Automatic classification of biological particles from electron-microscopy images using conventional and genetic-algorithm optimized learning vector quantization

J. J. Merelo and A. Prieto
Depto. Electrónica y Tecnología de las Computadoras, Facultad de Ciencias, Campus Fuentenueva, s/n, 18071 Granada (Spain)
E-mail: jmerelo@kal-el.ugr.es, http://kal-el.ugr.es/

F. Morán
Depto. de Bioquímica y Biología Molecular I Facultad de Químicas Universidad Complutense de Madrid,
E-mail: fmoran@solea.quim.ucm.es
28040 Madrid (Spain)

R. Marabini and J. M. Carazo
Grupo de Biocomputación, CNB
E-mail: {carazo|roberto}@cnb.uam.es
28040 Madrid (Spain)

(Received ..... ; Accepted in final form .....)

Abstract. Automatic classification of transmission electron-microscopy images is an important step in the complex task of determining the structure of biologial macromolecules. The process of 3D reconstruction from a set of such images implies their previous classification into homogeneous image classes. In general, different classes may represent either distinct biochemical specimens or from different directions of an otherwise homogenous specimen. In this paper a neural network classification algorithm has been applied to a real-data case in which it was known a priori the existence of two differentiated views of the same specimen. Using two labeled sets as a reference, the parameters and architecture of the classifier were optimized using a genetic algorithm. The global automatic process of training and optimization is implemented using the previously described G-LVQ (genetic learning vector quantization) (1) algorithm, and compared to a non-optimized version of the algorithm, Kohonen's LVQ (learning vector quantization) (2). Using a part of the sample as training set, the results presented here show an efficient (approximately 90%) average classification rate of unknown samples in two classes. The implication of this kind of automatic classification algorithms in the determination of three dimensional structure of biological particles is finally discused. This paper extends the results already presented in (3), showing also some improvement over them.

Key words: Genetic algorithms, neural networks, neural network optimization, image classification, image reconstruction

This paper has not been submitted elsewhere in identical or similar form, nor will it be during the first three months after its submission to Neural Processing Letters.

1. Introduction

DNA helicases are ubiquitous enzymes with fundamental roles in all aspects of nucleic acid metabolism: DNA replication, repair, recombination, and conjugation. Their activity leads to disruption of the hydrogen bonds between the two strands of duplex DNA. Presumably all species in nature contain a collection of helicases that participate in many, if not all, facets of DNA metabolism. In spite of their critical role, and of the amount of biochemical knowledge in the field, little is known about their structure. We are making use of electron microscopy and image processing techniques to study the structure of one representative hexameric helicase: the large T antigen of Simian Virus 40 (4).

When observed in the electron microscope, large T antigen preparations show mainly either a characteristic roughly circular view with a stain penetrating region in the centre (which we will call a *top view* in what follows) or a view with a rectangular shape (*side view*) (see Figure 3.

In general, we are interested in two kinds of final image processing approaches. The first one is an increase of the signal-to-noise ratio of the images by a process of averaging. The second one is to use the views of an specimen coming from different directions as input to a 3D reconstruction algorithm whose mathematical principles are very similar to the ones behind the familiar "Computerized Axial Tomography" in Medicine. Our biological goal is set to reach high resolution (subnanometer resolution), and to this end it is neccesary to have very significant image statistics, forcing us to process thousands of images. However, it is obvious that prior to any of these processes it is vital to be able to separate and classify these images into their main views in a fast, reliable, and as authomatic and objective way as possible.

Our strategic aim would then be to provide a reliable method to classify large quanties of images -in the order of thousands or ten of thousands- in an automatic manner as a way to obtain high resolution structural results in spite of the low signal to noise ratio of the individual images.

Neural network techniques have already been applied succesfully to biological particle classification and reconstruction (5; 6), showing results that are more robust, and sometimes faster, than traditional statistical techniques. In this paper, we will apply to the above mentioned problem a previously described technique, G-LVQ (1), based in a two-level genetic algoritm (GA) operating on variable size chromosomes, which codify the initial weights and labels for an LVQ network; results will be compared to Kohonen's Learning Vector Quantization (2) (LVQ) algorithm for codebook training. We will first present the state of the art in evolutionary classifier design in section 2. G-LVQ algorithm, used in this paper to classify helicases, is briefly described in section 3, to be followed by results obtained in section 4 and a brief discussion (section 5).

