Evaluating Usefulness for Dynamic Classification

Gholamreza Nakhaeizadeh
Daimler-Benz Research and Technology
89013 Ulm, Germany
nakhaeizadeh@dbag.ulm.daimlerbenz.com

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

Carsten Lanquillon
Daimler-Benz Research and Technology
89013 Ulm, Germany
carsten.lanquillon@dbag.ulm.daimlerbenz.com

Abstract

This paper develops the concept of *usefulness* in the context of supervised learning. We argue that usefulness can be used to improve the performance of classification rules (as measured by error rate), as well to reduce their storage (or their derivation). We also indicate how usefulness can be applied in a dynamic setting, in which the distribution of at least one class is changing with time. Three algorithms are used to exemplify our proposals. We first review a dynamic nearest neighbour classifier, and then develop dynamic versions of Learning Vector Quantization and a Radial Basis Function network. All the algorithms are adapted to capture dynamic aspects of real-world data sets by keeping a record of usefulness as well as considering the age of the observations. These methods are tried out on real data from the credit industry.[1]

Introduction

The learning task in classification is to obtain a classifier which can be applied later to determine the class of new, unseen observations. This task can usually be achieved by using a set of examples (the training data). In many practical situations in which the environment changes over time (concept shift) one needs to adapt the implemented classifier to retain its accuracy performance. Relevant examples include economic data (which may be affected by inflationary factors, changes in fiscal policy *etc.*), demographic data (which can show long-term trends over time) and biological data (dealing with growth of organisms *etc.*). More generally, technological or scientific advances can suddenly increase the accuracy of feature measurements so that noisy attributes become useful discriminators.

For the case that a concept shift is detected, Nakhaeizadeh *et al.* (1997) provide a collection of ideas for adaptive classification which can capture dynamic aspects of real-world data sets. Although implemented using some statistical methods, most of their ideas have a general character and could be applied to any supervised algorithm. They discuss two general approaches: the first one is based on the use of a monitoring process (akin to that used in quality control); the other one uses a combination of data editing – by age and propensity to mislead – to form a reduced data set.

The first approach is based on *manipulation* of the implemented **classifier**. The adaptation task in this approach results in a revised classifier that can be, for example, a weighted average of the old and new classification rules, where the new classification rule is learned by using recent training data. The second approach is just a relearning approach which uses a training set based on a reduced part of all of the observed examples up to the time of the relearning. Due to this fact, this approach is called *manipulation* of the **data**. In this approach, the manipulation (reduction) of data is useful because a lot of learning algorithms can not handle a large amount of data observed over a long period of time. Yet, even if the learning algorithms could handle a large amount of data, the storage of the data would be expensive and therefore in many cases not practicable. To reduce the data, Nakhaeizadeh *et al.* (1997) suggest the application of *usefulness*. Some aspects of this concept, especially in dealing with k-nearest neighbours is discussed in Nakhaeizadeh *et al.* (1996).

This concept of usefulness can be also be thought of in terms of a "similarity" rule in which we throw away observations in the current training set which are different – *i.e.* they are close in feature space, but have different class labels – from those recently observed. Alternatively, or in addition, 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. When deciding upon the representativeness of observations by their probabilities of class membership, however, this approach 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 current training set.

There are several methods which use a "forgetting" strategy to update the data. See, for example, the FLORA family of algorithms (Widmer & Kubat, 1992, 1993, 1996; Widmer, 1994) which generally monitor the error rates to update the rate of forgetting by use of a moving window. See also the reviews of Kubat & Widmer (1995) and Widmer (1997), and the algorithm FRAN (Kubat & Widmer, 1995) which is developed to handle concept drift in continuous domains. In this paper, the main question that should be addressed in reducing the amount of the training data is how the reduction should be performed so that the *quality* of the classification results are not affected. To answer this question, the tra-

