990), and its empirical success in the Recursive Least Squares and Least Mean Squares contexts was established in Haykin (1996). An 'optimal tracking' result for the RLS case was shown in Kushner and Yang (1995), albeit under certain strong and hard to verify assumptions on the type of drift employed.

The quantity α is a meta-learning rate, that controls how fast λ_t can be allowed to change. Although any value for α would ensure that λ will be guided away from its default position at 1 and fit the characteristics of the drift, small values are preferable to counteract the volatility in our estimated gradient direction. To minimise the need for user input, we resort to the *RPROP* algorithm (Riedmiller and Braun 1993), a standard optimisation tool for tuning learning rates in gradient descent methods. Here, the main idea is to increase the step-size whenever the gradient consecutively points in the same direction, whereas when it oscillates (suggesting stochastic movement about equilibrium), the idea is to decrease the step-size and therefore allow convergence.

3.2 Tuned forgetting

Although our algorithm has the same form as RLS, there is a significant difference that we need to consider. In RLS, the contribution of the novel information is roughly of the same order at each iteration—a random sample from x and xx^T. In this case, the novel contribution is a complicated function of the novel datapoint, which does not aim to correct the sample mean and covariance, but rather the first and second order

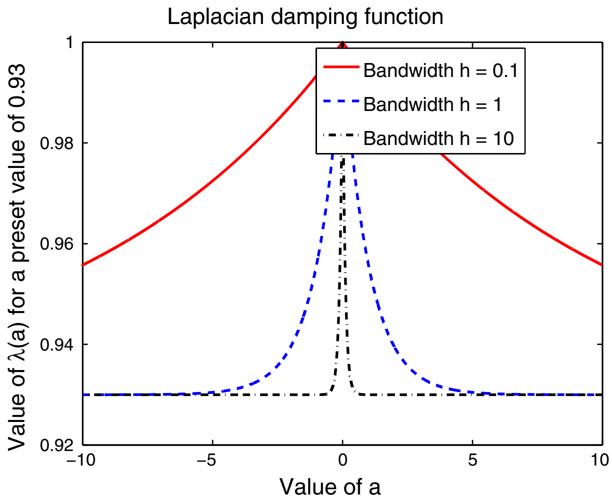


Fig. 1 The functional dependence of $\lambda(x)$ on x, for a given preset value $\lambda^\star$ and three choices of the bandwidth parameter

approximands to the true log-likelihood. These scale rather differently from timestep to timestep. In particular, the novel contributions are often very small (≈ 0), in which case a fixed forgetting factor shrinks estimated values unreasonably. In other words, to achieve a forgetting rate of $\lambda^\star$, we would preferably forget at that rate when $|a|$ is large (respectively for $|b|$), but not forget at all when $a \approx 0$. We therefore suggest that λ in equations (15) and (16) be a function of the magnitude of the respective novel contribution, according to the following 'Laplacian damping' formula:

$$\lambda(y) = \lambda^\star + (1 - \lambda^\star)\mathrm{e}^{-h|y|} \tag{24}$$

where $h \geq 0$ is a *bandwidth* parameter, determining the sensitivity of $\lambda(y)$ to y (Fig. 1). This argument invites us to replace equations (15)–(16) with

$$\hat{\Psi}_{t+1} = \lambda(\hat{a}_{t+1})\hat{\Psi}_t + \hat{a}_{t+1}x_{t+1}x_{t+1}^T \tag{25}$$

$$\hat{\theta}_{t+1} = \lambda(\hat{b}_{t+1})\hat{\theta}_t + \hat{b}_{t+1}x_{t+1} \tag{26}$$

A recursive update for $\hat{\beta}_{t+1}$ can then be obtained by suitably amending the derivation of the formulae (17)–(20):

$$\hat{P}_{t+1} = \frac{1}{\lambda(\hat{a}_{t+1})}P_t - \frac{1}{\lambda(\hat{a}_{t+1})}\hat{a}_{t+1}k_{t+1}x_{t+1}^T P_t \tag{27}$$

$$\hat{k}_{t+1} = \frac{P_t x_{t+1}}{\lambda(\hat{a}_{t+1}) + \hat{a}_{t+1}x_{t+1}^T P_t x_{t+1}} \tag{28}$$

$$\hat{\xi}_{t+1} = \hat{b}_{t+1} - \frac{\lambda(\hat{b}_{t+1})}{\lambda(\hat{a}_{t+1})}\hat{a}_{t+1}\beta_t^T x_{t+1} \tag{29}$$

$$\hat{\beta}_{t+1} = \frac{\lambda(\hat{b}_{t+1})}{\lambda(\hat{a}_{t+1})}\hat{\beta}_t + k_{t+1}\hat{\xi}_{t+1} \tag{30}$$

