Statistical Aspects of Classification in Drifting Populations

C. C. Taylor
Department of Statistics,
University of Leeds,
Leeds LS2 9JT, UK
charles@amsta.leeds.ac.uk

G. Nakhaeizadeh
Daimler-Benz Forschung und Technik,
Postfach 2360,
89013 Ulm DE
nakhaeizadeh@dbag.ulm.DaimlerBenz.com

G. Kunisch,
Department of Stochastics,
University of Ulm,
Ulm DE

Abstract

This paper discusses ideas for adaptive learning which can capture dynamic aspects of real-world datasets. Broadly, we explore two approaches. The first examines ways of updating the classification rule as suggested by some monitoring process (similar to those used in a quality control problem), and this is applied to linear, logistic and quadratic discriminant. The second approach examines nonparametric classifiers based explicitly on the data and ways in which the data can be dynamically adapted to improve the performance. These methods are tried out on simulated data and real data from the credit industry.

Key words: Adaptive Discrimination, Incremental Learning, Kernel Classifier, Nearest Neighbour, Pattern Recognition

1 Introduction

In classical supervised learning, the available examples (the training data) are usually used to learn a classifier. In many practical situations in which the environment changes, this procedure ceases to work. In the project StatLog (Michie, Spiegelhalter and Taylor, 1994) we encountered this situation in the case of a dataset on automatic transmissions and a credit-scoring application. Generally speaking, the application of a discrimination algorithm to classify new, unseen examples will be problematic if one of the following events occurs after the "learning phase":

(i) The number of attributes changes. We have experienced this case in a credit-scoring application that certain customer information can no longer be stored due to legal requirements. On the other hand, it has also been the case that one is able to get more information than before about the customers, *i.e.* one can consider more attributes;

(ii) The number of attributes remains the same but the interpretation of the records of the datasets changes over the time. In this situation the distribution of at least one class is gradually changing.

To solve this problem one could simply **relearn** the rule provided that there are enough new examples with known class, but this could be very wasteful. An alternative is **Incremental learning**. For example, Utgoff (1989) has developed an algorithm that produces the same decision tree as would be found if all the examples had been available at once, and Utgoff (1994) has recently devised an improved algorithm (ITI) which can handle continuous variables, multiple classes, noise *etc.*. However, this cannot deal with the problem in item (ii) above since some of the old data should be downweighted or discarded as no longer being representative of the current population.

A closely related topic is that of *drifting concepts*. For example, early work was done by Schlimmer and Granger (1986) with his STAGGER system. See also Schlimmer (1987) and more recent work by Kubat and Widmer (1995). De Raedt and Bruynooghe (1992) argue that incremental concept-learning — when formulated in a logical framework

— can be understood as an instance of the more general problem of belief updating. This insight allows for potential interesting cross-fertilization between these areas.

This paper discusses ideas for adaptive learning which can capture dynamic aspects of real-world datasets. Although some of these ideas have a general character and could be applied to any supervised algorithm, here we focus attention on nonparametric methods as well as linear, logistic and quadratic discriminant. The dynamic nearest neighbour classifier modifies the "training data" by keeping a record of usefulness as well as the age of the observation. The kernel density estimation uses a weighted average of kernel functions with the weight being determined by the age, whereas the other classical methods use a quality control type-system to update rules appropriately.

Suppose that batches of observations are used to monitor the performance of the classifier or any change in the class populations. There are several quantities that could be monitored using a Shewhart-type quality control plot:

1. the average probability of (maximum) class membership;
2. the means and covariances of the most recent batch of observations for each class;
3. the error rates;

These can be plotted in time-series fashion, with appropriate warning and action limits indicating that the process has changed. In this scenario we envisage that the learning will be done at discrete time points, preferably (but not necessarily) using the previous rule to find an updated rule. This approach enables us to model large sudden changes in the populations. A significant change detected by the measures in 1. and 2. above is a warning sign which suggests that the classifier should be adapted. If, in addition, the measure used in 3. is also significantly changing, then the necessity to adapt the algorithm is more pronounced. If it is the case that 1. and 2. suggest that an adaptation is required, but measure 3. is apparently not changing, then a more appropriate learning algorithm may be required.