2. State of the art

The technique used in this paper, G-LVQ, is an optimized version of Kohonen's LVQ, a supervised codebook design technique, and at the same time one of the most widely used neural techniques (second only to MLP+backprop neural nets).

Despite the vast amount of literature already produced on global optimization of neural networks and other classifiers using genetic algorithms, parameter tuning "by hand" seems to be yet the state of the art, for instance in backpropagation (7) (BP), and Kohonen's Self-Organizing Map (8).

The task of finding the correct values of the parameters in a classifier, that is, what is known in the neural network world as "training the weights of a neural net", is called by statisticians parameters estimation; and they face a similar problem: finding a combination of parameters which give optimal accuracy results, and maximum parsimony, that is, minimum amount of parameters. Some global optimization procedures have been tested: simulated annealing and stochastic approximation, but little mention is made in the literature to GAs, except for optimizing k-nearest neighbor (9; 10), an unsupervised clustering algorithm. For this kind of problem, Markov chain Monte Carlo (11) and bootstrap algorithms (12) are used to obtain the number of clusters, but these algorithms are rather costly in terms of computation time.

The usual approach to classifier optimization is to optimize one of the three metrics for classifier performance, like classification accuracy for supervised learning, or distortion for unsupervised learning, while

leaving size fixed. Several size values are usually tested, and the best is chosen.

Some other methods for optimizing LVQ have been based on incremental approaches (for a review, see (13)), which are still local error-gradient descent search algorithms on the space of networks or dictionnaries with different size. Other methods use genetic algorithms (14) to set the initial weights of a codebook with a maximum implicit weight; this maximum length limits the search space, which might be a problem to find the correct codebook, whose number of levels can be higher.

An incremental methodology proposed by Perez and Vidal (15), seems to offer the best results for this kind of methodology. This method adds or takes codevectors after presentation of the whole training sample, eliminating those that have not been near any vector in the training sample, and adding as new codevector a vector from the training sample that has been incorrectly classified, and is the most distant from all codevectors belonging to its same class. This approach has the advantage of not relying on threshold parameters, but it still has the problems of being a local search procedure that optimizes size step by step and of relying on heuristics for the initial weights.

A method for global optimization of LVQ was proposed by the authors in (1). That method has been simplified in this and previous papers (3), taking out unnecesary features.

With respect to the problem of classifying biological particles, usual pattern recognition techniques are a mixture of neural networks and personal experience, with some statistical techniques thrown in; particles are classified in an unsupervised way using principal component analysis or Kohonen's self-organizing map, and then labeled by hand (6). This method has been used to label the samples used in this paper, which means that its labelling is inherently unaccurate. In any case, usual statistical techniques can be used to assess the accuracy of the automatic classification and the previous manual classification. Obtaining a neural network for automatic classification that objectivizes the expert knowledge without needing to make it explicit would also be an achievement.

3. Method

The method used here for obtaining optimal biological particle classifiers is based on several pillars, which sometimes lean on each other: hybridization of local and global search operators, variable length chromosome genetic algorithm and vectorial fitness. The genetic algorithm

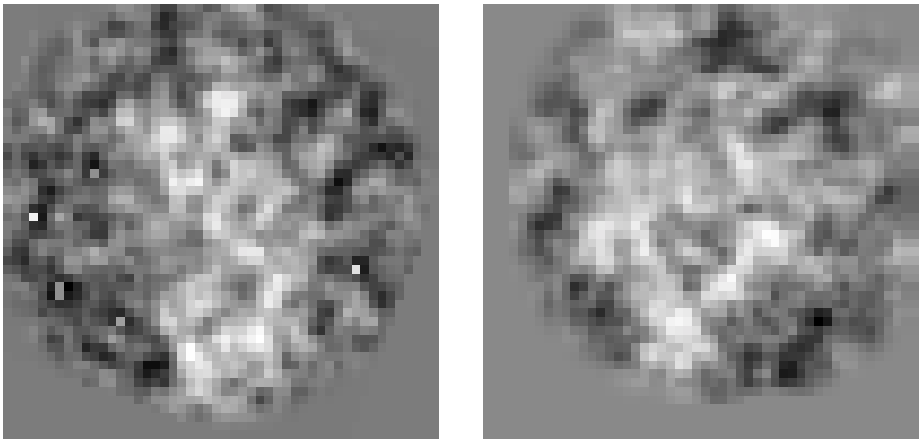


Figure 1. Two representative images from the training set; in this case it corresponds to a top (left) and side view (right) of a DNA helicase.