This is in fact a type of *adaptive* forgetting, in the sense that larger corrections will lead to faster forgetting rates. In this sense, the baseline forgetting factor $\lambda^\star$ should be understood more as a *lower bound* on the values that the forgetting factor can take, rather than a user-specified rate of discounting past information. The desired effect then is to define a scheme with two hyperparameters, the baseline forgetting factor $\lambda^\star$ and the bandwidth h, whose tuning is significantly easier and less context specific than the tuning of the single forgetting factor featured in a fixed forgetting scheme. Our experimental results support this statement.

To distinguish this adaptive scheme from the gradient-based algorithm described earlier, we refer to it as *tuned forgetting*.

4 Experiments

In this section we assess the empirical performance of the proposed streaming classification techniques against different types of simulated change, as well as two real datasets. We begin by describing our choice of performance metric.

As mentioned in the Introduction, assessing classification performance in streaming contexts is in itself a non-trivial matter (Hand 1997; Adams and Hand 2000). However, in simulated settings we can get an accurate intuitive performance measure from the instantaneous error of each classifier, since we can at each time t iteratively sample from the data distributions. Not only that but we can contrast this error rate to the true one computed from a classifier that is given the actual data distribution.

As such, in the following experimental results we examine the performance of five different schemes. Among these, we employ standard logistic regression as a benchmark and estimate it using an off-the-shelf Fletcher–Reeves conjugate gradient algorithm (Press et al. 1988). The estimation occurs off-line, employing the complete dataset. This scheme is denoted *Off-line*. Next we use the online Taylor approximation scheme described in Sect. 2, which we denote as *On-line*. The third scheme is equipped with a single, fixed forgetting factor, as described in Sect. 3. Different algorithms can be derived from this scheme by using different values for the forgetting factor. We denote this family of algorithms by $F - \lambda$. Next, we denote the adaptive forgetting approach, as described in Sect. 3.1, *AF*. Finally, "tuned forgetting', *TF* for short, represents the scheme described in Sect. 3.2. This scheme is again represented as a family of algorithms $TF - \lambda^\star$, where $\lambda^\star$ is the baseline forgetting factor employed in the Laplacian damping formula. At each experiment the true error rate will be denoted with the label *True*.

4.1 Static environments

We begin by investigating the behaviour of our streaming algorithms in static environments. In this case, we employ the following simulation engine:

$$P(c_t = l_j) = 0.5, \quad t = \{1, \ldots, n, \ldots\} \quad \text{and} \quad j = 1, 2$$
$$(x_t \mid c_t = l_j) \sim N(\mu^{(j)}, \Sigma^{(j)}),$$

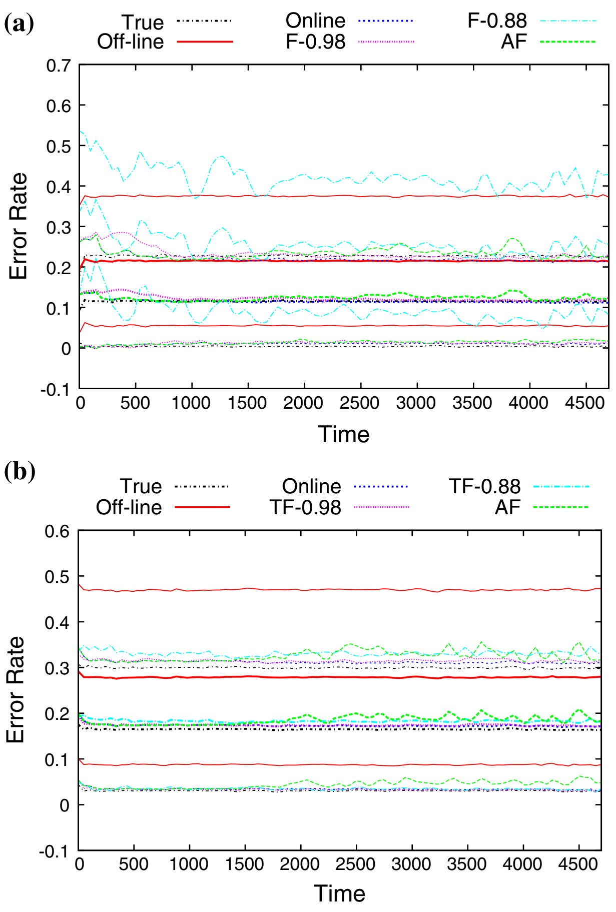


Fig. 2 The performance measured in terms of error rate for static environments for the $F - \lambda$ (**a**) and $TF - \lambda$ (**b**), schemes