2 Changing the Classifier

In this section we discuss ways of updating the classifier either by explicitly changing the rule which has been learned, or by changing the learning data by substituting or adding more recent observations. Suppose that we examine the data in batches of size m and that, at time $t + mk$ we detect a change which requires adaptation in the learned rule. Any algorithm can be totally relearned from recent observations after a change in one of the classes has been detected. More interestingly, we can consider how best to re-use previously learned information.

2.1 Updating the "rules"

One method is to allow the weights to depend on the performance of the classifier. Suppose that we have a current rule which is used on the next batch of observations and results in an error rate of $e_{current}$. Suppose also that if the new data is used to obtain a rule, then this would give an error rate for this batch of e_{new}, although this rate could be badly biased, since the training and test data are the same. If these error rates are very different this suggests that the current rule should be updated. One approach is to obtain a revised rule which is a weighted average of these two rules with weights proportional to $1 - e_{current}$ and $1 - e_{new}$. Another possibility is to build an error rule from previous error rates and compare the current error $e_{current}$ to that or some function thereof.

2.2 Updating the "data"

We can use a "similarity" rule to throw away observations in the current training set which are different – *i.e.* they are close in feature space, but have different class label – from those recently observed. Alternatively, the older observations can be eliminated or a kind of moving window with a predefined number of observations, possibly representative and new ones, could be used. We try a possible implementation whereby old data are discarded and new data are included in the "template" used for establishing a rule according to their perceived usefulness. As presented, this system will have some limitations. For example, if there are drifting populations then new data will be incorrectly classified, but should nevertheless be included in the template set. This point is taken up in a nearest neighbour implementation in section 3.2. Note that this approach can be used for any (not just similarity-based) algorithms.

3 Algorithm Details

For logistic, quadratic and linear discrimination we have implemented a type of quality control system in which the error rate is monitored. There are various scenarios: (i) in the case of small increases in error, the rule is updated; (ii) in the case of large increases, the rule is re-learned only using the most recent batch of data; (iii) in the case of small decreases in error, no change is made to the existing classifier. For the linear and logistic rules, the classifier is completely relearned when the error rate for the current batch exceeds an action limit of α standard deviations (SDs) above the mean, and is otherwise modified unless the error rate is less than δ SDs below the mean. The quadratic rule has a similarly defined action limit (ϵ) and warning limit (δ) SDs as well as a moving window (ws) and an editing factor, γ. Full details of the approach can be found in Nakhaeizadeh *et al.* (1997).

3.1 Kernel Density Estimation and Classification

Suppose that we have data $x_1, x_2, \ldots, x_n$ from some unknown distribution $f(x)$. The usual kernel estimator is given by

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where $K(.)$ is the kernel function, and h is the bandwidth which controls the amount of smoothing, with appropriate modifications for multivariate observations. Numerous researchers have tackled the problem of choosing an appropriate bandwidth which is based on the data – see, for example, Wand and Jones (1995) for references. However, most of the results are related to minimizing integrated mean squared error (IMSE) which is given by $E\int(\hat{f} - f)^2$. It is worth noting that such a policy for choosing h may not work very well in a classification setting when we want to minimize the expected misclassification rate which for two classes is given by

$$\pi_1 \int_{\hat{f}_{h_2}(x) > \hat{f}_{h_1}(x)} f_1(x)dx + \pi_2 \int_{\hat{f}_{h_1}(x) > \hat{f}_{h_2}(x)} f_2(x)dx = \pi_1 I_1 + \pi_2 I_2, \qquad \text{say,} \tag{1}$$

