Neurocomputing 50 (2003) 461–472

NEUROCOMPUTING

www.elsevier.com/locate/neucom

A training algorithm for classification of high-dimensional data

Armando Vieira[a,b,*], Nuno Barradas[b,c]

[a]*ISEP, Dep de Fisica, R. São Tomé, 4200 Porto, Portugal, and Centro de Física Computacional, Universidade Coimbra, 3000 Coimbra, Portugal*

[b]*Instituto Tecnológico e Nuclear, Estrada Nacional 10, Apartado 21, 2686-953 Sacavém, Portugal*

[c]*Centro de Fí sica Nuclear da Universidade de Lisboa, Av. Prof. Gama Pinto 2, 1699 Lisboa Codex, Portugal*

Received 20 July 2001; accepted 11 May 2002

Abstract

We propose an algorithm for training multi layer preceptrons (MLP) for classification problems, that we named hidden layer learning vector quantization. It consists of applying learning vector quantization to the last hidden layer of a MLP and it gave very successful results on problems containing a large number of correlated inputs. It was applied with excellent results on classification of Rutherford backscattering spectra and to a benchmark problem of image recognition.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Classification; Learning vector quantization; Hidden layer learning vector quantization; Feature extraction; Rutherford backscattering

1. Introduction

The cornerstone of classification algorithms is to extract relevant features from raw data so that discrimination between classes can be easily performed. However, in many cases these features are not easily identifiable and appropriate data transformations have to be applied. Linear transformations, like Fourier transform or principal component analysis, are easiest to perform, but may not be sufficient for some hard problems like image recognition. For instance, it is known that the visual cortex uses heavy

* Corresponding author. ISEP, Dep de Fisica, R. Saõ Tomé, 4200 Porto, Portugal. Tel.: +351-934561748.
E-mail address: asv@isep.ipp.pt (A. Vieira).

pre-processing before presenting the light impulses to the brain, which is an indication that features are extracted at intermediate layers of neurons [7].

Neural networks have been used in classification problems with considerable success, either with supervised or unsupervised learning [14]. A common approach for supervised learning is multi layer preceptrons (MLP) with a 1-of-*c* coding scheme, where the number of outputs is the same as the number of classes [19]. To train the net, we have to choose an error function and an error minimization algorithm. For the error function, the sum of squares or the cross entropy, derived from the maximum likelihood principle, are the most common. The backpropagation algorithm [13], or some of its variants, are widely used for training. The network architecture and learning parameters can be obtained using an optimization procedure, like genetic algorithms, or simply guesses based on some heuristics.

Several algorithms have been used for competitive learning, mainly in unsupervised classification problems [10]. Among them is learning vector quantization (LVQ) which is a common approach to data clustering [9]. This algorithm projects the data onto a set of representative vectors, called prototypes or code vectors, corresponding to each of the classes into which the input space is divided. LVQ is a data compression technique particularly useful for feature extraction in unsupervized learning. However, it usually performs badly on supervized learning.

Although competitive for some problems, both MLP and LVQ are not very efficient, for instance, in problems with high-dimensional inputs or when patterns to be classified are similar, particularly when the inputs components are correlated.

The objective of this work is to propose a new algorithm to train MLP that is more efficient in defining decision surfaces in the feature space, thus achieving better results in hard classification problems. We apply it to the problem of Rutherford backscattering (RBS) spectra classification [17], and to other problems, including a benchmark test of image recognition.

2. RBS technique and data simulation

RBS is an ion beam analysis technique extensively used in laboratories for compositional analysis of thin films [16]. From the experimental data, one can extract the elemental depth profile of thin films. Although this task is simple in principle, since RBS is based on classical mechanics, it can be very time-consuming, and in practice some data are too hard to analyse with traditional methods. Recently, this task was simplified by the use of computational techniques that may lead to instantaneous automated RBS analysis in a production line [2].

