Dynamic Aspects of Statistical Classification

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

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

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. Although some of these ideas have a general character and could be applied to any supervised algorithm, here we focus attention on a nearest neighbour classifier as well as linear, logistic and quadratic discriminant. The nearest neighbour classifier modifies the "training data" by keeping a record of usefulness as well as the age of the observation. The other classifiers use a quality control type-system to update rules appropriately. These methods are tried out on simulated data and real data from the credit industry.

Key words: Incremental Learning, Nearest Neighbour, Adpative Discrimination, Pattern Recognition

1 Introduction

In classical supervised learning, the available examples (the training data) are usually used to learn a classifier which can be applied later to determine the class of new, unseen examples. 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 credit-scoring application. For example, it might be that the customers with a monthly income of more than DM 5000 were previously regarded by the credit institute as "good risks". But in the recent years the institute has observed that a lot of its customers with a monthly income of more than DM 5000 have not repaid their loans and thus belong to the group of "bad risks". On the other hand, it might

be that due to strong competition which exists in the credit market, the credit institute has had to reduce the threshold for monthly income from DM 5000 to DM 4000 to be able to give more loans. It means that a customer who has a monthly income of DM 4000 and was up to now regarded as a "bad risk" is now eligible for credit.

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":

1. 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.

2. 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. As mentioned above, the changes of the payment behaviour of customers in the case of credit-scoring or the changes of the diagnosis device in the case of automatic transmissions are examples of this problem (see Michie *et al.* (1994), chapter 9).

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 2 above since some of the old data should be down-weighted or discarded as no longer being representative of the current population.

Filosofov (1992) gives some general principles for the development of classification systems for dynamic objects with variable properties that are observed under changing conditions. He outlines the following requirements for dynamic classification systems: to handle changing composition of features, different feature types, and varying accuracy and statistical characteristics of observations; to give precedence in time and quantitatively change proximity of the object to a particular class; to rely on a single prior description of objects or processes. He also outlines three stages in the classification process: prepare the input data for execution of the next classification cycle; execute the classification cycle; augment the available information about the objects with new data. The approach, which is fairly general, uses conditional probabilities. He concludes by recommending a hierarchical system for classification of dynamic objects, retaining a catalogue that stores the results of past observations, and generalized distributions of features on one of the classification levels.

Joseph and Hanratty (1993) explore the concept of incremental learning using Artificial Neural Networks (ANNs) to provide on-line adaptation to changing process conditions in a quality control application. Two key ingredients are: the **forgetting factor** (how fast the network model should discount old data) in which past measurements are often weighted with an exponential decay; an **updating algorithm** which can takes a number of forms (randomly replace a part of the training set with the new data, replace the oldest training data with new data or replace the oldest training data that has similar operating conditions (*i.e.* class) as the new data).

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 considers the situation described in item 2 above, namely that the distributions of the classes may be changing over time. Broadly, we explore two approaches. The first examines ways of updating the classification rule as suggested by some monitoring process. The second examines a (nearest neighbour) classifier based explicitly on the data and ways in which the data can be dynamically adapted to improve the performance.

2 Monitoring Process

2.1 Introduction

In order to make visible how the allocation rule works it is necessary to observe some values regularly. This is what we mean by the expression process monitoring. In this case batches of observations should be taken to monitor the performance of the classifier or any change in the class populations. There are several quantities that could be monitored:

1. the average probability of (maximum) class membership – if this is drifting downwards (towards $1/q$ where there are q classes) then adaptation is required;
2. the means and covariances of the most recent batch of observations for each class;
3. the error rates (assuming that the class is known);
4. the estimated prior group probabilities (if the observed data is acquired by mixture sampling).

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 warranted.

2.2 Warning and Action Limits

In the absence of any theoretical results on standard errors for estimated error rates, which are available for normal-based discrimination procedures; see McLachlan (1992) for example, a general ap-

proach using a Shewhart quality control chart can be implemented. Smith (1994) discusses the use of quality control chart in a neural network setting.