where π_i is the prior probability that the data belongs to class C_i. The usual approach of taking a Taylor series expansion to obtain an asymptotic quantity does not work here, and the fact that the limits of the integral are random variables makes this look intractable. A simulation illustrates that the optimal h_1, h_2 to minimize IMSE can be quite different to those which minimize expected error rate. In this example, 100 observations were simulated from each of $N(0,1), N(2, 0.5^2)$. For each of 100 samples we calculated $\hat{f}_1$ and $\hat{f}_2$ using a range of different smoothing parameters. For these distributions, $(h_1, h_2) = (0.422, 0.211)$ minimize the asymptotic IMSE, whereas the error rate is minimized for $(h_1, h_2) = (0.720, 0.215)$. We could consider estimates of $I_i, i = 1, 2$ in equation (1) given by (for $i = 1$)

$$\hat{I}_1(h) = \frac{1}{hn_1} \sum_{x_j \in C_1} \int_{\hat{f}_{h_2}(x) > \hat{f}_{h_1}(x)} K\left(\frac{x - x_j}{h}\right) dx. \tag{2}$$

In this case $h = 0$ gives the usual leave-one-out, or cross-validation estimate of the error rate, since the integral in (2) will be 1 if $f_{h_2}(x_j) > \hat{f}_{h_2}(x_j)$ and 0 otherwise. So although $h = 0$ will give an unbiased estimate of the error, a value of $h > 0$ can give a better estimate (in terms of mean squared error). Again, analytical results are difficult and simulations are required in order to determine good choices of h and thence good choices of h_1 and h_2.

We now turn to the *dynamic* version of the kernel estimator, which in the simple case is given by

$$\hat{f}_T(x) = \frac{1}{hN_T} \sum_{t=1}^{T} w_t K\left(\frac{x - x_t}{h}\right) \tag{3}$$

as an estimate of the density $f_T(x)$ at time T when we have previously observed x_t in the class of interest. Here w_t are weights and N_T is a normalizing constant. For example, we could choose $w_t = e^{-\lambda(T-t)}$ in which case $N_T = (1 - e^{-\lambda T})/(1 - e^{-\lambda})$ or

$$w_t = \begin{cases} 1 & \text{for } T \geq t > T - W \\ 0 & \text{otherwise} \end{cases}$$

in which case $N_T = W$. Using either of these parameterizations for w_t requires choice of either λ or W, in addition to the smoothing parameter h. Of course both parameters must be chosen for each class. Again, analytic calculations appear to be intractable even for IMSE and very simple dynamic models. However, numerical calculations are simplified

by noting that simple updating formulae can be derived expressing $\hat{f}_T(x)$ in terms of $\hat{f}_{T-1}(x)$ and a kernel function of X_T.

In this paper we consider experiments on real and simulated dynamic data in which we train on an initial set of data (ordered by time), choosing any parameters by cross-validation and then testing on observations in the second part of the data. The kernel function is is the Normal density which gives a classifier equivalent to a nearest neighbour rule for h sufficiently small, although there are numerical difficulties for very small h.

3.2 Nearest Neighbour

A dynamic 1-nearest neighbour algorithm has been developed in which examples are given a nominal weight of 1 when they are first observed. Then all observations "age" at a rate ν, to allow for the fact that in a dynamic system older examples are likely to be less useful. In addition, future examples which are classified affect the existing weights so that: (i) If the new observation is correctly classified by the nearest neighbour, but would be incorrectly classified by the second nearest neighbour, then the nearest neighbour gets its weight increased by γ (a sort of condensing factor); (ii) If the new observation is incorrectly classified by the nearest neighbour, but would have been correctly classified by the second nearest neighbour, then the nearest neighbour gets its weight decreased by ϵ (a sort of editing factor). This approach essentially keeps a "record of usefulness" for each of the observations in the current training set.

