Imperial College London

Efficient Streaming Classification Methods

Niall M. Adams[1], Nicos G. Pavlidis[2], Christoforos Anagnostopoulos[3], Dimitris K. Tasoulis[1]

[1]Department of Mathematics
[2]Institute for Mathematical Sciences
Imperial College London

[3]Statistical Laboratory
University of Cambridge

July 2010

Contents

- Streaming data and classification
- Our approach: incorporate self-tuning forgetting factors
- Illustrations and examples
- Issues (current work in progress)
  - *free* parameters
  - non-linearity, regularisation, over-fitting
  - label timing
- Conclusion

Research support by the EPSRC/BAe funded ALADDIN project:
www.aladdinproject.org

Streaming Data I

A data stream consists of a sequence of data items arriving at high frequency, generated by a process that is subject to unknown changes (generically called drift).

Many examples, often financial, include:

- credit card transaction data (6000/s for Barclaycard Europe)
- stock market tick data
- computer network traffic

The character of streaming data calls for algorithms that are

- efficient, one-pass - to handle frequency
- adaptive - to handle unknown change

Streaming Data II

A simple formulation of streaming data is a sequence of p-dimensional vectors, arriving at regular intervals

$$\ldots, x_{t-2}, x_{t-1}, x_t$$

where $x_i \in \mathbb{R}^p$.
In a real application, with credit card transactions, the variables include: transaction value and time, location, card response codes.

Since we are concerned with K-class classification, need to accommodate a class label. Thus, at time t we can conceptualise the label-augmented streaming vector $y_t = (C_t, x_t)'$, where $C_t \in \{c_1, c_2, \ldots, c_k\}$.

However, in real applications C_t arrives at some time $s > t$, and the streaming classification problem is concerned with predicting C_t on the basis of x_t in an efficient and adaptive manner.

Streaming Data and Classification

Implicit assumption: single vector arrives at any time.

Assumption common in literature, which we use, is that the data stream is structured as

$$\dots, (C_{t_3}, x_{t_2}), (C_{t_2}, x_{t_1}), (C_{t_1}, x_t),$$

That is, the class-label arrives at the next tick.

We will treat the streaming classification problem as: predict the class of x_t, and adaptively (and efficiently) update the model at time x_{t+1}, when C_t arrives.

This is naive, but the problem is challenging even formulated thus. Will return to label timing later.

Streaming Data and Classification

Can use the usual formulation for classification

$$P(C_t|x_t) = \frac{p(x_t|C_t)P(C_t)}{p(x_t)} \tag{1}$$

and construct either

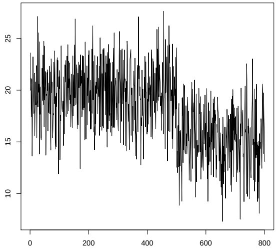
- Sampling paradigm classifiers, focusing on class conditional densities
- Diagnostic paradigm classifiers, directly seeking the posterior probabilities of class membership

Note the we will usually restrict attention to the $K = 2$ class problem.

Eq.1 also illustrates where changes can happen: the prior, $P(C_t)$, the class conditionals, $p(x_t|C_t)$, or both.

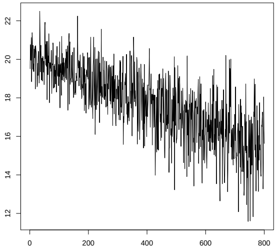
Notional drift types

1. Jump



(in mean)

2. Gradual change



(in mean and variance) Trend, seasonality etc.

Drift: Real Examples

Consumer credit classification (conditionals)

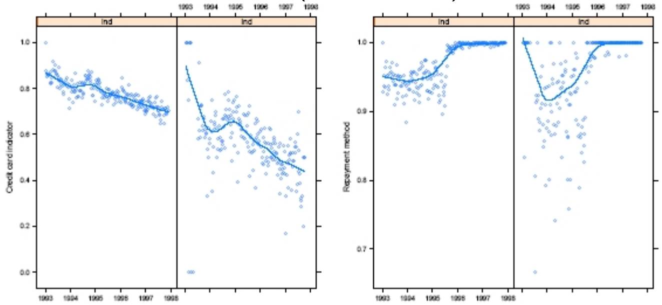


Figure 1: (a): Weekly averages for credit card indicator. (b): Weekly averages for repayment method indicator. Each plot includes good risk (left) and bad risk (right).