Tests of the mean and variance of each class can use Hotelling's T^2 and Box's M- test respectively; see Mardia, Kent and Bibby (1979). In this case, since the monitoring is for a multivariate quantity we may plot the actual test statistics in the first instance. If a deviant batch is detected then we could carry out further diagnostics to investigate which of the attributes has changed. Although these tests make Normal distribution assumptions, they can also be used for the nonparametric classification methods just for monitoring purposes. An alternative nonparametric version would be to use a multivariate version of the Kolmogorov-Smirnov test which detects whether there has been any change in the distribution.

3 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.

3.1 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.

Figure 1 illustrates the process. An initial learning phase learns the required parameters as well as estimating an error rate by cross-validation. The test data is then classified in batches. According to the size of the error rate there may be total re-learning or merely adjustment of the current rules if the change is deemed to be small. In the case that the error rate is less than the norm, no updating is carried out.

3.2 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. Figure 2

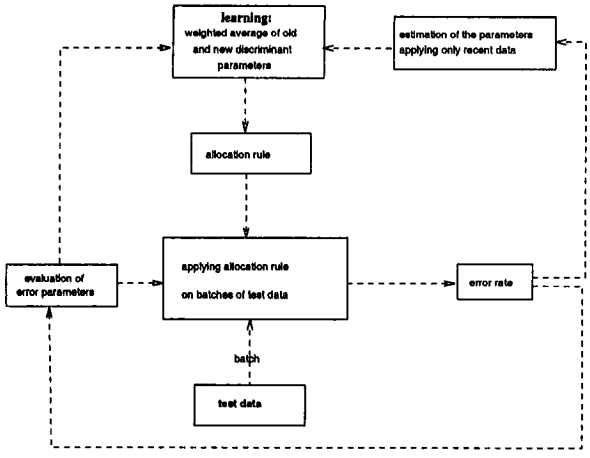

Figure 1: Learning by error reaction

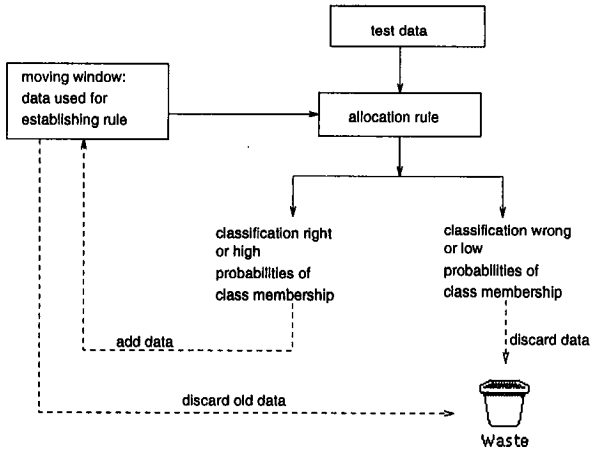

Figure 2: Learning by data manipulation

illustrates 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 4.2. Note that this approach can be used for any (not just similarity-based) algorithms.

4 Algorithm Details

4.1 Updating rules

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:

- in the case of small increases in error, the rule is updated;
- in the case of large increases, the rule is re-learned only using the most recent batch of data;
- in the case of small decreases in error, no change is made to the existing classifier.

Full details of the above methods are given in (Nakhaeizadeh *et al.* , 1996) who in addition use a moving window to retain a good template set to estimate the quadratic discriminant rule. A similar approach was also used by Gibb, Auslander and Griffin (1994) who partitioned their data into three groups: a *template set*, a *training set* and a *test set*. The template set was used for feature selection. After feature selection and parameter estimation the algorithm classifies observations in the test set. If a gradual drift is detected the most recently classified object is added to the template set, the feature selection process is repeated and the parameters re-estimated from the training set. In their application, the true class of the observations in the test set is never revealed to the algorithm, so there is a risk that misclassified objects can be added to the template, thus contaminating it.