The weight, $w(x)$, of each example then behaves like a Markov Chain with an absorbing barrier at 0, whose properties can be studied in order to get some idea of suitable choices for the three parameters: ν, γ and ϵ. For example, the drift of the weight of an example, say at x, is

$$\text{drift}(w(x)) = -\nu + \gamma p(x) - \epsilon q(x)$$

where $p(x)$ is the probability that this example will be the nearest observation to the next example and that the situation described in (i) above occurs, and $q(x)$ is the probability that this example will be the nearest observation to the next example and that the situation described in (ii) above occurs.

This can give the expected time for an example to become "extinct" ($w(x) \leq 0$) (*i.e.* no longer used for future classification), and the expected number of examples in the classification set at any given time (assuming some sort of equilibrium).

Note that this use of weights is distinct from that used by Cost and Salzberg (1993) in the modified value difference metric (MVDM) in which the weights were used to adjust the **distance** between observations by associating an importance for each example. In our case the weights are merely used to reflect the age of the observation, as well as a measure of usefulness, in order to determine whether the example should be retained at this time; they are not used to modify the distance metric. Related ideas on feature weighting can also be found in Aha (1989) and Wettschereck and Aha (1995).

4 Results

In this section we describe some results of our adaptive updating ideas and compare them to conventional statistical classification methods in an example. The simplest and nonadaptive approach is to use the classification rule that was learned from the training data and apply that to all batches, with no updating. A small modification is to update the priors according to the new data, and a further modification is to use the priors which are estimated using only the last batch. An alternative benchmark method is to completely re-learn the rule at each time point.

4.1 Simulated data

We tried out some of the above ideas on two simulated datasets (dat1 and dat2). At each of 1000 time points we generate an example with 3 variables (X_1, X_2, X_3) from 2 classes. We use the first 500 observations as the *training data* and the remaining 1500 as *test data*. The distributions of each class has 2 independent normal variables (with unit standard deviation) and a uniformly distributed (on $[0, 1)$) "noise" variable. The mean of the noise variable $\mu_3 = 0.5$ was independent of time; the means of the normal variables vary with time as follows:

dat1: Class 1 has $\mu_1, \mu_2 = 0$ for $t \leq 750$ and $\mu_1, \mu_2 = t/1000$ for $751 \leq t \leq 1000$, whereas Class 2 differs in that $\mu_2 = 2$ for $t \leq 750$ and $\mu_2 = 2 + t/1000$ for $751 \leq t \leq 1000$, so that there is no change to the distributions until two thirds of the testing phase, when there is a sudden jump followed by a slow drift.

dat2: Class 1 has $\mu_1, \mu_2 = 0$ and Class 2 has $\mu_1 = 2t/1000, \mu_2 = 2 - 2t/1000$ for $1 \leq t \leq 1000$. In this case there is a gradual shift in the training and test phase of the second group in the mean of (X_1, X_2) from $(0, 2)$ to $(2, 0)$.

Since we split the testing data into batches of 50 observations (which always corresponds to 25 observations from each class) the change should happen in batch 21 when applying dat1 and should go through the whole training and testing phase when considering dat2. Note that for the both datasets, the observations were ordered so that the priors (however estimated) were always equal.

Table 1 shows the results of our calculations using the updating-rule approach. It can be seen that our updating routine achieves better performances than both conventional approaches. In order to see where the improvement occurred, we monitored the error rates of single batches. As expected due to the large training set, there was no great

data set	dat1		dat2	
parameters/method	linear	logistic	linear	logistic
$\delta = 0.5, \alpha = 2$	0.166	0.163	0.228	0.231
$\delta = 1, \alpha = 3$	0.160	0.168	0.229	0.223
no-learn	0.186	0.175	0.360	0.366
re-learn	0.166	0.163	0.271	0.273