Note that $\mu^{(j)}$, $\Sigma^{(j)}$ do not have a time index, indicating that they remain unchanged over time. For each simulation instance, we generate $\mu^{(j)}$ from a uniform distribution on $[-2, 2]^p$, where p is the dimension of the data, and sample $\Sigma^{(j)}$ from a Wishart distribution with q degrees of freedom, $\Sigma^{(j)} \sim \mathcal{W}_p(\mathbf{I}, q)$. We set q to the value $5p$, ensuring well-conditioned but fairly generic covariance matrices.

The results with respect to the error rates of each method are exhibited in Fig. 2 for the case of $p = 5$. These results are averaged over 100 different simulations; at each time tick the mean error rate over the 100 runs is plotted. We also plot the observed variance of the error rate values (mean $\pm$ 0.5 variance) with lines of small width. Throughout these simulations we moderately smooth the series for readability using a standard splines smoother. This does not affect relative performance.

Most of the algorithms exhibit very low error rate almost equal to the true error rate. The Off-line, On-line and AF algorithms are used in both Fig. 2a and b for reference. Note, however, that the performance of these three methods differs from one plot to another, as each plot employs a different simulation instance. The $F - 0.88$ algorithm performs poorly, since the forgetting factor value is too low to allow convergence to the correct solution. The offline method does not perform well on either simulation, presumably due to poor convergence of the underlying gradient descent algorithm. This effect could potentially have been mitigated by more sophisticated offline gradient descent methods, an issue which we have not explored further here. Interestingly, the AF algorithm selects a value for the forgetting factor close to 1, which agrees with the optimal value in this case. This sensible behaviour on static data is reassuring.

4.2 Generating evolving data streams

In order to test the proposed algorithm against both gradually and abruptly evolving data streams, we have employed two types of simulation engines, each modelling a certain type of change in the conditional distribution of the data X, given the class label C. We proceed to describe the simulation engines in some detail.

4.2.1 $DGEN_{\text{drift}}$: gradually drifting data streams

This data generation mechanism simulates gradual change in $P(X \mid C)$. To achieve this, the mean vectors of both densities are perturbed at each timestep according to a damped random walk:

$$\mu_t^{(j)} = 0.99\mu_{t-1}^{(j)} + \epsilon_t, \quad \epsilon \sim N(0, \sigma^2) \tag{31}$$

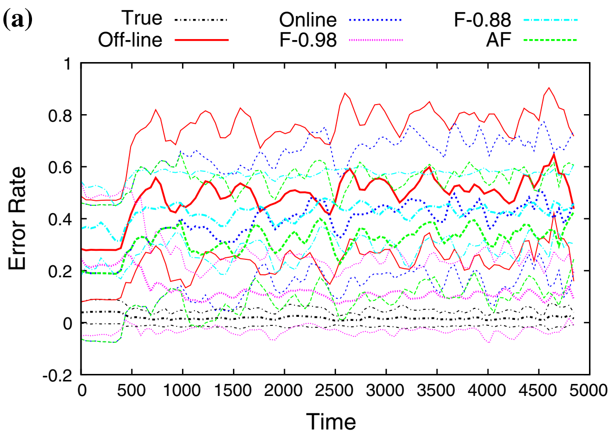
A damping factor of 0.99 is customary in such contexts, to ensure that the movement of the means does not overly diverge over large time horizons, whereas the parameter σ controls the 'speed' of the drift and typically is restricted to the range [0, 1]. The initial values $\mu_0^{(j)}$ are sampled uniformly as before.

To generate a dynamic sequence of covariance matrices is a harder problem, as auto-regressive processes on positive-definite invertible matrices are hard to tune for large dimensions and can very quickly generate unstable, ill-conditioned matrices. Instead, we employ the following simple and easy to control method. For each class j we randomly generate two covariance matrices $\Sigma_{\text{start}}^{(j)}$ and $\Sigma_{\text{end}}^{(j)}$ and then move smoothly between them in a convex manner. Namely, for a sequence of length n, we set

$$\Sigma_t^{(j)} = \frac{n-t+1}{n}\Sigma_{\text{start}}^{(j)} + \frac{t-1}{n}\Sigma_{\text{end}}^{(j)} \tag{32}$$

This technique can equally well allow us to consider 'faster' covariance drift: we may divide our timescale into regular intervals by introducing 'breakpoints', $t_1, \ldots, t_i, \ldots$, generate random covariance matrices to be placed at each breakpoint and then move smoothly between these change-points in the manner described above. This idea essentially transforms the difficult problem of generating *smoothly changing* covariance matrices to the problem of generating arbitrary covariance matrices, which can be addressed easily via the use of the Wishart distribution as described earlier.