ditional sampling approaches in statistics are the classical tools. These approaches, however, devote to all examples the same importance. Furthermore, they pay no attention to the classification algorithms that learn by using the sample data. To overcome this shortcoming, recently, John & Langley (1996) have suggested the concept *Probably Close Enough* that is based on dynamic sampling. Concept windowing (Quinlan 1993), different IBL approaches that are introduced by Aha *et al.* (1991) and various approaches on prototype learning – see, for example, Datta (1997) – can be viewed in this connection as well. In most of these approaches the reduction of the data set is performed by involving the learning algorithms in the focusing process. It should, however, be mentioned that involving the learning algorithms brings some advantages but causes an increase in the required computing time that in some cases could be considerable. In dealing with large spatial databases, Ester *et al.* (1995) introduce a method to focus on "representative" objects. This method selects the most central object as a representative, where the most central object in a data page is the object with minimal distance of its centre from the centre of the data page.

None of the above mentioned approaches, however, explicitly uses the concept of usefulness for reducing the number of data sets. A related method has been used in DARLING (Salganicoff, 1993) in which (old) examples are discarded if new examples which are close (in attribute space) are deemed to supersede them.

Note also that this problem is somewhat different to **incremental learning** – see Utgoff (1989, 1994) and Crawford (1989) for example – in which one of the design goals is that the tree that is produced by the incremental algorithm should depend only on the set of instances, without regard to the sequence in which those instances were presented. However, if there is population drift or concept change some of the old data must be downweighted since it is no longer representative of the current population.

In the following section, we develop the concept of keeping a record of usefulness, which was initially used in a nearest-neighbour classifier, and we show how this can be used more generally. The remainder of the paper focuses on *batch* learning in which the observations will be naturally grouped (into a specified time period, for example), and any learning and classification will be carried out in this discrete framework.

Having introduced the concept of usefulness in the next section, the following two two sections outline the approaches of Radial Basis Function networks and Learning Vector Quantization and these are extended to handle dynamic data by using a record of usefulness. In a final section we apply these methods to some credit data.

Record of Usefulness

Initially, we developed a dynamic 1-nearest neighbour algorithm 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:

- 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).
- 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 of each example then behaves like a Markov Chain with an absorbing barrier at 0, whose properties (for example, the drift) can be studied in order to get some idea of suitable choices for the three parameters: λ, γ and ϵ.

Note that this use of weights is distinct from that used by Cost & 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. Further discussion and results for this dynamic nearest neighbour approach can be found in Nakhaeizadeh *et al.* (1996).

Another approach of defining a record of usefulness is based on the following idea. An observation that frequently appears in one class is very useful for establishing a classifier because it is very likely to occur frequently in the future and is therefore representative. However, if an observation has occurrences in different classes (caused by noise, for example), it seems less useful, even though it may frequently occur. Assuming a binary decision problem, in the most extreme case, a observation has the same number of occurrences in both classes. Then the true class membership is unknown and the observation is of no use. Hence an observation is the more useful, the more frequently it occurs in only one class. Formally, we define the usefulness μ_x of an observation x for the binary decision problem as

$$\mu_x = | \, n_1(x) - n_2(x) \, |$$

where $n_i(x)$ denotes the number of occurrences of observation x in class i. Each observation x is labeled with that class having the most occurrences of x. The observation x, its class label and the usefulness μ_x are kept in a list that is sorted by μ_x. This record of usefulness will be used for the radial basis function networks in the next section.

Radial Basis Function Networks

Introduction

Radial basis function (RBF) networks were introduced by Broomhead & Lowe (1988) and are based on the theory of functional approximation. They are known as universal approximators (Park & Sandberg, 1991), and are as well suited for classification problems.

Given a training set of n examples (x_i, y_i) $(x_i \in \mathbb{R}^k, y \in \mathbb{R})$, y is to be approximated by a function f of the form

$$f(x) = \sum_{i=1}^{p} w_i h_i(x) \quad (1)$$

where the h_i are p $(p \leq n)$ radial basis functions with a response that either increases or decreases monotonically with distance from a central point. A typical radial basis function is the Gaussian kernel.