Table 1: Average error rates of linear and logistic discrimination rule applying the updating approach

data set	dat1	dat2
parameters/method	mean error	
$\gamma = 0.25$,ws= 300, $\delta = 3, \epsilon = 2$	0.164	0.232
$\gamma = 0.25$,ws= 600, $\delta = 3, \epsilon = 2$	0.168	0.241
$\gamma = 0.5$,ws= 300, $\delta = 3, \epsilon = 2$	0.165	0.236
$\gamma = 0.5$,ws= 500, $\delta = 2.5, \epsilon = 1.5$	0.164	0.229
$\gamma = 1$,ws= 300, $\delta = 3, \epsilon = 2$	0.175	0.239
no-learn	0.190	0.353
re-learn	0.175	0.267

Table 2: Average error rates of quadratic rule applying the data learning approach (4.2) (ws =window size)

difference between our classification procedures until the drift occurred (*i.e.* the no-learn routine achieved best results in that area) in the data dat1. But from batch 21 onwards our classifier was much better than conventional methods, with the no-learning rule worst. The error rates of the linear updating rule with parameters $\alpha = 3, \delta = 1$ after this batch were 5.2% and 8.4% smaller than the re-learn and no-learn rules. The second data set caused greater problems for our discriminant function. But here also our algorithm showed higher success rates than the other approaches.

If we consider the quadratic (data-learn) approach, we get similar results (see Table 2). For the first 20 batches of dat1 all routines seem to have almost the same performance, although the relearning method is slightly more accurate than the other ones. But again after the shift and during the drift (dat1) the no-learn method loses performance, whereas the relearning rule works almost as well as our algorithms. But as soon as we apply the overall drift data set (dat2) to the different approaches the enhancement by our algorithm is obvious. The monitoring process shows almost constant error rates, while the error rates of the re-learn approach gradually increases (the no-learn rule completely ceases to work: three batches produced error rates of 50 percent). We used a set of parameters from which the reader can observe the effects of changing them.

The kernel classifier showed a similar pattern. We tried 4 approaches: (i) h_1, h_2 were trained on the test data, but the classifier was not updated (no-learn); (ii) the classifier was updated using all observations thus far observed (with no change in the smoothing parameters); the dynamic kernel estimator given by equation (3) with (iii) a weighted exponential decay (λ) and (iv) a window of width W learned from the training data. In the latter two cases, 2 parameters for each of the two classes were chosen by cross-validation. The results are given in Table 3.

Method	Error rate		estimated parameters	
	dat1	dat2	dat1	dat2
(i) no-learn $((h_1, h_2))$	0.201	0.357	(.75, .8)	(1.5, 1.5)
(ii) all-data $((h_1, h_2))$	0.167	0.274	(.75, .8)	(1.5, 1.5)
(iii) exponential wieghts $((h_1, h_2, \lambda_1, \lambda_2))$	0.173	0.239	(3, 3, .08, .08)	(1, 1, .06, .06)
(iv) moving window $((h_1, h_2, W_1, W_2))$	0.165	0.231	(1.5, 1.5, 210, 245)	(1.6, 1.6, 95, 250)

Table 3: Error rates for kernel classifier on simulated data. See text for further details of (i)–(iv)

4.2 Real data

The data set concerns applications for credit; the class indicates the success of the application (*i.e.* two classes). There are 156 273 observations which cover a 2-year period. Originally there were many qualitative attributes which were then coded into indicator variables (since all the programs used here require real-valued data). Subsequently, 15 of these 0/1 variables were used for classification purposes. The initial 10-fold cross-validation was computed using the first 5000 observations. Subsequently, we used batch sizes of 1000. The error rates are shown in Table 4 for the linear and logistic approach and in Table 5 in the case of the quadratic method. Figure 1 suggests that there is a shift in

param. / approach	linear	logistic
$\delta = 0.25, \alpha = 2$	7.97	–
$\delta = 0.5, \alpha = 2$	7.96	7.45
$\delta = 0.5, \alpha = 1$	8.54	–
$\delta = 1, \alpha = 3$	8.49	7.46
$\delta = 1.5, \alpha = 2$	8.56	7.36
$\delta = 2, \alpha = 4$	–	7.33
$\delta = -1, \alpha = 1$	7.73	7.45
no-learn	8.97	8.26
re-learn	8.97	8.97