Among these techniques are artificial neural networks (ANN), recently applied on RBS data analysis of Ge implanted in Si [3] and Er implanted in sapphire [4], with good results. The data used for training consists of generated RBS spectra for all conditions that are expect to occur in experiments. The space of implant, beam, and detection parameters (training space) used was not uniform. Instead, more training spectra were generated for the beam and detection conditions that are more common in real experimental situations. We thus provide a larger amount of training examples

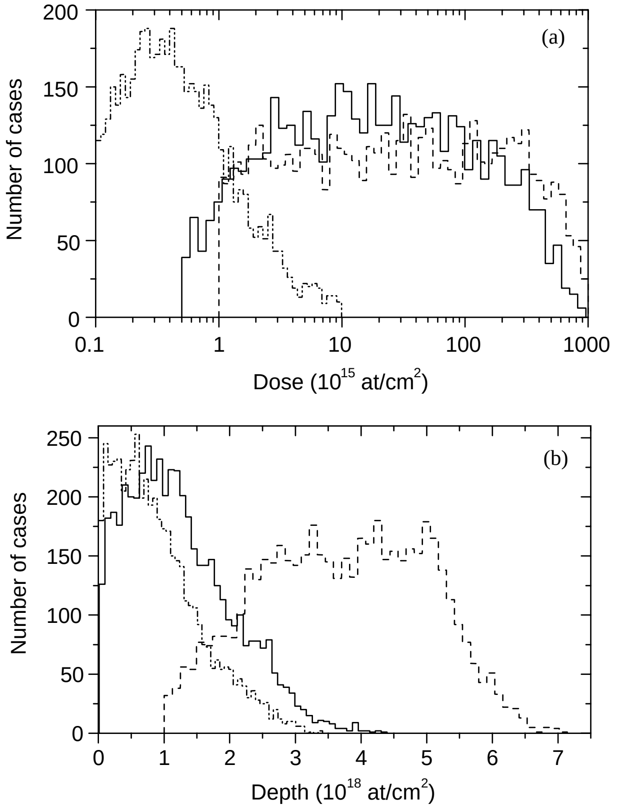

Fig. 1. Dose and depth distribution considered for the RBS classes.

corresponding to common and difficult cases. The training space used is shown in Fig. 1.

In previous work, we notice that the reliability of using ANN for data interpretation varies widely. The reason is that we have a large variety of spectra corresponding to distinct conditions of implantation and data analysis. The implanted doses ranges in three orders of magnitude, from 1×10^{14} to 1×10^{17} at/cm^2, and the implanted depth from 1×10^{17} to 7×10^{18} at/cm^2. Since this represents a large diversity of samples that may hinder data analysis, we decided to divide the data into three categories: class 1, class 2 and class 3. Provided that we have a reliable classification method to discriminate these categories, a specialized ANN for each of these classes will certainly reduce the interpretation error.

The spectra were classified according to the following criteria: class 1—samples with high Ge doses a Ge peak well separated from the Si background; class 2—Ge peak superimposed to the Si signal; and finally class 3—all other cases, that is, small Ge dose with separated signals. Class 1 should be the easiest to analyse since the Ge peak is large enough and well separated. Class 2 will be harder and class 3 the most difficult due to the small Ge signal with respect to the background noise. Fig. 1 presents the

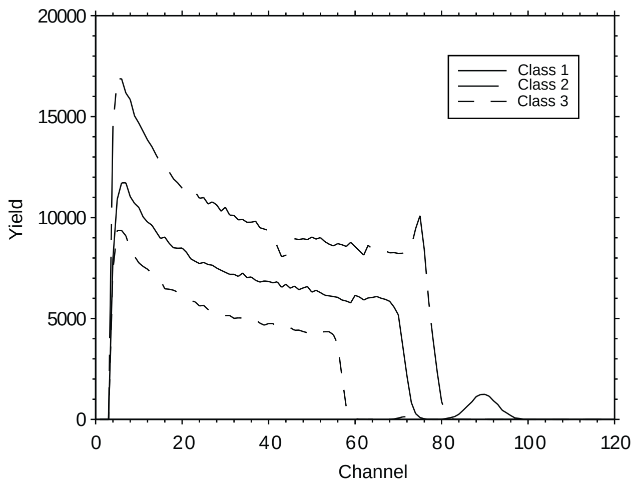

Fig. 2. Representative spectrum for each of the three RBS classes.

distribution of relevant parameters used to build the training set for each of the three classes. Notice the superposition of classes.

We should note that some spectra could not be clearly labeled as classes 1, 2 or 3. For instance, the transition from class 3 to class 1, which occurs when the Ge peak is high enough with respect to noise level, is not abrupt but rather smooth. The same occurs for spectra moving from class 2 to class 1, as partial superposition occurs in some cases. In order to take into account these smooth transitions, we created an identical database, but instead of classifying a spectrum into a unique class, we estimated a class membership probability. This proved to be a better approach, leading to a lower classification error. Some typical spectra for each of the three classes are presented in Fig. 2.