Since the function f is to be an approximation of the given training set, the following error function E is to be minimized:

$$E[f] = \sum_{j=1}^{n} (y_j - f(x_j))^2 .$$

In case the RBF network is used for classification, the output y requires transformation onto the set of predefined classes. In a binary classification problem, for example, a simple threshold value θ may be introduced for separating the output values for each class.

If all training examples are distinct and each observation is selected as a centre $(p = n)$ then the weights w_i of equation (1) can simply be computed by solving a system of linear equations (see, for example, Orr (1996)) such that the error E vanishes. Obviously, this approach is not suitable for large training sets because of its computational complexity. Moreover, when using too many centres, this approach easily suffers from overfitting. A simple way of reducing computing time and increasing the ability of generalization is to select a smaller number of centres. However, this raises the question of how to best choose the number and location of the centres.

Selecting Parameters

Considering equation (1) as the model of our RBF network, we assume that Gaussian kernels are used as radial basis functions and the distance is measured by the Euclidean norm. Thus the number, location and size of centres (p, c_i and r_i) and the weights w_i remain as free parameters ($i = 1, \ldots, p$).

Many ideas have been proposed as how to choose the parameters of an RBF network. On the one hand, the method can be as simple as randomly selecting some input observations as centres, using a predefined radius for each centre and calculating the weights w_i between hidden and output layer. On the other hand, a comprehensive approach that iteratively adjusts all parameters to optimize the performance of the network can be applied. Whatever method is chosen, the selection of centres should respect the structure of the training set. There is no need for centres in empty regions of the input space or an abundance of centres in densely populated parts; see Schürmann (1996, p. 243). This is why approaches that randomly select examples from the training data are often doomed to failure. Yet iteratively learning all parameters from the training set is very time consuming. We will use the following implementation as a compromise.

We implemented a dynamic "supervised" clustering algorithm that makes use of the record of usefulness as described in the end of section 2 to improve the process of selecting centres. The algorithm is dynamic in the sense that the number of clusters is not fixed but only limited by a given number. Although clustering is usually unsupervised, this algorithm is characterized as supervised because it makes use of the class labels of the observations. When mapping a observation to the nearest cluster, the distance of that observation to a cluster centre of a different class is doubled in order to keep classes separated. Given the maximum number p of cluster centres to be created and the training set T of observations with their class labels, the pseudo code of the algorithm is as follows:

Dynamic-Supervised-Clustering (p, T)

create one cluster for each class;
map examples to cluster of corresponding class;
repeat
  calculate cluster centres according to current mapping;
  map all examples to the closest cluster;
  if *current number of clusters smaller than* p
  and *any examples mapped to cluster of wrong class*
    find cluster with most misclassifications weighted by usefulness;
    create new cluster;
    use misclassified example with largest usefulness as new centre;
until *mapping of examples to clusters is unchanged;*

Simulations on our credit data set have shown that the performance of the RBF networks are very sensitive to the selection of the radii. The error rates drastically increase when the radii are too small or too large. A suitable selection of radii should ensure that at least one centre responds to each observation, and conversely, that no pair of centres belonging to distinct classes respond to observations with a high and nearly identical activation. Obviously, in general this can not be achieved by using a fixed radius for all centres. Therefore we evaluate the radius of each centre as the distance to the nearest centre of another class, multiplied by a constant scaling parameter.

We make use of the Stuttgart Neural Network Simulator (SNNS) to calculate the weights between the output and hidden layer. The obtained centres and radii are used to create a SNNS network file, and a data set for training the weights is obtained by selecting all unique observations from the record of usefulness.

Dynamic Implementation

When trying to adapt an RBF network to dynamic changes, two basic approaches that make use of the concept of usefulness may be considered. First, the training set could be updated, using the same method of creating the RBF network after each batch. Second, the network itself could be updated, for example by altering, adding or dropping centres, according to the observed changes. And of course, both approaches can be combined.