Consumer credit classification (prior)

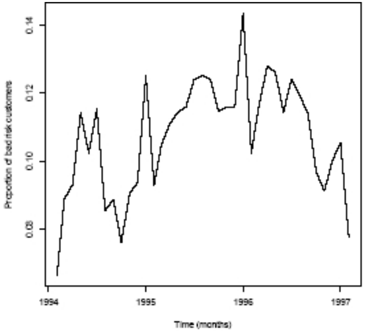


Figure 2: Proportion of bad risk accounts, by month, over the entire observation period.

Methods

A variety of approaches for streaming classification have been proposed, including

- Data selection approaches with standard classifiers. Most commonly, use of a fixed or variable size window of most recent data. But how to determine size in either case?
- Ensemble methods. One example is the adaptive weighting of ensemble members changing over time. This category also includes learning with expert feedback.

Example: Window method happens in consumer credit scoring - not for efficiency, but because the populations have changed.

Forgetting-factor methods

We are interested in modifying standard classifiers to incorporate forgetting factors - parameters that control the contribution of old data to parameter estimation.

We adapt ideas from adaptive filter theory, to tune the forgetting factor automatically.

Simplest to illustrate with an example: consider computing the mean vector and covariance matrix of a sequence of *n* multivariate vectors. Standard recursion

$$m_t = m_{t-1} + x_t,\ \hat{\mu}_t = m_t/t,\ m_0 = 0$$
$$S_t = S_{t-1} + (x_t - \hat{\mu}_t)(x_t - \hat{\mu}_t)^T,\ \hat{\Sigma}_t = S_t/t,\ S_0 = \mathbf{0}$$

For vectors coming from a non-stationary system, simple averaging of this type is biased.

Knowing precise dynamics of the system gives chance to construct optimal filter. However, not possible with streaming data (though interesting links between adaptive and optimal filtering).

Incorporating a forgetting factor, $\lambda \in (0, 1]$, in the previous recursion

$$
\begin{aligned}
n_t &= \lambda n_{t-1} + 1,\ n_0 = 0 \\
m_t &= \lambda m_{t-1} + x_t,\ \hat{\mu}_t = m_t / n_t \\
S_t &= \lambda S_{t-1} + (x_t - \hat{\mu}_t)(x_t - \hat{\mu}_t)^T,\ \hat{\Sigma}_t = S_t / n_t
\end{aligned}
$$

λ down-weights old information more smoothly than a window. Denote these forgetting estimates as $\hat{\mu}_t^\lambda$, $\hat{\Sigma}_t^\lambda$, etc.

n_t is the effective sample size or memory. $\lambda = 1$ gives offline solutions, and $n_t = t$. For fixed $\lambda < 1$ memory size tends to $1/(1 - \lambda)$ from below.

Setting λ

Two choices for λ, fixed value, or variable forgetting, λ_t. Fixed forgetting: set by trial and error, change detection, etc (cf. window).

Variable forgetting: ideas from adaptive filter theory suggest tuning λ_t according to a local stochastic gradient descent rule

$$\lambda_t = \lambda_{t-1} - \alpha \frac{\partial \xi_t^2}{\partial \lambda}, \;\; \xi_t\text{: residual error at time } t,\ \alpha \text{ small} \qquad (2)$$

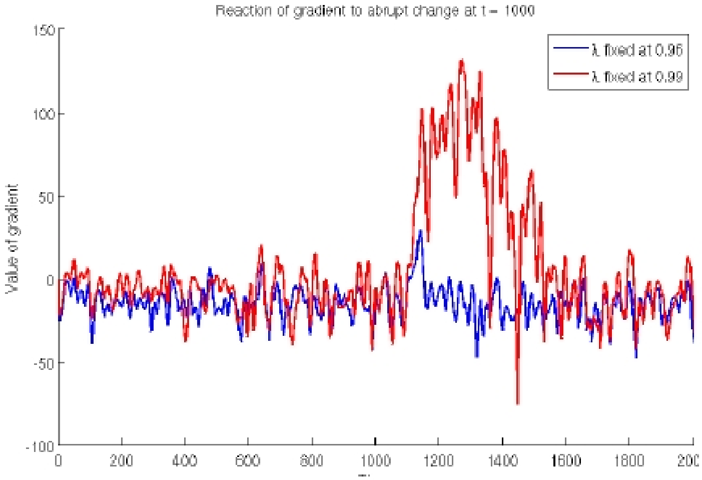
Efficient updating rules can implemented via results from numerical linear algebra ($O(p^2)$).