3. Classification of RBS spectra with an MLP

The separation of spectra into the three classes considered is a difficult task. For instance, a spectrum from class 2 with a small dose is practically indistinguishable from the spectra of class 3. The same ambiguity occurs between spectra of class 1 with low Ge doses and spectra of class 3 with relatively high doses.

First we used a MLP trained with backpropagation to classify the data. For the inputs we choose the yield of 128 channels, normalized to the charge—solid angle product, the beam energy and energy resolution, the angle of incidence, the scattering angle and the pileup. We used the 1-of-*c* coding scheme, and therefore there are three outputs—one for each class. We used a unipolar activation function.

The sum-of-squares error function is not the most appropriate for classification problems since it was derived from maximum likelihood assumption of Gaussian distributed target data. However, the 1-of-*c* coding scheme are binary, hence far from having a Gaussian distribution. A better alternative is to use a cross-entropy error function, which

Table 1
Misclassification errors for different MLP architectures

ANN architecture	Train set	Test set
(I, 100, O)	20.99	39.01
(I, 250, O)	17.34	33.66
(I, 100, 80, O)	11.99	17.67
(I, 100, 50, 20, O)	13.99	17.01
(I, 100, 80, 50, O)	10.01	13.67
(I, 100, 80, 80, O)	9.33	15.67
(I, 100, 50, 100, O)	10.01	13.99
(I, 100, 80, 80, 50, O)	10.67	13.66
(I, 100, 80, 50, 30, 20, O)	12.67	17.67

for a two-class problem is [5]

$$E = -\sum_n (t^n \ln y^n + (1 - t^n)\ln(1 - y^n)), \tag{1}$$

where t^n are the target values (0 or 1) and y^n the actual outputs of the network. Using a sigmoid output activation function, the error term applied to the output layer is just

$$\delta^n = (y^n - t^n). \tag{2}$$

The cross-entropy error function performs better than sum-of-squares when estimating small probabilities. The use of the logistic activation function for the outputs allow us to interpret their values as probabilities of presence to the corresponding classes [12].

Since we have a large number of examples to train an MLP, its specific architecture is not determinant. We test several configurations with between 1 and 5 hidden layers, and different numbers of nodes in each hidden layer, using, respectively, 15,000 and 2000 generated spectra for training and test. In each set, one-third of the examples belong to one of the three categories. The learning rate and the momentum were fixed at 0.1 and 0.25, respectively. The results obtained are presented in Table 1.

With a single hidden layer the training and test error decreases when the number of nodes increases from 100 to 250. However, the test error remains very large, indicating that a single hidden layer may be inefficient for classifying RBS spectra. Next, we consider two hidden layers with 100 and 80 nodes, which leads to a decreasing in the test and training error by a factor of two. Increasing the number of layers to three, while decreasing the number of nodes, leads to slightly worse performance. We hence kept two hidden layers and added one extra layer with 50 nodes, which led to a further reduction in the test set error.

Increasing, further, the number of nodes of the third hidden layer to 80 decreases the error in the training set, but the test set error becomes worse, indicating overtraining. Increasing the number of layers does not have any improvement. We finally chose the architecture consisting of three hidden layers, with 100, 80 and 50 nodes which is the ANN with the smallest test set error: 13.67%.

A comment should be made concerning the use of such a large ANN, with about 25,000 weights. It is known that an MLP with a single hidden layer is capable of discriminating between arbitrary complex decision regions. However, in some difficult problems the inclusion of more hidden layers may achieve better results at the cost of a longer training.