In order to detect changes, we use the idea of the basic Shewhart quality control chart for monitoring the error rates of our classifiers (see Taylor *et al.* (1997), for example). The observations will be processed in batches. This will average and thus reduce stochastic perturbations. For each previous batch i, the error rate e_i is known. We estimate the expected error rate $\bar{e}$ from previous error rates as their mean value weighted with the corresponding batch sizes. Then let $V_j = \bar{e} + j\sqrt{\bar{e}(1-\bar{e})/n}$, $j = 1,3$ indicate warning and action limits, respectively, for the error rates. Suppose that the error rate of the current batch of size n is e_{curr}. If $e_{\text{curr}} \leq V_1$ the performance of the classifier is acceptable and we obtain a monitor status **ok**. If $e_{\text{curr}} > V_3$ some major changes are likely to have occurred which is denoted by the status **action**. Otherwise, we obtain a **warning** status.

For the adaptive approach in this paper, we apply the method of creating the RBF network as described in the previous section only in case of a **warning** or **action** status based on the current record of usefulness which depends on the monitor status as follows:

- **ok**: The current record of usefulness is aged by the factor $1 - \gamma_{\min}$ ($0 \leq \gamma_{\min} \leq 1$). Since the performance is still good, $\gamma_{\min}$ should be rather small, thus keeping the record of usefulness almost unchanged. The aged record of usefulness and a record obtained from the current batch are merged to a new record of usefulness. This is done by directly keeping all examples with their usefulness that occur only in one of the records. If an example occurs in both records, the new value of its usefulness is determined as a function of the current values. In case the example has the same class label in both records, these values may simply be added. Otherwise the absolute value of the difference may be applied and the class label is determined by the class label of the example that was more useful.
- **warning**: The record of usefulness is updated in the same way, except that the ageing factor is $1 - \gamma$, where γ is linear function of e_{curr} between $\gamma_{\min}$ and $\gamma_{\max}$ ($\gamma_{\min} \leq \gamma_{\max} \leq 1$).
- **action**: The old record of usefulness is assumed not to be representative any more. Therefore, a new record is created from the observations of the most recent batch.

Learning Vector Quantization

Description of the basic package

Learning Vector Quantization (LVQ) is a public domain algorithm of a method due to Kohonen (1989), although it has a longer history; see Hertz *et al.* (1991) for earlier references. Learning Vector Quantization is an example of competitive learning often used for data compression.

Kohonen's (1989) procedure is just the use of competitive learning in a supervised learning problem. In both Kohonen and LVQ, classification is based on a nearest neighbour rule, the difference being in the dimensionality of the output space. Our description follows the programs contained in LVQ_PAK which is documented in Kohonen *et al.* (1995). Assume that a number of "codebook vectors" m_i (free parameter vectors) are placed into the input space to approximate various domains of the input vector x by their quantized values. Usually several codebook vectors are assigned to each class of x values, and x is then decided to belong to the same class to which the nearest m_i belongs. Let

$$c = arg\min_i\{\|x - m_i\|\}$$

define the nearest m_i to x, denoted by m_c.

Values for the m_i that approximately minimize the misclassification errors in the above nearest-neighbour classification can be found as asymptotic values in the learning process. In all, there are several variants of the program (LVQ1, OLVQ1, LVQ2.1, LVQ3) and about 6 parameters to be selected: α (the learning rate parameter); ϵ (the relative learning rate parameter used in LVQ3); the relative width of the window; k (the number of nearest neighbours used for initial rejection by k-NN misclassification); and the number of steps used in training the vectors. However, experience gives reasonable guidance on all the above. The number of codebook vectors to use – and whether to select them evenly, or in proportion to the prior probabilities of class membership – are perhaps the most sensitive choices and this selection is not such a simple matter. More codebook vectors can lead to *overfitting* with poor performance on the test data.