Table 4: Average error rates (%) of linear and logistic discrimination rule applying the updating approach (section 4.1) and using real data

parameters of quadratic / approach	$\bar{e}$
$\gamma = 0$,ws= 3000, $\delta = 100, \epsilon = 100$	11.94
$\gamma = 1$,ws= 3000, $\delta = 7, \epsilon = 8$	8.789
$\gamma = 0.25$,ws= 5000, $\delta = 1, \epsilon = 3$	8.865
$\gamma = 0.7$,ws= 7000, $\delta = 1, \epsilon = 2$	8.861
$\gamma = 0.5$,ws= 5000, $\delta = 1.5, \epsilon = 3$	8.830
$\gamma = 0.3$,ws= 8000, $\delta = 1, \epsilon = 4$	8.863
no-learn	8.974
re-learn	8.974

Table 5: Average error rates (%) of quadratic discrimination rule applying the data-learn-approach (section 4.2) and using real data

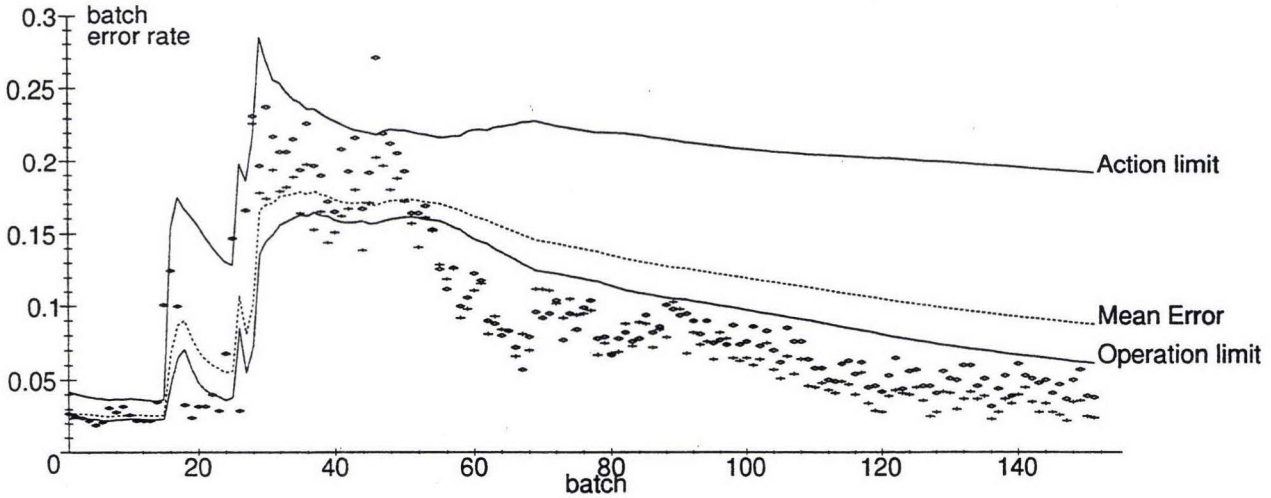

Figure 1: Typical output of monitoring process for linear and logistic discrimination applying the "errorlearn approach". This example uses the real data and compares linear adaptive discrimination (+) with parmeters $\delta = 0.5, \alpha = 2.0$ and re-learn method ($\diamond$).

the data in batches $30 - 50$ which was where our algorithms showed the greatest difference in the success rates. For example the linear update process only differed from the conventional methods in this region. Unexpectedly the re-learn and no-learn approaches had the same success rates when using the linear discriminant rule. As expected the logistic approach showed the best performance, but even here we could find parameters that improved the overall performance. Also astonishing were the results when applying the parameters ($\alpha = 1, \delta = 1$), which causes only relearning at certain steps but no updating at all.