4. Hidden layer learning vector quantization (H-LVQ)

4.1. Learning vector quantization

Vector quantization is a data compression technique to encode a multidimensional input signal into representative code vectors, one for each class in which we want to discriminate the data. The code vector of each class c_i is composed of a set of weights $\vec{w}_{ci}$, obtained by competitive training. There are several training algorithms, but for supervised LVQ an iterative process adjusts the weights, initially random, where the increments are given by

$$\begin{aligned} \Delta\vec{w}_{ci} &= \alpha(t)(\vec{x} - \vec{w}_{c_i}) \quad \text{if } \vec{x} \in \text{class } c_i, \\ \Delta\vec{w}_{ci} &= 0 \quad\quad\quad\quad\quad\;\; \text{if } \vec{x} \notin \text{class } c_{hi}, \end{aligned} \tag{3}$$

where $\alpha(t)$ is the learning rate, which should decrease with iteration t to guarantee convergence. We choose the following expression:

$$\alpha(t) = \alpha_0 \times (0.1)^{t/N_I}, \tag{4}$$

where N_I is the number of training examples presented to the network, and $\alpha_0 \in [0, 1]$.

4.2. Hidden layer LVQ

During classification, the role played by the hidden layers of MLP is to find the weights of the final layer so as to produce an optimum discrimination of the classes of input vectors by means of a linear transformation. Minimizing the error of this linear discriminant requires that the input data undergo a non-linear transformation into the space, spanned by the activations of the hidden units, in such a way as to maximize a discriminant function.

The weights of the hidden layer of the MLP can be seen as intermediate processing units to extract relevant features for classification from the data. We may consider this as a non-linear map of the form

$$\vec{h} = \mathbf{M}(\vec{x}), \tag{5}$$

where $\vec{x} = x_1, \ldots, x_N$ is the vector of the inputs, N their number of attributes, and $\vec{h} = h_1, \ldots, h_{N_h}$ the vector containing the outputs of the N_h nodes of a MLP hidden layer.

This map may be very useful since it projects the input domain onto a lower dimensional space containing relevant features for classification.

The method proposed here, and label H-LVQ, is implemented in three steps. First, a specific MLP is previously trained for classification. Second, supervised LVQ is applied to the last hidden layer of the MLP to extract code vectors $\vec{w}_{ci}$ corresponding to each class c_i in which data are to be classified. Each example, $\vec{x}_i$, is classified as belonging to the class c_k with the smallest Euclidean distance to the respective code vector:

$$k = \min_j \|\vec{w}_{c_j} - \vec{h}(\vec{x})\|, \tag{6}$$

where $\|\cdot\|$ denotes the usual Euclidean distance. We should remark that by applying LVQ to the output of the last hidden layer of the MLP, and not the raw data itself, we are introducing a intermediate processing that should increase the discriminative capabilities of LVQ.

The third step consists of retraining the MLP with the backpropagation algorithm (see [12]), but with two important differences. First the error correction is applied not to the output layer but directly to the last hidden layer, ignoring from now on the output layer. The second difference is that the error applied is a function of the difference between $\vec{h}(\vec{x})$ and the code vector weights, $\vec{w}_{ck}$, of the respective class c_k to which the input pattern $\vec{x}$ belongs. Several expressions may be used, but we tested the following:

$$E_1 = \frac{1}{\beta}\sum_i (\vec{w}_{ck} - \vec{h}(\vec{x}_i))^{\beta}. \tag{7}$$

To reduce the contribution of outliers the coefficient β is set to small values (less than 2). Note that for $\beta = 2$ Eq. (7) is just the sum-of-squares error.

In some cases it is useful to include an extra repulsive term in the error function (3) to better separate the code vectors. Thus the error expression becomes

$$E_2 = \sum_i \left\{ \frac{1}{\beta}(\vec{w}_{ci} - \vec{h}(\vec{x}_i))^{\beta} - \gamma \sum_{c_j \neq c_i} \frac{\operatorname{sgn}(\vec{w}_{cj} - \vec{h}(\vec{x}_i))}{(\vec{w}_{cj} - \vec{h}(\vec{x}_i))^2} \right\}, \tag{8}$$

where the parameter γ should be small, compared to an unit. After training a new set of code vectors, $(\vec{w}_{ci})^{\text{new}} = \vec{w}_{ci} + \Delta\vec{w}_{ci}$, is obtained according to the following training scheme:

$$\begin{aligned} \Delta\vec{w}_{ci} &= \alpha(t)(h(\vec{x}) - \vec{w}_{c_i}) && \text{if } \vec{x} \in \text{class } c_i, \\ \Delta\vec{w}_{c_i} &= 0 && \text{if } \vec{x} \notin \text{class } c_i. \end{aligned} \tag{9}$$

Steps two and three are repeated following an iterative process. The process stops when a minimum classification error on the test set is found.