Examining the usefulness of codebook vectors in a dynamic setting

It is possible that, in the presence of moving populations, the codebook vectors that have been learned will become less useful. In this case, it will be necessary to revise the codebook vectors by relearning. By analogy with the nearest neighbour implementation above, the current codebook vectors are assigned a weight which decreases with time. Then, codebook vectors are updated when their weight drops below a certain threshold. A further refinement would be to use the weights in the classification procedure itself. However, this dynamic adaptation would require detailed modification of the source code of LVQ_PAK, so instead we implemented a variant still based on the concept of usefulness.

When comparing two classifiers on the same set of data, it is straightforward to cross-tabulate the number of observations correctly/incorrectly classified by each method. McNemar's test (or an equivalent χ^2 test) can then be used to decide whether the probabilities of misclassification are equal. In our approach we assess the usefulness of a codebook vector by comparing the classification of $\mathcal{M}$ with $\mathcal{M}_{(i)}$, where $\mathcal{M}$ is the full set of codebook vectors and $\mathcal{M}_{(i)}$ is the full set with the ith codebook vector deleted. This "leave-one-out" procedure then carries out a formal test (at the 5% level) to determine the usefulness of each codebook vector in turn. The *overall* usefulness of $\mathcal{M}$ is then assessed by fixing two integers, say W_1 and W_2 which are akin to the warning and action limits used above. In this case the values of W_i will be determined by – for example proportional to – the number of codebook vectors. In the case that the number of *useful* codebook vectors is less than W_1 then the set of codebook vectors is completely relearned, whereas in the

case that the number of *useful* codebook vectors is between W_1 and W_2 then minor revision of the codebook vectors is made. Note that, unlike the limits used in monitoring the RBF network, we make no assessment of the (change in) error rates as a possible consequence of population movement. In this case the monitoring is solely on the overall usefulness of the set of codebook vectors. It is clear that the larger the values of the W_i, the more frequent will the classification rule be relearned, but more experience is needed to determine suitable guidelines for the W_i.

Results

We have applied the above methods to one set of data. At this stage, we are more interested in how much improvement can be achieved in using an adaptive classifier rather than either continually relearning, or else taking no action at all. So it is of secondary importance to compare the results across different classifiers, and in any case it would be premature to draw any conclusions on the basis of only one data set. The data set 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 selected (using a stepwise linear regression) for classification purposes.

RBF Network

The adaptive RBF network is applied to the credit data set using the dynamic supervised clustering approach with a maximum of 10 clusters. While the ageing parameter $\gamma_{\max}$ is fixed to one, different values for the minimum ageing parameter $\gamma_{\min}$ are tested. The data set was split into a initial training set of size $5\,000$ and about 151 batches of size $1\,000$ for simulating the time flow. The results are compared to the *no-learn* and the *re-learn* approach. While the RBF network is created using the initial training data and is then left unchanged in the no-learn approach, it is learned from scratch after each batch using data only from the most recent batch in the re-learn approach. The results are presented in Table1.

method		$\gamma_{\min}$			
no-learn	re-learn	0.0	0.2	0.3	0.4
9.18	8.05	7.84	7.82	7.79	7.82

Table 1: Average error rates (%) of RBF network using different approaches.

Compared to the no-learn approach, the performance of the re-learn approach demonstrates that there are some changes within the stream of observations to be classified. Hence, the need for altering the applied classifier becomes obvious. All adaptive approaches performed better than the corresponding non-adaptive ones. However, the increase in performance as measured by the error rates does not seem to be really significant. Note, however, that the adaptive approaches updated the network only about 20 times throughout the whole simulation (compared to the 151 updates after each batch in the re-learn approach) thus saving some computing time. The best result was obtained for $\gamma_{\min} = 0.3$. However, determining a good value of $\gamma_{\min}$ based on the initial training set is impossible because future changes within the stream of observations are unknown. In further research, however, methods for adjusting this parameter while the classifier is already applied could be considered. Note that using an ageing factor $\gamma \geq \gamma_{\min} > 0$ is important for reducing the size of the training set. Otherwise, the number of observations kept for training will constantly increase.