4.2 Updating data

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 λ, 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" [1] 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: λ, γ and ϵ. For example, the drift of the weight of an example, say at x, is

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

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.

The above scenario can be adapted to deal with more sudden changes in the population means. In this case we have an adaptive λ which depends on time and the change in error rate such that

$$\log \lambda_{\text{new}} = a^{e_{\text{new}} - e_{\text{old}}} \log \lambda_{\text{old}} \qquad (1)$$

where $a > 1$ determines the allowed rate of change, e_{new} is the most recent error rate, and e_{old} is the previous error rate. This equation ensures that $1 > \lambda > 0$ which is required to avoid total extinction of the classifying data. Also note that

$$\lambda_{\text{new}} > \lambda_{\text{old}} \quad \text{if and only if} \quad e_{\text{new}} > e_{\text{old}}$$

so that observations are discarded more quickly in the case that the error rate has gone up, which is evidence that one of the populations has moved.

[1] A similar idea was suggested during a useful discussion with Shirin Bakhtari

5 Results

5.1 Simulated data

In order to illustrate the effect of the parameter λ on the error rate data were simulated from two models. In both models there were two classes and two variables, with one variable independent of class (i.e. noise) with distribution $N(0,2)$; the other variable had variance 1 with distribution

$$X = \begin{cases} f(t) + 1 + Z & \text{in class 1} \\ f(t) - 1 + Z & \text{in class 2} \end{cases}$$

where $Z \sim N(0,1)$ is a standard normal random variable and $f(t)$ denotes a dynamically moving boundary between the two classes, with

$$f(t) = \begin{cases} \sin(mt) & \text{in the first model} \\ mt & \text{in the second model} \end{cases}$$

and m is a parameter which controls the rate of change in each model. Taking $\gamma = \epsilon = 1$ we

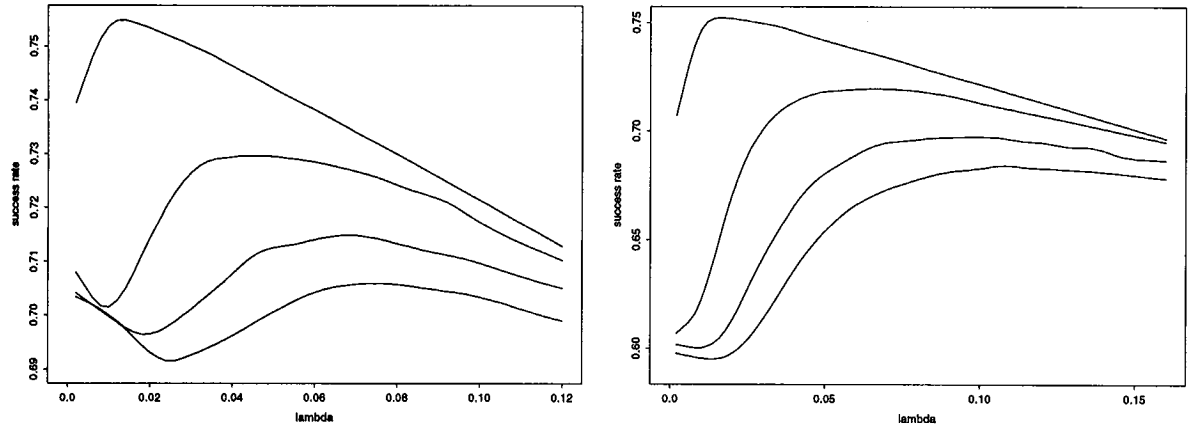

Figure 3: Smoothed success rates over 100 simulations. Left: sin model for drift, Right: linear model for drift. In each graph curves are (from top to bottom) for $m = 0.005, 0.04, 0.075, 0.1$.

found the error rate for a range of λ for a variety of values of $m = 0.005, 0.010, 0.015, \ldots, 0.1$ and one observation simulated (with equal class probabilities) at each time point $t = 1, \ldots, 1000$. A selection of results is given in figure 3. As expected the error rate increases with m. Also, note that the optimal choice of λ increases with m – but only up to a point in the sin model. It seems that for large m (in which case the boundary oscillates very rapidly) the required λ may result in having insufficient data in order to get a reliable classifier.