The H-LVQ is basically a new method to redefine the decision boundaries. Instead of defining a decision using a unique output, it may use all the information contained on the hidden nodes to achieve a better class discrimination.

This method has the same difficulties as other approaches concerning the choice of adequate learning parameters, namely setting the best neural net architecture, i.e., the number of hidden layers and the number of nodes on each layer. The correct number of nodes on the hidden layer depends on the complexity of the problem and cannot be determined a priori. It should be as small as possible as training of the MLP and the code vectors is more difficult for large N_h. On the other hand, too few nodes on the hidden layer may wash out important details and make impossible the separation of classes by the network.

Finally, we should note that this method could also be used to perform feature extraction on data. The nodes of the hidden layer with higher variance over all the training data can be seen as the most important to classify the data, and thus being considered as components of a feature vector. This is similar to principal component analysis but with some advantages. First, we can consider not only linear, but also non-linear combinations of the input components that have the greater variance, and second, the mapping is obtained through a supervised training. This has the obvious benefit that the features extracted from the input are coupled to their relevance of the determination of the outputs.

To understand why H-LVQ can be more efficient than traditional MLP we will consider the following example. Suppose we have two inputs, x_1 and x_2, differing only by a small quantity, $\delta x = x_2 - x_1$, to be classified into two classes by a MLP with a single output neuron y. To achieve a good class discrimination for the two inputs we should have very different outputs (one close to 0 and other close to 1). We can estimate the capability of the network to discriminate similar inputs by computing its sensitivity, S. Let us calculate the sensitivity of the output node, S_o:

$$S_o = \frac{\partial y}{\partial h}\frac{\partial h}{\partial x} = \sum_{i=1}^{N_h}\left(\frac{\partial y}{\partial h}\right)_i (S_h)_i \tag{10}$$

where h are the outputs of hidden layer, and S_h the sensitivity of the hidden layer with respect to the input. If we use the sigmoid function, g, as the activation function, we have

$$\left(\frac{\partial y}{\partial h}\right)_i = g'(w^i h^i)w^i = g(w^i h^i)(1 - g(w^i h^i))w^i, \tag{11}$$

where w^i are the weights connecting the hidden layer to the output. If x_1 belongs to class 1 and x_2 to class 2, the absolute value of the argument in the sigmoid should be large. Consequently its derivative should be small, as well as $\partial y/\partial h$. If this quantity is smaller than 1, we conclude that the sensitivity of the output layer is smaller than that of the last hidden layer: $S_o < S_h$. We conclude that, given a trained network, it is preferable to use the values of the last hidden layer for classification that directly the output node.

5. Results

We tested our method in several problems. First we applied it to our main task, classification of RBS spectra, as it was described above. Then we applied it to a benchmark problem of image recognition and to a problem of bankruptcy prediction.

5.1. Application of H-LVQ to RBS

H-LVQ was applied to the problem of classification of the RBS spectra into three categories, as mentioned in Section 2. We used the 50 nodes of the last MLP hidden

Table 2
Misclassification errors on the test set for the best MLP. The total misclassification error was 13.67%

NN \ real	Class 1	Class 2	Class 3
Class 1	—	0.43	1.75
Class 2	2.16	—	0.77
Class 3	0.89	7.37	—

Table 3
Misclassification errors with H-LVQ. The total misclassification was 2.59%

NN \ real	Class 1	Class 2	Class 3
Class 1	—	0.1	0
Class 2	0.63	—	0.23
Class 3	0.20	1.43	—

layer and selected the following learning parameters: $\alpha_0 = 0.5$, $\beta = 2$, and $\gamma = 0$. The results are presented in Tables 2 and 3.

After training with 100 epochs, we achieved a minimum error on the test set of 2.6%. This represents an improvement by more than five folds with respect to MLP. The major error (1.4%) occurs for class 2 being misclassified as class 3. The smaller error (0.0%) was obtained for class 3 being misclassified as class 1.

Next, we tested our method with the same parameters, except that we used a different number of hidden nodes $N_h = 40$, arriving at practically the same results. For $N_h = 30$ the test error in MLP was now noticeably higher (16.8%) and the performance of H-LVQ also deteriorated (3.6% misclassified).