Learning Vector Quantization

We used 10 codebook vectors in all the results reported here which is the same as the number of clusters used for the RBF networks. The codebook vectors were selected in proportion to the relative frequencies in the training data. Initially OLVQ1 was run for 400 iterations, then LVQ3 was run for $1\,500$ iterations with $\alpha = 0.03$, $\epsilon = 0.1$ and relative window size 0.3.

In the no-learn approach the codebook vectors were unchanged after training which used, as before, $5\,000$ observations. In the re-learn approach, the codebooks were completely relearned (using the same set of parameters) after each new batch. The first adaptive approach used monitoring values of $W_1 = 3$ and $W_2 = 4$. In the case that the number of useful codebook vectors was less than or equal to 3 a complete relearning took place; if the number of useful codebook vectors was equal to 4 then the existing codebook vectors were simply adjusted using OLVQ1 and LVQ3, but initial values were not re-set. This means that, in this case, the number of codebook vectors in each class would not be changed.

A second adaptive approach was also used in which, in addition to the strategy above, we also "aged" the set of codebook vectors. In this case, relearning was forced to take place after A batches, if no relearning had been indicated by the W_i limits. So the ageing clock was re-set each time any relearning was done.

The error rates for these 4 approaches are given in Table 2. For comparison, we also give the error rate produced by taking the best of the no-learn and re-learn methods. It is clear from the results that the adaptive method, which takes account of the usefulness of the codebook vectors, gives the lowest error rate. It also seems that sensible choices of A can lead to further improvements. In practice the value of A

updating method	error rate	Ageing factor (A) 10	20	30
no-learn	10.52			
re-learn	10.09			
best(no-learn,re-learn)	9.95			
adaptive	9.65	9.62	9.28	9.29

Table 2: Error rates (%) for several dynamic approaches, including the ageing modification with a clock reset after A batches.

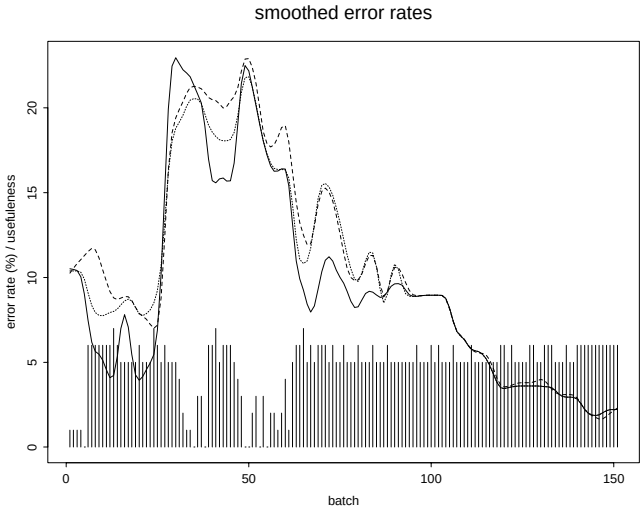

Figure 1: Smoothed error rates for three approaches: adaptive approach (continuous line); re-learn (short dashed line); no-learn (long dashed line).The number of codebook vectors is also plotted in the vertical lines.