(a)



(b)

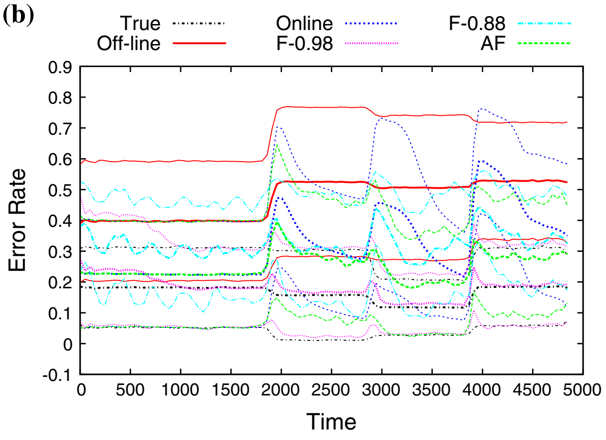


Fig. 3 The error rate for $F - \lambda$ methods for $DGEN_{\text{drift}}$ (**a**), and $DGEN_{\text{abrupt}}$ (**b**) generated data

4.2.2 $DGEN_{\text{abrupt}}$: abruptly changing data streams

To test the ability of the algorithm to deal with abrupt changes in the data distribution we construct data sets by entirely changing the density $P(X, C)$ every τ time steps, where τ is the parameter that in this case controls the "speed of change". At each changepoint, the means and covariances are generated randomly in the manner outlined earlier. Throughout the simulations in this section, τ is set to 1,000.

4.2.3 Performance

In Fig. 3 the performance of $F - \lambda$ is illustrated for the two different data stream generating mechanisms. The true error rate is also plotted for comparison purposes. Note that the true error rate varies with time both in the case of smooth and abrupt change. This is explained by the fact that the data generating system is changing with time, which can affect the difficulty of the classification problem. The variation is,

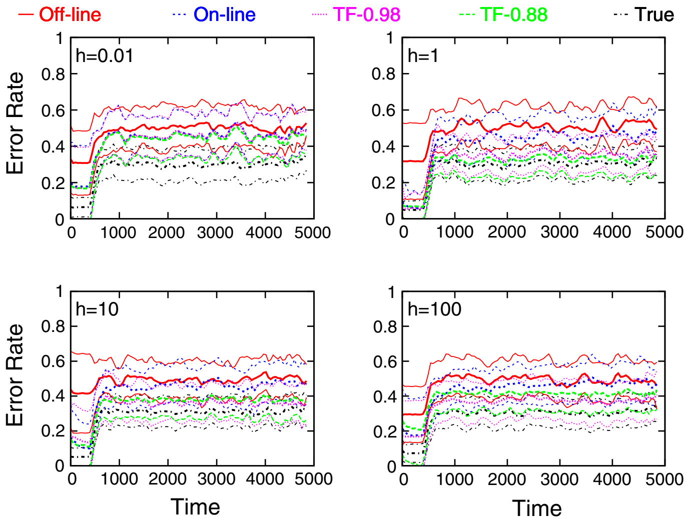


Fig. 4 Performance in terms of error rate for tuned forgetting using the $DGEN_{\text{drift}}$ data and different bandwidths h

however, small, and negligible for purposes of comparison to other schemes. For both $DGEN_{\text{abrupt}}$ and $DGEN_{\text{drift}}$, the $F - 0.98$ algorithm performs significantly better than all the other algorithms, suggesting that a high forgetting value is optimal in this case. Interestingly, the *On-line* algorithm exhibits a (slow) adaptation to the changing data streams, although it was not designed for such a task. The $F - 0.88$ algorithm improves on both the *Off-line* and *On-line* approaches for the $DGEN_{\text{abrupt}}$ case, but with somewhat unstable behaviour and large variability. This behaviour does not allow the algorithm to improve even on the *On-line* approach with the $DGEN_{\text{drift}}$ generated data as shown in Fig. 3b. The AF approach provides a compromise between these two approaches for both the $DGEN_{\text{abrupt}}$ and $DGEN_{\text{drift}}$ case. It does not perform as well as $F - 0.98$, but it improves on $F - 0.88$ and *On-line* significantly.