Note that the larger the value of λ the smaller the resulting training set which is used for future classification. Roughly speaking, if $\gamma = \epsilon$, then we expect there to be about $1/\lambda$ observations in the training set. Obviously the classification program will be faster for larger λ.

Additional work is needed in order to determine the best values for γ and ϵ. If $\epsilon \geq 1$ then this will "kill" any observation which has not already been found to be useful, and in general equilibrium indicates that these parameters should be roughly equal. Our simulations suggest that $0 < \gamma, \epsilon \leq 1$ works best, and in most cases we took $\gamma = \epsilon = 1$.

A second simulation was carried out on the sin model with the intention of examining the effect of a which controls the amount of adaptation that is possible in the decay parameter λ. In order to do this we need to examine the error rate at two time points, and estimate the error rate within a window of observations. This window corresponds roughly to a "batch size" and in many cases it will be pre-determined. Here, however, we investigate how the size of the window and a together affect the error rate. For $m = .005$ in the above sin model, we run 1000 simulations, and for each pair of window size, and a we obtain an average rate. These are then plotted in figure 4. The results here are perhaps disappointing, though still worth reporting since the surface is rather rough even when averaged out over so many simulations. Perhaps we can conclude that the choice of these parameters is not critical to the performance of the classifier, although the overall performance is better than that using a fixed λ. In general, we would expect the best choice of a to depend on the size of the window, and in many cases the batch size will be determined in isolation of the algorithm.

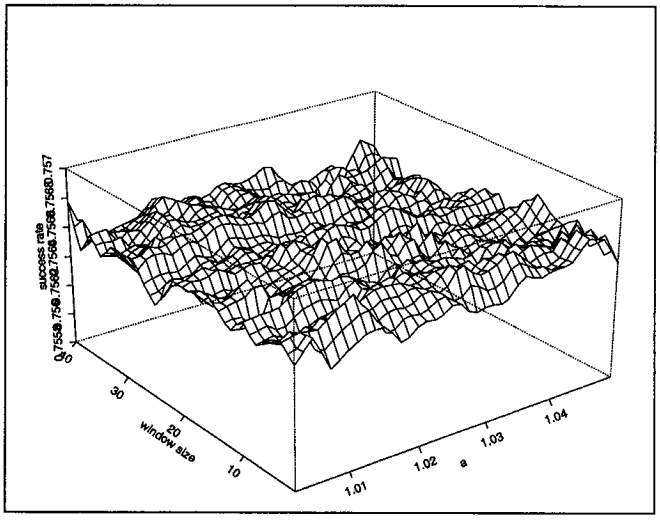

Figure 4: Success rates averaged over 1000 simulations for various choices of window (batch size) and a in the model given in equation (1). The maximum of the surface is at $a = 1.02$ and batch size 16, and corresponds to a slight improvement when no updating of λ is carried out.

5.2 Real data

The dataset concerns applications for credit and the class is whether or not the application is successful (*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. Figure 5 illustrates the results for the nearest neighbour adaptive classification. The initial value of λ was trained using the first 5000 observations, with the finding that $\lambda = 1.0001$ was optimal. In addition, we found the best updating parameter a (see equation (1)) using batch sizes of 1000. The plot shows the impact of dynamic learning since we have also "frozen" the training data and showed the equivalent error rate with no updating. The overall difference in error rates here is about 2%, but the dynamic version is faster since, on average, only 3334 observations are stored at any one time (rather than 5000), and the testing time is roughly proportional to the number of observations. For larger values of λ (possibly appropriate for other datasets) the difference in speed may be more dramatic. The effect of dynamic adjustment of λ according to equation