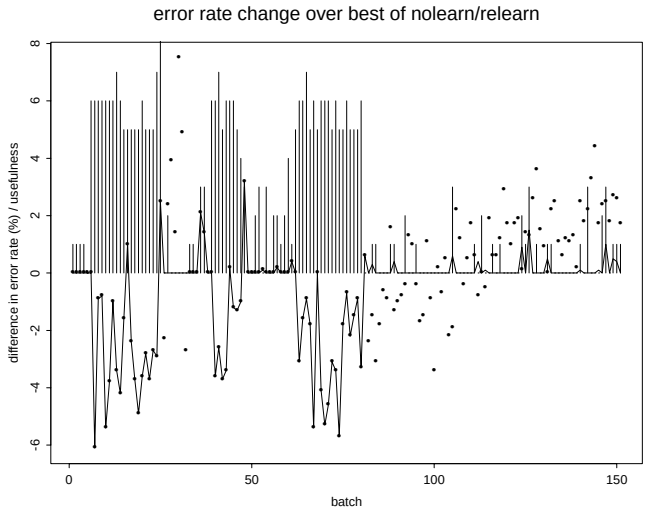

Figure 2: The difference in error rate for the simple adaptive classifier (points) and the ageing adaptive classifier with $A = 20$ (line) compared with the best of the no-learn and re-learn approaches at each time point. Values below 0 have a smaller error rate. Also shown is the number of useful codebook vectors for the ageing adaptve classifier (vertical lines). At time points 24 and 80, re-learning took place due to "old" codebook vectors rather than a lack of usefulness.

could be based on a longer period of training data, or even by some adaptive method based on recent history as was used in the nearest neighbour classifier above.

The smoothed error rates for the no-learn, re-learn and adaptive methods are shown in Figure 1, which also indicates the drift in the population and the consequent attempts of the adaptive method to react.

The difference in error rates *vs.* time are shown in Figure 2. The number of useful – as determined by a McNemar test on a leave-one codebook-out procedure – codebook vectors is also shown. From this it can be seen when this value drops below 3 or 4 which then corresponds to retraining.

A comparison of two adaptive classifiers (with $A = \infty$ and $A = 20$) with the best of the no-learn and re-learn at each time point is also shown in Figure 2. The "ageing" adaptive classifier is modified at batches 24 and 80. This is helpful at batch 24 but the effect is less marked at batch 80; much more re-learning takes place after this time due to useless codebooks as the program struggles to find a classifier to beat the default (in which every observation is classified as good credit). Towards the end of the time period, however, this strategy seems to be superior. The overall error rate also shows a small improvement when using this modification.

Conclusions

Overall, the RBF network works better on this data set than LVQ, but as mentioned previously we are more concerned with the improvement that can be achieved in using an adaptive approach rather than not relearning or updating after each batch. Tables 1 and 2 show that for both algorithms the adaptive approach is superior, and the diagnostics shown in Figures 1 and 2 indicate that the method is working as intended though there is clearly more work required on some of the parameter selection.

In both the RBF and LVQ experiments it was found that an ageing component improved the performance, even though the error rate or proposed measure of usefulness did not suggest the need for any updating. This is not totally satisfactory, and suggests the need for further research of how to optimally determine the ageing factor, based on certain characteristics of the population movements.

The proposed "record of usefulness" has been used for two different purposes. In the case of the RBF network, *usefulness* is used in order to get a template set of examples that is rather small, in good quality and fully representative. Then the reduced data set is used for selecting prototypes (called "centres" in RBF terminology) and calculating the weights in the learning process. This process is fast, and thus it was decided to completely relearn the network from scratch based on the manipulated training set in case the monitoring process said so. A further possibility is to reuse the prototypes and update them according the detected change. In the case of LVQ the concept of usefulness is used to judge the selected prototypes (called "codebook vectors" in LVQ terminology) and even to detect changes. In further research, it would be of interest to combine these two ideas.

References

Aha, D.; Kibler, D.; and Albert, M. 1991. Instance-based learning algorithms. *Machine Learning* 6:37–66.

Broomhead, D. S., and Lowe, D. 1988. Mutlivariate functional interpolation and adaptive networks. *Complex Systems* 2:321–355.

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

Crawford, S. L. 1989. Extensions to the cart algorithm. *International Journal of Man-Machine Studies* 31:197–217.

Data, P. 1997. *Characteristic Concept Representations*. Ph.D. Dissertation, University of California Irvine.