Performance very sensitive to α. Very careful implementation required, including bracket on λ_t and selection of learning rate α.

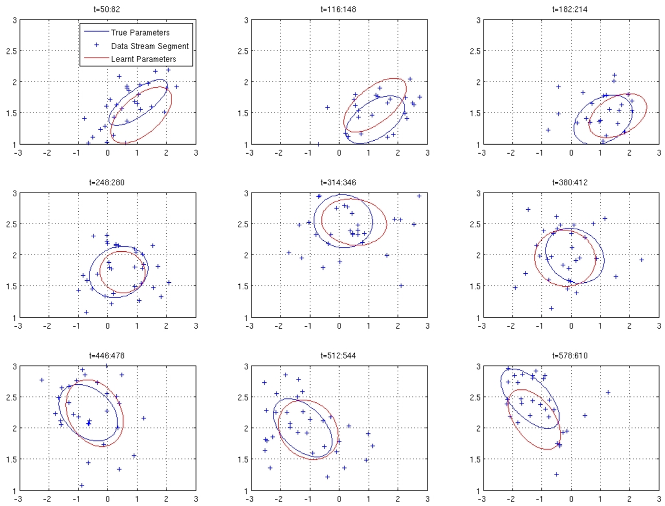
Framework provides an adaptive means for balancing old and new data. Note slight hack in terms of interpretation of λ_t.

Tracking illustrations

Does fixed forgetting respond to an abrupt change?
5D Gaussian, two choices of λ, change in σ_{23}: gradient



Tracking mean vector and covariance matrix in 2D.



Adaptive-Forgetting Classifiers

Our recent work involves incorporating these self-tuning forgetting factors in

- Parametric
  - Covariance-matrix based
  - Logistic regression
- non-parametric
  - Multi-layer perceptron

(sampling paradigm)   (diagnostic paradigm)

We call these AF (adaptive-forgetting) classifiers.

Streaming Quadratic Discriminant Analysis

QDA can be motivated by reasoning about relationship of between and within group covariances, or assuming class conditional densities are Gaussian.

For static data, latter assumption yields discriminant function for jth class

$$g_j(x) = \log(P(C_j)) - \frac{1}{2}\log(|\Sigma_j|) - \frac{1}{2}(x - \mu_j)^T \Sigma_j^{-1}(x - \mu_i) \quad (3)$$

where μ_j and Σ_j are mean vector and covariance matrix, respectively, for class j.

Frequently, plug-in ML estimates for unknown parameters: μ_j, Σ_j, $P(C_j)$.

Idea here is to plug-in the AF estimates, $\hat{\mu}_t^{\lambda}$ etc.

Results in CA's thesis show that the AF framework above can be generalised, using likelihood arguments, to the whole exponential family. Thus, the priors, $P(C_t)$ can also be handled.

The approach is then:

- Forgetting factor for prior (binomial/multinomial)
- Forgetting factor for each class

The class of x_t is predicted when it arrives. Immediately thereafter, the class-label arrives, and the true class parameters are updated.

This will be problematic for large K or very imbalanced classes: few updates complicates the interpretation of the update equation for λ_t (Eq. 2).

Streaming LDA

The discriminant function in Eq.3 reduces to a linear classifier under various constraints on the covariance matrices (or mean vectors).

We consider the case of a common covariance matrix: $\Sigma_1 = \Sigma_2 = \ldots = \Sigma_K = \Sigma$. Again, we will substitute streaming estimates μ_j^λ, Σ^λ.

Have a couple of implementations options. One approach is

- Forgetting factor for prior
- Forgetting factor for each class
- Compute pooled covariance matrix, using streaming prior

Streaming logistic regression

Diagnostic methods do not have to handle prior probabilities separately, easing interpretation issues with updating λ.