Besides the value of the forgetting factor, the tuned forgetting schemes feature another parameter of interest, namely the bandwidth, h, of the Laplacian damping formula (24). We explore the dependence of the scheme's performance on this parameter in Figs. 4 and 5. The results shown were obtained for $DGEN_{\text{abrupt}}$ and $DGEN_{\text{drift}}$, respectively. The On-line, Off-line algorithms are depicted here, as reference points. In both cases we can see that very small choices of bandwidth (e.g., 0.01) enforce a 'forget nothing' tactic which aligns the scheme's performance to that of the On-line algorithm. For values $h \geq 1$, a tradeoff between the baseline forgetting factor and the bandwidth is observed: aggressive tuning performs best when the forgetting factor is set correspondingly high, otherwise it invites instability; whereas moderate tuning, achieved by low bandwidth, must be counteracted by correspondingly very low values for the baseline forgetting factor to achieve comparable performance. Consequently, $TF - 0.98$ with $h = 100$ and $TF - 0.88$ with $h = 1$ perform comparably. This is reassuring: it seems as long as we avoid overly large values for h, the baseline forgetting factor can be set conveniently low, allowing the possibility for drastic forgetting

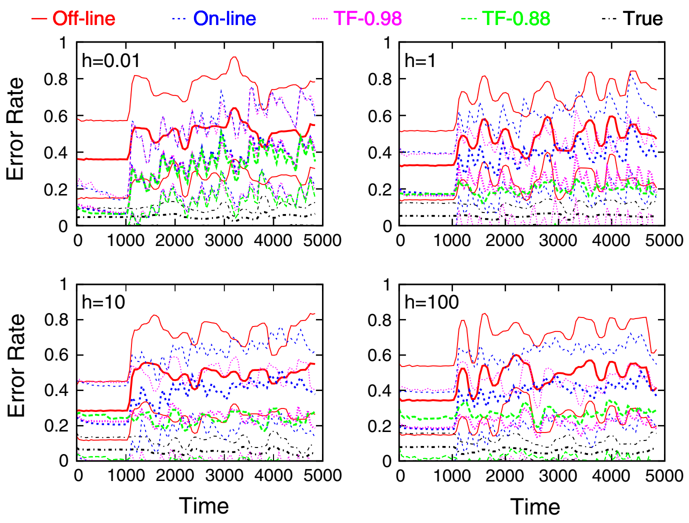


Fig. 5 Performance in terms of error rate for tuned forgetting using the $DGEN_{\text{abrupt}}$ data and different bandwidths h

if necessary, while at the same time preserving good performance in slow systems where a higher fixed forgetting factor would be optimal.

4.3 Real datasets

We now proceed to test our method against a real dataset, which we denote In this section we examine the performance of the algorithms with two different real datasets. The first dataset, which we refer to as D_{elec}, is described in Harries (1999), and holds information for the Australian New South Wales (NSW) Electricity Market. It contains 45,312 instances dated from May 1996 to December 1998, with each record referring to a period of 30 min. These records have five fields: two time stamp indicators (day of week, time), and three covariates, NSW electricity demand, Vic electricity demand, and scheduled electricity transfer between states. We only employ the covariates in our classification rule and use time stamp information to ensure the dataset is in order. The class label associated with each record is a binary indicator that identifies the one-step-ahead price change relative to a moving average of the last 24 h ('price has gone up' vs 'price has gone down'). From the total 45,312 we have used the 27,502 records that do not have any missing values.

An appealing property of this dataset is that it is expected to contain drifting data distributions since, during the recording period, the electricity market was expanded to include adjacent areas. This allowed for the production surplus of one region to be sold in the adjacent region, which in turn dampened price levels. Moreover, it has already been used in similar papers (Baena-Garcia et al. 2006) so it provides a benchmark of sorts. Note finally that in what follows we measure performance via *correct classification rate*.