As in previous studies the quadratic discriminant rule did not work as well as the other classifiers in the case that the data was binary. This was even more true of the kernel classifier, although the poor performance may have been due in part to the fact that we neither scaled the binary data, nor considered a different smoothing parameter in each dimension. So in effect we worked with the assumption that the variables were independent with common variance

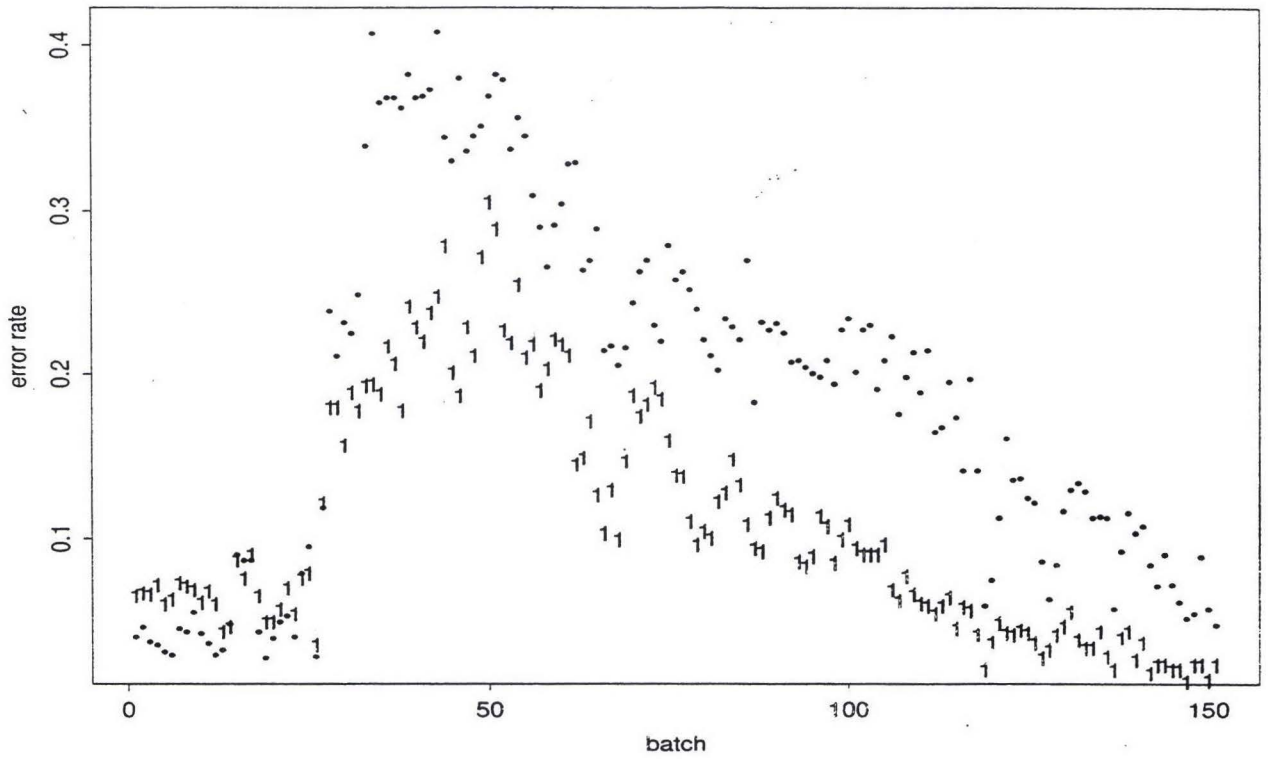

Figure 2: Error rates for two kernel classifiers. The "•" points are for a non-dynamic classifier in which the priors were updated according to the proportions observed in the previous batch. The '1' points were for a moving window classifier, equation (3) with $(W_1, W_2) = (400, 3000)$ and $(h_1, h_2) = (.25, .25)$

which was certainly not the case. The kernel classifier which was kept fixed gave an overall error rate of over 20% (note that the default rule only gives 10.5%), whereas updating the prior to reflect the proportions in the last batch gave a small improvement to 18.2%. The dynamic version (moving window) gave a large improvement to 10.4%; still only just better than the default rule, but with more careful choice of smoothing, indicates what may be possible.