In two class problems, logistic regression models the log-odds ratio as a linear function

$$\log\left(\frac{p(x|C_1)}{p(x|C_2)}\right) = \beta^T x$$

yielding simple forms for the posterior probabilities.

With static data, under mixture sampling, the likelihood function is simple, but requires non-linear optimisation.

The idea will be as before, to incorporate a forgetting factor in the opimisation criterion, and tune it according to the gradient.

Thus, we incorporate the (exponential) forgetting factor into the log-likelihood

$$l(\beta_t|y_{1\dots t}) = \sum_{i=1}^{t} \lambda^{t-i} l(\beta_t|y_t) = l(\beta_t|y_t) + \lambda l(\beta_t|y_{1\dots(t-1)})$$

As before, $\lambda \in (0, 1]$.

We update the parameter vector sequentially with

$$\beta_{t+1} = \beta_t - \alpha \nabla_\beta l(\beta_t|y_{1\dots t})$$

where$\nabla_\beta l(\beta_t|y_{1\dots t})$ is the gradient of the likelihood w.r.t. β, and α is another learning rate, as before.

An online version of backpropagation with momentum.

Calculations in Pavlidis et al (2010b) show that this computation, and the corresponding sequential optimisation of the forgetting factor, analogous to Eq. 2, can be made efficiently online via incremental update.

Streaming Multilayer perceptrons

We have extended the AF approach further, noting that the multilayer perceptron (MLP) can be regarded as a generalisation of logistic regression. This allows the opportunity for much more flexible models.

This introduces a large number of tunable parameters, and a model parameter - the number of hidden nodes.

NPs work shows that the same modification to the optimisation criterion can yield an efficient updating mechanism. Can avoid explicit computation and storage of Hessian.

Extensive experimental analysis suggests that recursive computation of the derivative of the optimisation criterion w.r.t. λ becomes unstable. We thus prefer to approximate this gradient using a finite difference approach.

The forgetting factor, introduced through the optimisation criterion as with logistic regression, can be adapted as before.

A regularisation term, with parameter γ, can also be incorporated, yielding an optimisation criterion like

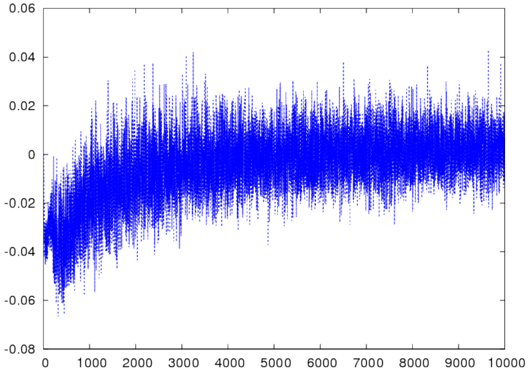
$$\sum_{i=1}^{t} \lambda^{t-i} l(w|y_i) + \gamma ||w||^q$$

This raises the question of the relationship between forgetting and model complexity, which can (in principle) be tuned. More on this later.

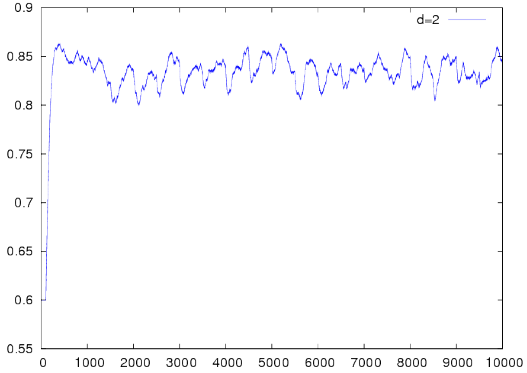
We restrict attention to an MLP with a single hidden layer. Initial experiments make us favour $q = 1$ (analogous to the LASSO), over $q = 2$ (analogous to ridge regression/weight decay)