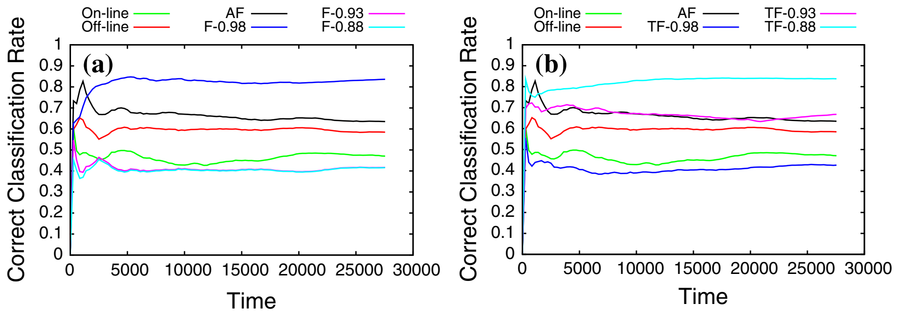


Fig. 6 The performance for D_{elec} using $F - \lambda$ (**a**) and $TF - \lambda$ (**b**) schemes

The results for D_{elec} are shown in Fig. 6. As shown the On-line algorithms performs quite poorly. The $F - 0.98$ and $TF - 0.88$ algorithms seems to provide the best results overall. Both of these scheme, however, need careful selection of the forgetting factors to perform well. On the other hand, AF with minimal user intervention may fail to produce the best results, but still manages to improve on both the Off-line scheme and the On-line scheme without forgetting.

Next we illustrate the application of the proposed streaming methodology for the automated detection of tumors in colonoscopic images. On-line classification of this imaging data has the potential to increase early detection of colorectal cancer precursors, and assist the physician to make early diagnosis of colorectal cancer (Magoulas et al. 2004b).

During colonoscopy the physician interprets physical surface properties of the tissue. The main difficulties in this task arise from high variability of the resulting images due to resolution change, noise and variable perceptual conditions (shading, shadows, lighting conditions, and reflections). In the dataset $Dset_{\text{Colon}}$, used here, each of four video frames contains textures of both normal and abnormal tissue. A rolling window of size 16×16 pixels is passed over each video frame and used as the basis of a feature extraction methodology (Haralick 1979). This results in a stream of 16×1 feature vectors and respective class labels, designating correspondence, or not, to a window containing tumor pixels. A detailed description of the feature extraction method can be found in Magoulas et al. (2004a). In this particular application, the class labels are available to us offline, and provided by the physician. The dataset is pre-processed to ensure that the class labels become available in the right rate and order. In this way we enforce our assumption of regular class arrival, whose feasibility and usefulness were discussed in the Introduction. The *co-occurrence matrices* method has been used for feature extraction (Haralick 1979), which represents the spatial distribution and the dependence of the gray levels within a local area using an image window of size 16 by 16 pixels. Finally, each feature vector contained sixteen elements and a class label that designated a window containing tumor pixels.

The results are shown in Fig. 7. Similarly to the previous cases there are two specific 'forgetful' schemes that outperform all others, namely $F - 0.93$ and $TF - 0.88$. It is interesting to note that $F - 0.98$ performed poorly, which is exactly the opposite to the D_{elec} case. This serves to further emphasize the benefit of self-tuned forgetting factors.

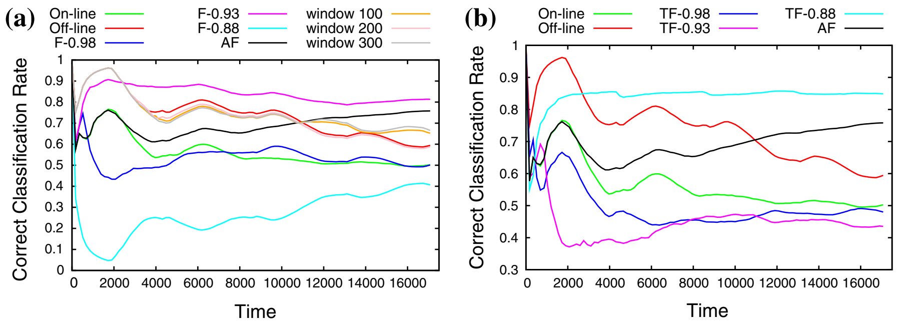


Fig. 7 The performance for D_{colon} using $F-\lambda$ (**a**) and $TF-\lambda$ (**b**) schemes

In particular, tuned forgetting achieves a dramatic improvement in performance from $F-0.88$ to $TF-0.88$ while retaining the same baseline level of forgetting. Moreover, it is important to point out that $TF-0.88$ performs consistently well throughout all experiments, in contrast to $F-0.88$ whose ranking varies wildly from one experiment to another. This evidence acts in support of the claim that tuned forgetting with a low baseline forgetting factor has self-tuning properties with minimal need for user input, notwithstanding the presence of two tuning hyperparameters.