We also applied H-LVQ on each of the two other hidden layers that compose the MLP network: the first containing 100 nodes and the second with 80 nodes. In both cases we found larger errors on the test set, particularly in the first hidden layer. This should be expected since the first layers are only capable of implementing a number of hyperplanes that split the feature space into disjoint open half-spaces, thus not being able to describe the interior of the regions it defines.

Fig. 3 presents the code vectors corresponding to the three classes after training. Notice that of the 50 components, all but about 8 had almost vanished. This means that out of the 134 inputs, only about 8 of its non-linear combinations are effectively necessary to separate linearly the decision regions.

Moreover, the final code vectors for the 3 classes are weakly correlated, which is very convenient to solve the problems of class membership overlapping. The capability of H-LVQ to obtain such well-separated code vectors should be one of the reasons for the good performance of this method.

5.2. Application to DNA helicases

This is a problem of electron microscopy image classification. The objective of the classification is to characterize the structure of a representative hexametric helicase: the

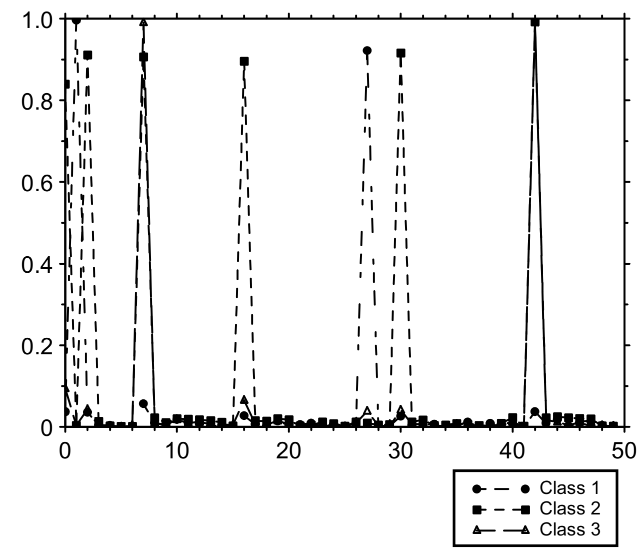

Fig. 3. Codebook vectors for the three classes after training. Note that only about 8 of the 50 nodes are used to distinguish the classes.

large T antigen of Simian Virus 40. When observed in the electron microscope, large T antigen preparations mainly show either a characteristic roughly circular view with a stain penetrating region in the center (which we will call a top view) or a view with a rectangular shape (side view). The training and test set for this problem are the same as used in Ref. [11]. This set consists of 197 examples, divided into the training set (78 examples), validating set (60) and testing set (59). Each example consists of 25 inputs (the 25 blocks of the image to be classified), and the output: 0 for side view and 1 for top view. The tests were carried out using our method.

The MLP architecture used for this problem has a single hidden layer containing 20 nodes. After training for 100 epochs, the number of cases in the test set misclassified was 6, that is a 10% error. Next we tested our method using the following learning parameters: $\alpha_0 = 0.5$, $\beta = 1.5$, $\gamma = 0.1$. After only 5 iterations, and in less than 1 s, the number of misclassified cases dropped to 2, which corresponds to an error of only 3.3%. Table 4 compares these results with that obtained by other authors [6]. We can see that H-LVQ outperforms other methods, including simulated annealing.

5.3. Application to bankruptcy prediction

Detecting when a company is going to fail is a difficult problem that requires a good knowledge of the company [1,8]. Although this is a difficult topic, it is of extreme importance for the company's shareholders. Traditionally carried out by accounting experts using heuristic rules, lately this problem has also been tackled by automatic methods, based on statistical and empirical analyses, or adaptive techniques such as neural networks [15].

Table 4
H-LVQ applied to the benchmark problem of DNA Helicases

Method	Misclassified cases (test set)
QP	5
SA	5
GP-Prop II	4
MLP	6
H-LVQ	2

QP designates quick propagation, SA simulated annealing, GP-prop genetic programming optimized neural network, MLP is the traditional multi layer preceptron.

In this case the sample consists of 450 non-financial companies, half of which boasted of good financial health and the other half had failed. This last group corresponds to those companies that had suspended payments or had declared legal bankruptcy, in accordance with Spanish Law. Healthy companies were randomly selected among a set of 150,000.

The dependent variable takes a value either 1, in the case of legal failure, or 0, for a healthy firm. We used 20 quantitative independent variables, as described in Ref. [18]. We trained an MLP containing a single hidden layer of 10 nodes, and a single output node, with the usual 1-of-*c* coding scheme. After 100 epochs, a minimum misclassification error on the test set of 21% (24 miss-classified cases out of 114) was reached.