Illustrations

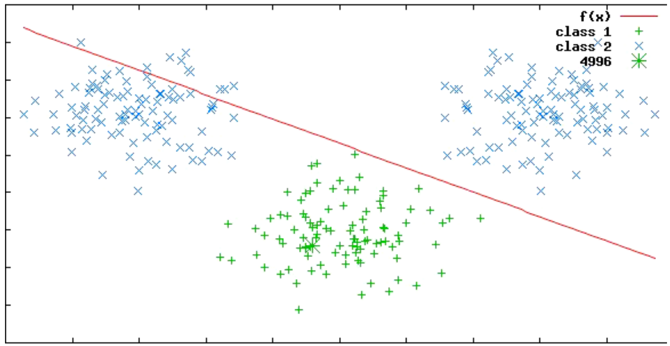
Logistic regression, gradient of forgetting factor, static environment



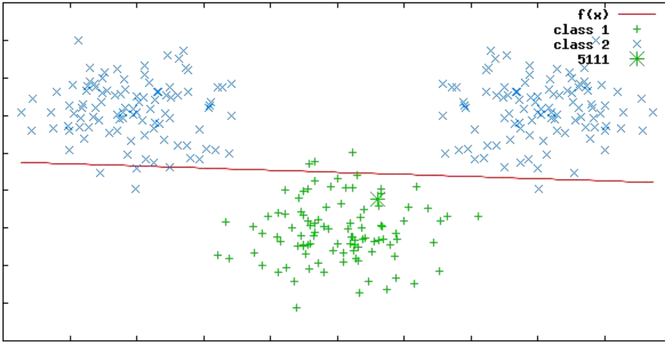
Logistic regression, forgetting factor, jump + drift environment. Evidence of monitoring capacity of AF?



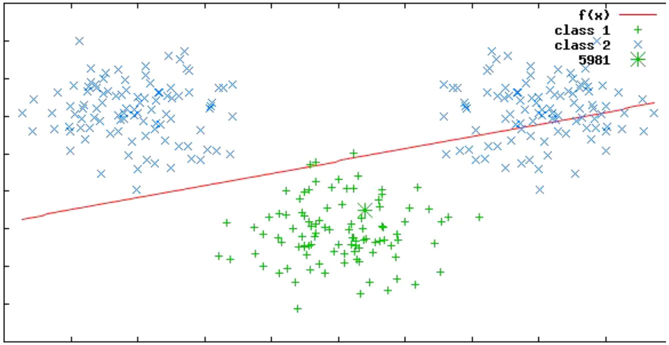
Logistic regression: before change point