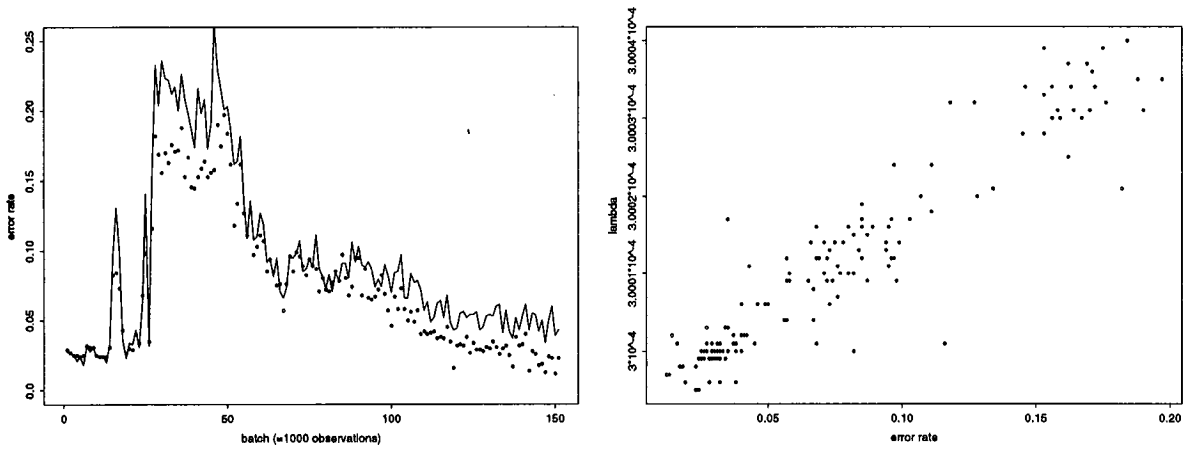

Figure 5: Left: Error rates for the adaptive nearest neighbour classifier (points) and a non-adaptive benchmark (line); Right: Relationship between the error rate and the adaptive λ.

(1) is only very slight: an improvement in the error rate of 0.007%, which corresponds to only 10 observations. However, the relationship between λ and the error rate is as desired, and for other datasets a more marked improvement may be seen.

The results for the dynamic linear discriminant show a similar pattern. In this case the improvement in accuracy is 4.8% compared with the non-dynamic version. However, the noticeable improvement occurs in batches $30 - 50$ in which the error rate suddenly increased.

6 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. For this reason it seems that any method should include a monitoring process even if this monitoring is not normally used to update the rules. In the above nearest neighbour approach, if there was no change to the class populations during training, then the best performance (though not the best speed) would be achieved with $\lambda = 0$. In this case, it is recommended that we always take $\lambda > 0$ in order to safeguard against this as well as speed up the classifier.

References

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.

Filosofov, L. V. (1992). Dynamic classification systems with learning from observation catalogs. *Cybernetics and Systems Analysis* **28**, 368–376.

Gibb, W. J., Auslander, D. M. and Griffin, J. C. (1994). Adaptive classification of myocardial electrogram wave-forms. *IEEE Transactions on Biomedical Engineering* **41**, 804–808.

Joseph, B. and Hanratty, F. W. (1993). Predictive control of quality in a batch manufacturing process using artificial neural-network models. *Industrial and Engineering Chemistry Research* **32**, 1951–1961.

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

Mardia, K. V., Kent, J. T. and Bibby, J. M. (1979). *Multivariate Analysis*. London: Academic Press.

McLachlan, G. J. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. New York: John Wiley.

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 supervised learning. Some basic issues and application aspects. In *submitted*.

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.

Smith, A. E. (1994). X-bar and R control chart interpretation using neural computing. *Interntaional Journal of Production Research* **32**, 309–320.

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.