H-LVQ was applied with learning parameters $\alpha_0 = 0.5$ and $\beta = 2$. After only 10 iterations, a minimum error on the test set of 14% was reached. This result should be compared with a minimum test error of 18%, achieved by a genetic optimized MLP consisting of an intermediate layer of 30 neurons.

5.4. Application to other problems

We used H-LVQ on other benchmark problem: predict the onset of diabetes on patients based on eight measured parameters [20]. The training data consists of 658 examples and cross validation were used. The test set for each run consists of 10% of the data. We used an MLP with a single hidden layer with 10 neurons, and after 100 epochs the average train set error was 24%. We then trained the MLP with H-LVQ using the following set of training parameters $\alpha_0 = 0.5$, $\beta = 1.3$, $\gamma = 0$. The minimum error found was 26%, thus a worse result. We tested different architectures with two hidden layers without gaining any improvement.

These poor results may be due to the fact that code vectors are very similar—their internal product is 0.12. The internal product of the prototypes of a two class problem may be used as a measure to quantify the class separability. When this value is small the separation is hard.

6. Conclusions

We presented a new algorithm for training MLP on classification problems, that we designated H-LVQ. It proved to be very efficient not only for classification of RBS spectra, but also on other difficult problems, although in some cases we found no advantages over traditional MLP.

We conclude that the method presented is more indicated for problems with a large number of correlated inputs requiring significant pre-processing transformations where significant gains over other methods can be obtained.

Acknowledgements

We would like to thank P. Castillo and the Geneura group, from University of Granada for sharing the data and results of DNA helicases and bankruptcy prediction. The support of the International Atomic Energy Agency under Research Contract No. 11317/R1/Regular Budget Fund is gratefully acknowledged.

References

[1] E.I. Altman, J. Banking Account. Finance 8 (1984) 171.
[2] N.P. Barradas, C. Jeynes, R. Webb, Appl. Phys. Lett. 71 (1997) 291.
[3] N.P. Barradas, A. Vieira, Phys. Rev. E 62 (2000) 5818.
[4] N.P. Barradas, A. Vieira, E. Alves, Nucl. Instrum. Methods B 108 (2001) 175–177.
[5] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1997.
[6] P.A. Castillo, J. González, J.J. Merelo, A. Prieto, V. Rivas, G. Romero, Neurocomputing 35 (2000) 149.
[7] P.S. Churchland, T.J. Sejnowski, The Computational Brain, MIT Press, Cambridge, MA, 1992.
[8] A.I. Dimitras, S.H. Zanaki, C. Zopounidis, European J. Oper. Res. 90 (1996) 487.
[9] T. Kohonen, Self-Organization and Associative Memory, Springer, Berlin, 1989.
[10] B. Kosko, Neural Networks for Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1992.
[11] J.J. Merelo, A. Prieto, F. Morán, R. Marabini, J.M. Carazo, Neural Process. Lett. 8 (1998) 55.
[12] D.E. Rumelharb, R. Durbin, R. Golden, Y. Chauvin, in: D.E. Rumelharb, Y. Chauvin (Eds.), Backpropagation: Theory, Architectures, and Applications, Vol. 1, Lawrence Erlbaum, Hillsdale, NJ, 1995.
[13] D.E. Rumelharb, G.E. Hinton, R.J. Williams, in: D.E. Rumelharb, J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, Cambridge, MA, 1986, p. 318.
[14] S.S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall, Englewood Cliffs, NJ, 1994.
[15] K.Y. Tam, M.Y. Kiang, Management Sci. 38 (7) (1992) 926.
[16] J.R. Tesmer, M. Nastasi (Eds.), Handbook of Modern Ion Beam Materials Analysis, MRS, Pittsburgh, 1995.
[17] A. Vieira, N.P. Barradas, Nucl. Instrum. Methods B 170 (2000) 235.
[18] P.A. Castillo, J.M. de La Torre, J.J. Merelo, I. Roman, in: 24th Annual Congress European Accounting Association, Athens, April 2001, p. 182.
[19] H. White, Neural Comput. 1 (1989) 425.
[20] http://www.kdnuggets.com/datasets/index.html.