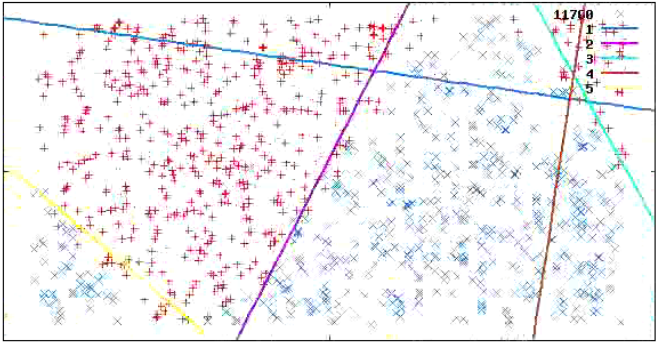
Logistic regression: at change point



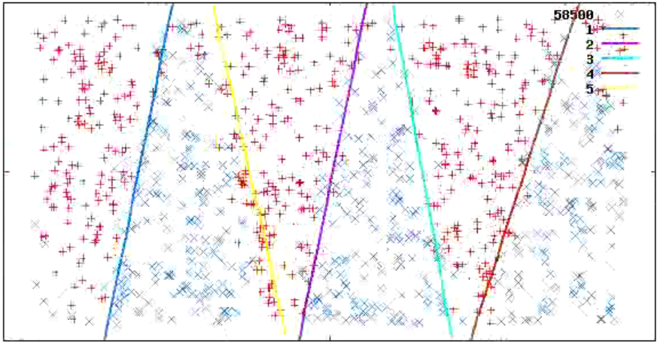
Logistic regression: after change point



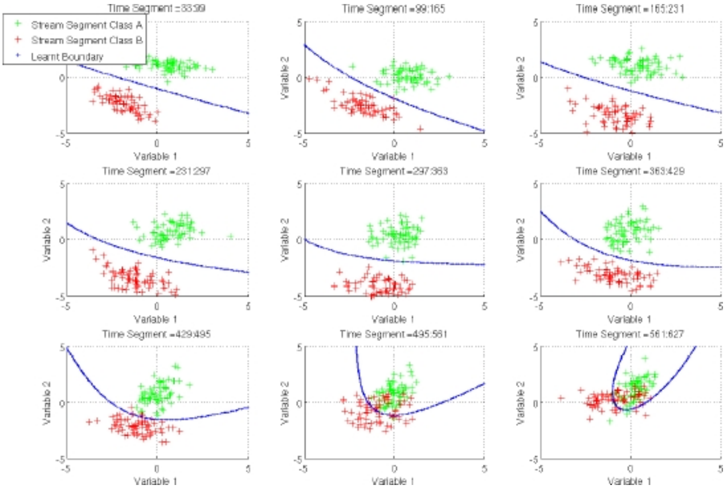
MLP- decision boundary implied by hidden nodes, drifting scenario



MLP- decision boundary implied by hidden nodes, drifting scenario



AF-QDA - 2D drifting problem



Performance assessment

Assessing the performance of streaming classifiers is complicated, due to the changing nature of data streams. For simulated assessment (which is crucial), we proceed as follows

- The data stream can have jumps, drift, or both.
- Data generation from an appropriate (usually simple) stochastic mechanism (Gaussian, mixture)
- Generate extra points, at each tick, to facilitate comparison with ground truth

Then compute performance measures, such as error rate, at each tick. For comparison between classifiers, can summarise average across time also.

Real data more difficult - usually rely on time-averaged pointwise error rate

Simulation Conclusions

We have conducted extensive simulation experiments (and PD and real data!), exploring and comparing these AF methods with comparable approaches in the literature.

- GENERAL: AF methods
  - handle drift/jumps/both automatically
  - are competitive with comparable methods (PA,PA-II, OLDA, PERC), but more general
  - generally recover better from change
  - exhibit performance degradation with speed of drift, and dimension (as do other methods).
- QDA:
  - Can suffer from problem of many parameters
  - but more responsive to change

Some general comments

So, we have a collection of different classifiers for handling the simple streaming classification problem.

The methods are efficient (typically $O(p^2)$) per tick. While this is slow for information retrieval, it is appropriate for learning.

Testing algorithms is difficult. We much prefer the simulation approach above, than the use of PD data sets such as STAGGER, which are not particularly challenging or interesting.

Issues

At least three interesting things to explore

- optimisation parameters (learning rates, bracket limits)
- Complexity and forgetting in streams
- timing issues

Optimisation parameters

Let us revisit the updating mechanism for λ

$$\lambda_t = \lambda_{t-1} - \alpha \frac{\partial \xi_t^2}{\partial \lambda}$$

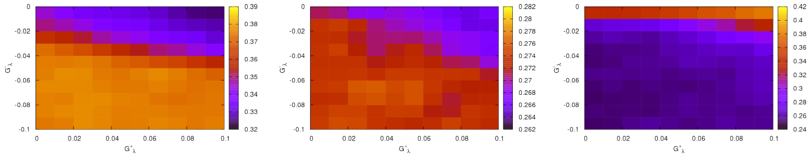
This is normally implemented with a bracket, to constrain the movement of λ

$$\lambda_t = \lambda_{t-1} - \alpha \frac{\partial \xi_t^2}{\partial \lambda}\bigg|_{\lambda_{min}}^{\lambda_{max}}$$

Brackets issues is open, but critical to good performance.

Learning rates and bracket values interact

Error rates, logistic, gradual drift - (learning rate=0.01,0.1,0.5)



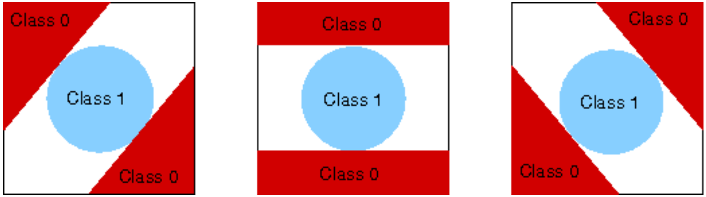
Seems clear brackets should not be fixed.

We have had some success with using the RPROP algorithm to control the movement of λ.