Ester, M.; Kriegel, H. P.; and Xu, X. 1995. Knowledge discovery in databases: Focusing techniques for efficient class identification. In *Proceedings of the Fourth International Symposium on Large Spatial Databases (SSD'95)*.

John, G. H., and Langley, P. 1996. Static versus dynamic sampling for data mining. In Simoudis, E.; Han, J.; and Fayyad, U., eds., *Proceedings the Second International Conference on Knowledge Discovery & Data Mining*, 367–370. Menlo Park, California: AAAI Press.

Kohonen, T.; J, H.; Kangas, J.; Laaksonen, J.; and Torkkola, K. 1995. LVQ_PAK. The learning vector quantization program package version 3.1. Technical report, Laboratory of Computer and Information Science, Helsinki University of Technology.

Kohonen, T. 1989. *Self-Organization and Associative Memory*. Berlin: Springer-Verlag, 3rd edition.

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

Kubat, M., and Widmer, G. 1995b. Adapting to drift in continuous domains. In *Proceedings of the 8th European Conference on Machine Learning, Heraclion, Crete, Greece*, 307–310. Springer Verlag.

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*, 55–64. Menlo Park, CA: AAAI Press.

Nakhaeizadeh, G.; Taylor, C. C.; and Kunisch, G. 1997. Dynamic supervised learning: Some basic issues and application aspects. In Klar, R., and Opitz, O., eds., *Classification and Knowledge Organization*, 123–135. Berlin: Springer-Verlag.

Orr, M. J. L. 1996. Introduction to radial basis function networks. Centre for Cognitive Science, University of Edinburgh, Scotland. hyper-text document: http://www.cns.ed.ac.uk/people/mark/intro/intro.html.

Park, J., and Sandberg, I. W. 1991. Universal approximation using radial-basis-funtion networks. *Neural Computation* 4:246–257.

Quinlan, R. 1993. *C4.5: Programs for machine Learning*. Morgan Kaufmann.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge, UK: Cambridge University Press.

Salganicoff, M. 1993. Density-adaptive learning and forgetting. In *Proceedings of the Tenth International Conference on Machine Learning*, 276–283. San Mateo, CA: Morgan Kaufmann.

Schürmann, J. 1996. *Pattern Classification: A Unified View of Statistical and Neural Approaches.* John Wiley & Sons, Inc.

Stuttgart Neural Network Simulator, ftp.informatik.uni-stuttgart.de. 1997. subdirectory: /pub/SNNS.

Taylor, C. C.; Nakhaeizadeh, G.; and Lanquillon, C. 1997. Structural change and classification. In Nakhaeizadeh, G.; Bruha, I.; and Taylor, C. C., eds., *ECML '97 – Workshop Notes on Dynamically Changing Domains: Theory Revision and Context Dependence Issues*, 67–77. Faculty of Informatics and Statistics, University of Economics, Prague: Laboratory of Intelligent Systems.

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.

Widmer, G., and Kubat, M. 1992. Learning flexible concepts from streams of examples: FLORA2. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*. Chichester, UK: Wiley.

Widmer, G., and Kubat, M. 1993. Effective learning in dynamic environments by explicit context tracking. In *Proceedings of the Sixth European Conference on Machine Learning*, 227–243. Berlin: Springer-Verlag.

Widmer, G., and Kubat, M. 1996. Learning in the presence of concept drift and hidden contexts. *Machine Learning* 23:69–101.

Widmer, G. 1994. Combining robustness and flexibility in learning drifting concepts. In *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-93)*, 468–472. Chichester, UK: Wiley.

Widmer, G. 1997. Learning in dynamically changing domains: recent contributions of machine learning. In Nakhaeizadeh, G.; Bruha, I.; and Taylor, C. C., eds., *ECML '97 – Workshop Notes on Dynamically Changing Domains: Theory Revision and Context Dependence Issues*, 79–83. Faculty of Informatics and Statistics, University of Economics, Prague: Laboratory of Intelligent Systems.