The "static" nearest neighbour classifier gave an error rate of 9.1% whereas the dynamic version is faster (since the number of sites stored was about 2000 less) and gave an error rate of 7.4%. Details of the training and parameter selection can be found in Nakhaeizadeh *et al.* (1996).

5 Conclusions

Stationarity is a key issue which will affect the performance of any dynamic classification algorithm. For example, if the changes which occur in the training phase are very different from the nature of the changes which take place during testing then the way the parameters are updated is likely to be deficient. It could be agued that it is best to update whenever a new observation becomes available (as is done with the kernel classifier and nearest neighbour algorithm above) but we would argue that methods should include a monitoring process even if this monitoring is not normally used to update the rules. The reason for this is that, even if a continuous updating method is adopted, the *optimal amount* of updating (or the way in which updating is performed) can still vary over the time span of the data. Furthermore, most forms of monitoring will make use of batch – whether natural or artificial – information in order to be more confident about the size of any change that has occurred. However, we leave open the question of using a moving window for monitoring purposes.

References

Aha, D. W. (1989). Incremental, instance-based learning of independent and graded concept descriptions. In *Proceed-*

ings of the Sixth International Workshop on Machine Learning, Ithaca, NY, pp. 387–391. Morgan Kaufmann.

Cost, S. and Salzberg, S. (1993). A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning* **10**, 57–78.

De Raedt, L. and Bruynooghe, M. (1992). Belief updating from integrity constraints and queries. *Artificial Intelligence* **53**, 291–307.

Kubat, M. and Widmer, G. (1995). Adapting to drift in continous domains. In *Lecture Notes in Artificial Intelligence*, Volume 912, pp. 307–310.

Michie, D. M., Spiegelhalter, D. J. and Taylor, C. C. (Eds.) (1994). *Machine Learning, Neural and Statistical Classification*. Chichester: Ellis Horwood.

Nakhaeizadeh, G., Taylor, C. C. and Kunisch, G. (1996). Dynamic aspects of statistical classification. In *Intelligent Adaptive Agents, AAAI Technical report No. WS-96-04*, Menlo Park, CA, pp. 55–64. AAAI Press.

Nakhaeizadeh, G., Taylor, C. C. and Kunisch, G. (1997). Dynamic supervised learning: Some basic issues and application aspects. to appear.

Schlimmer, J. C. (1987). Incremental adjustment of representations for learning. In *Fourth International Workshop on Machine Learning*, pp. 79–90. Irvine, CA:Morgan Kaufmann.

Schlimmer, J. C. and Granger, R. (1986). Incremental learning from noisy data. *Machine Learning* **1**, 317–354.

Utgoff, P. E. (1989). Incremental learning of decision trees. *Machine Learning* **4**, 161–186.

Utgoff, P. E. (1994). An improved algorithm for incremental induction of decision trees. In *Proceedings of Eleventh Machine Learning Conference*, Rutgers University. Morgan Kaufmann.

Wand, M. P. and Jones, M. C. (1995). *Kernel Smoothing*. London: Chapman and Hall.

Wettschereck, D. and Aha, D. W. (1995). Weighting features. In M. Veleso and A. Aarnodt (Eds.), *Proceedings of the 1st International Conference on Case-Based Reasoning Research and Development*, Volume 1010 of *LNAI*, Berlin, pp. 347–358. Springer Verlag.