Complexity and Forgetting

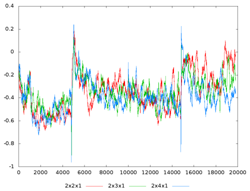
It is interesting to speculate about the relationship between forgetting (discarding old data) and complexity control (managing flexibility).

It is easy to conceive problems where a simple classifier has to change, where a more complex classifier can simply learn everything. Typically, these would be based on data in a bounded region.
Example: *continuous XOR*



With the MLP, we have experimented with tuning γ (complexity) the same way as λ (forgetting). Some interesting initial results.

Drift and jumps, continuous XOR variant: 4 network architectures, complexity? 4 networks? correlation between tuned parameters



Methods exhibit good performance. Correlation appreciable only at jumps. Forget everything, suppress complexity?

label timing

In real problems, the label is delayed in complicated ways.
Examples

- credit scoring: under standard definitions, good risk customers cannot be so flagged until near the end of a loan term. In contrast, bad risk customers are so labelled anytime during the loan term.
- fraud detection: fraudulent transactions (that pass filters) identified months after event. In contrast, *legitimate* transactions are never explicitly flagged – status presumed since not flagged as fraud

One approach is to use semi-supervised approaches. For example, incorporate the unlabelled vector to the predicted class. Reject option very important here, to prevent classes *merging*.

Another (of many) timing issue is that more than one vector can arrive at a time (eg. fraud)

The QDA/LDA streaming models admit efficient batch block updating, via rank k update. Thus, multiple vectors can be incorporated at one time.

Experiments with credit application data suggested that this approach works no better (in terms of error rate) than a random ordering (within tick) and sequential updating.

Of course, this depends very much on the underlying dynamics of the data.

Conclusion

- AF-classifier works well for streaming problems (and indeed for large data sets).
- Varying forgetting factor does respond to change, and gives something to monitor, though sequential monitoring difficult, because gradient-like quantities very volatile.
- Interesting relationship between forgetting and complexity.
- Many of the assumptions are restrictive compared to real problems.

Future Work

- THEORETICAL ANALYSIS - difficult, perhaps adapt ideas from stochastic approximation
- BAYESIAN FORMULATION - based on the power-prior, can alleviate (or at least shift) some difficulties with the bracket on λ
- HYBRID MODELS - exploit links between optimal filters
- TIMING ISSUES - motivated by real problems. Semi-supervised approach, change λ as function of time

References

- Adams, N.M., Tasoulis, D.K., Anagnostopoulos, C. and Hand, D.J. 'Temporally-adaptive linear classification for handling population drift in credit scoring', In Lechevallier, Y. And Saporta. (eds), COMPSTAT2010, Proceedings of the 19th International Conference on Computational Statistics, 2010, Springer, 167-176.
- Anagnostopoulos, C. 'A statistical framework for streaming data analysis', PhD Thesis, Department of Mathematics, Imperial College London, 2010.
- Anagnostopoulos, C., Tasoulis, D.K., Adams, N.M. and Hand, D.J., 'Streaming Gaussian classification using recursive maximum likelihood with adaptive forgetting', Machine Learning, (2010), under review.
- Anagnostopoulos, C., Tasoulis, D.K., Adams, N.M. and Hand, D.J. 'Temporally adaptive estimation of logistic classifiers on data streams'. Adv. Data An. Classif.,3(3) (2009),243-261.
- Haykin, S. 'Adaptive filter theory', 4th edition, Prentice Hall (1996).
- Kelly, M.G., Hand, D.J. and Adams, N.M., 'The impact of changing populations on classifier performance' in 'KDD 99, Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', Chaudhuri, S. and Madigan, D. ed(AAAI), 1999, 367371.
- Pavlidis, N.G., Adams, N.M and Hand, D.J., 'Adaptive Online Logistic Regression', Pattern Recogn., (2010) under review.
- Pavlidis, N.G., Tasoulis, D.K., Adams, N.M. and Hand, D.J. 'Adaptive consumer credit classification', Europ. J. Finance, (2010), under review.
- Weston, D.J., Anagnostopoulos, C., Tasoulis, D.K., Adams, N.M. and Hand, D.J. 'Handling missing feature values for a streaming quadratic discriminant classifier', Data Mining and Knowl. Disc., (2010), under review.