The AF adaptive forgetting method in this case seems to go through a relatively lengthy burn-in period, but finally manages to produce very good performance. For comparison purposes in Fig. 7a we evaluate the performance of logistic regression re-trained continuously over a rolling window of data of different sizes (100, 200 and 300 pixels). As shown all different window based methods perform similarly to the Off-line method, offering no real advantage despite the fact that they are discounting old information. We suggest this to be a result of the fact that information discounting occurring via data selection produces unstable estimation, because of small sample sizes. This effect is mitigated by smooth alternatives, such as forgetting factors.

Our experimental results suggest that it is generally difficult to pre-decide on the appropriate forgetting factor value. Tuning has a clear advantage over simple, fixed forgetting, allowing us to employ low forgetting factors without destabilising the Taylor approximation scheme too much. This is particularly true in real data, where model mis-specification can be expected to further confuse the approximation scheme. However, we can still find cases where such a methodology becomes unstable (see Figs. 4, 5). The adaptive forgetting mechanism provides a middle ground approach, consistently outperforming the Off-line and On-line schemes, and achieving a compromise between further performance improvement and minimal user intervention.

5 Conclusion

In this paper we are concerned with temporally adaptive supervised classification for streaming data, which arises in a wealth of applications varying from target recognition to sensor networks. Techniques for streaming classification can also prove useful for applications where off-line estimation is intractable due to the dataset size (Street and Kim 2001).

Data streams often arise from time-varying, non-stationary systems. Our approach is based on the premise that such systems may be tracked successfully so long as obsolete information is discounted at the right rate, via the use of forgetting factors. In a novel contribution, we describe an estimation scheme for logistic regression equipped with temporal adaptivity. Additionally, we design algorithmic schemes that tune the forgetting rate in an online, data-dependent manner and are hence able to deal with both abrupt and smooth change drift. To achieve this, we combine an online estimation scheme for the logistic classifier based on a quadratic approximation to the log-likelihood (Balakrishnan and Madigan 2008) with a stochastic gradient descent adaption mechanism. The proposed methods are evaluated on appropriately simulated data as well as real datasets.

It is interesting to note that our approach can be shown to generalise to the full Generalized Linear Model family, as well as, for unconditional modelling, to the full Exponential Family of distributions. We intend to explore this generalization in future work. We also intend to attempt to relax the assumptions of regular sampling and negligible lag in the class labels, allowing us to offer realistic solutions to a wider range of real-life streaming classification problems. Finally, a complete Matlab package, implementing this, and other streaming classification technology, is under development.

Acknowledgments The work of Christoforos Anagnostopoulos was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) project and is jointly funded by a BAE Systems and the EPSRC (Engineering and Physical Research Council) strategic partnership, under EPSRC grant EP/C548 051/1. The work of David Hand was partially supported by a Royal Society Wolfson Research Merit Award.

References

Adams NM, Hand DJ (2000) Improving the practice of classifier performance assessment. Neural Comput 12(2):305–311

Aggarwal, CC, Han J, Wang J, Yu PS (2004) On demand classification of data streams. In: KKD'04 proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 503–508

Alaiz-Rodrigez R, Japkowicz N (2008) Assessing the impact of changing environments on classifier performance. In: Advances in artificial intelligence, Lecture Notes in Computer Science, vol 5032/2008. Springer, Heidelberg, pp 13–24

Baena-Garcia M, del Campo-Avila J, Fidalgo R, Bifet A, Gavalda R, Morales-Bueno R (2006) Early drift detection method. In: ECML PKDD 2006 workshop on knowledge discovery from data streams, pp 77–86

Balakrishnan S, Madigan D (2008) Algorithms for sparse linear classifiers in the massive data setting. J Mach Learn Res 9:313–337

Benveniste A, Priouret P, Métivier M (1990) Adaptive algorithms and stochastic approximations. Springer, New York

Black M, Hickey R (2004) Learning classification rules for telecom customer call data under concept drift. Soft Comput 8(2):102–108

Carbonara L, Borrowman A (1998) A comparison of batch and incremental supervised learning algorithms. In: Principles of data mining and knowledge discovery, LNCS, vol 1510, pp 264–272

Chapelle O, Zien A (2005) Semi-supervised classification by low density separation. In: Proceedings of the tenth international workshop on artificial intelligence and statistics, vol 2005

Copsey KD (2005) Automatic target recognition using both measurements from identity sensors and motion information from tracking sensors. In: Automatic target recognition, Proc SPIE, vol 5807, pp 273–283

Copsey K, Webb A (2004) Classifier design for population and sensor drift. In: structural, syntactic and statistical pattern recognition, LNCS, vol 3138, pp 744–752
Crabtree B, Soltysiak SJ (1998) Identifying and tracking changing interests. Int J Digit Libr 2(1):38–53
Darken C, Chang J, Moody J (1992) Learning rate schedules for faster stochastic gradient search. In: Neural networks for signal processing 2—Proceedings of the 1992 IEEE workshop
Fdez-Riverola F, Iglesias EL, Diaz F, Mendez JR, Corchado JM (2007) SpamHunting: an instance-based reasoning system for spam labelling and filtering. Decis Support Syst 43:722–726
Fung G, Mangasarian OL (2002) Incremental support vector machine classification. In: Proceedings of the second SIAM international conference on data mining, pp 247–260
Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. SBIA Brazilian symposium on artificial intelligence, pp 286–295
Hand DJ (1997) Construction and assessment of classification rules. Wiley, London
Hand DJ (2006) Classifier technology and the illusion of progres (with discussion). Stat Sci 21(1):1–34
Hand DJ, Whitrow C, Adams NM, Juszczak P, Weston DJ (2008) Performance criteria for plastic card fraud detection tools. J Oper Res Soc 59:956–962
Haralick RM (1979) Statistical and structural approaches to texture. Proc IEEE 67(5):786–804
Harries M (1999) Splice-2 comparative evaluation: electricity pricing. Tech. rept. University of New South Wales, School of Computer Science and Engineering
Harville DA (1997) Matrix algebra from a statistician's perspective. Springer, Berlin
Haykin S (1996) Adaptive filter theory. Prentice-Hall, Upper Saddle River
Kuncheva LI (2004) Classifier ensembles for changing environments. In: Multiple classifier systems, LNCS, vol 3077, pp 1–15
Kushner HJ, Yang J (1995) Analysis of adaptive step size sa algorithms for parameter tracking. IEEE Trans Automat Contr 40(8):1403–1410
Ljung L, Gunnarsson S (1990) Adaptation and tracking in system identification—a survey. Automatica 26(1):7–21
Magoulas GD, Plagianakos VP, Vrahatis MN (2004a) Neural network-based colonoscopic diagnosis using on-line learning and differential evolution. Appl Soft Comput 4:369–379
Magoulas GD, Plagianakos VP, Tasoulis DK, Vrahatis MN (2004b) Tumor detection in colonoscopy using the unsupervised k-windows clustering algorithm and neural networks. In: Proceedings of fourth European symposium on biomedical engineering, Patras, Greece, pp 152–163
McCullagh P, Nelder JA (1989) Generalized linear models. Chapman & Hall/CRC, London, UK
Penny WD, Roberts SJ (1999) Dynamic logistic regression. In: International joint conference on neural networks, vol 3, pp 1562–1567
Press W, Teukolsky S, Vetterling W, Flannery B (1988) Numerical recipes in C. Cambridge University Press, Cambridge, UK
Riedmiller M, Braun H (1993) A direct adaptive method for faster backpropagation learning: theRPROP algorithm. In: IEEE international conference on neural networks, pp 586–591
Saad D (1998) On-line learning in neural networks. Cambridge University Press, Cambridge, UK
Salgado ME, Goodwin GC, Middleton RH (1988) Modified least squares algorithm incorporating exponential resetting and forgetting. Int J Contr 47(2):477–491
Schraudolph NN (1999) Local gain adaptation in stochastic gradient descent. Artif Neural Netw 470:569–574
Street WN, Kim Y (2001) A streaming ensemble algorith (SEA) for large-scale classification. In: KKD'01: proceedings of the seventh acm SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 377–382
Sutton RS (1992) Gain adaptation beats least squares? In: Proceedings of the seventh Yale workshop on adaptive and learning systems
Tasoulis DK, Adams NM, Hand DJ (2007) Selective fusion of delayed measurements in filtering. In: IEEE international workshop on machine learning for signal processing (IEEE MLSP), pp 336–341
Whittaker J, Whitehead C, Somers M (2008) A dynamic scorecard for monitoring baseline performance with application to tracking a mortgage portfolio. J Oper Res Soc 58(11):911–921
Williams RJ, Zipser D (1989) A learning algorithm for continually running fully recurrent neural networks. Neural Comput 1:270–280
Yang Y (2007) Adaptive credit scoring with kernel learning methods. Euro J Oper Res 183(3):1521–1536
Zheng W (2006) Class-incremental generalized discriminant analysis. Neural Comput 18:979–1006