used in this work, based in the previous considerations, can be described in the following way:

1. Initialize randomly a chromosome population with lengths ranging from half the number of classes to twice the number of classes.
2. For each member of the population:
   a) Using the values stored in the chromosome as initial values for classifier parameters, create a classifier and train it using the classifier algorithm (a local search method), which has been selected in advance. Then evaluate accuracy and distortion in classification using the training set as test set. Set vectorial fitness to the triplet (accuracy, size, distortion).
   b) With a certain probability, apply "directed" chromosome length changing operators, suppresing or adding new significant structures to the initial values of the classifier depending on the exit achieved.
3. Select the P best chromosomes in the population, according to their fitness vector, and make them reproduce, with mutation and 2-point crossover (acting over the length of the smallest chromosome of the couple). Apply, if chosen, nondirected duplication and elimination operators.
4. Finish after a predetermined number of generations.

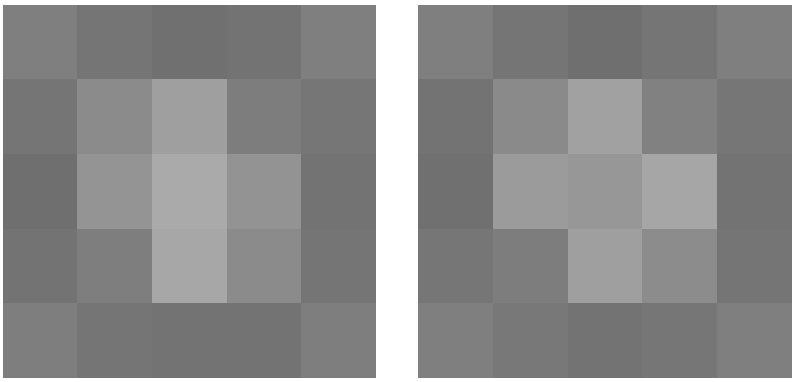


Figure 2. Average images for type 1 (side view) and type 2 (top view) after averaging 10x10 pixel blocks.

4. Results

To assess the accuracy of the method, there were 1817 samples available, obtained after a process of segmentation of the electron-microscopy images of the helicase T antigen of SV40 (4). This set has been classified by hand, yielding 741 samples of type 1 (which corresponds to particle side view) and 1076 of type 2 (top views). Each sample is a 50x50 pixel 8-bit gray-level image, as shown in figure 3.

Each particle is then represented by a 2500-component vector, which is usually too much for a neural network. The *curse of dimensionality* states that the difficulty of training a neural network (or estimating parameters with any other procedure, for that matter) increases exponentially with the number of dimensions. In this case, it implies that some preprocessing must be done on the image before feeding it to the neural network. The simplest one that can be done is averaging the image by pixel blocks. The result of averaging all images by 10x10 pixel blocks and computing the average image for each class still presents substantial difference, as is shown in figure 4, which leads to think that this preprocessing still keeps much of the information that makes classes different. Of course, this a priori estimation must be cross-checked with final results.

Using these preprocessed images, three files were created, one for training, another one for testing, and another one for validation. Training files were used to train al neural nets, test files to select one among all trained nets, and validation file to check accuracy for that network; validation file has never been shown before to the chosen network. Each

file contained 198 samples, half from each class. Only the training and validation files were used for the LVQ algorithm, since instead of selecting one net, the average and lowest errors are taken.

It must be taken into account that classes need not be well classified in advance: this automatic classification procedure could serve to confirm or dismiss manual classification; that is why, in principle, high values of error should be expected.

Two classification algorithms were then tested on these sets: Kohonen's classical Learning Vector Quantization, or LVQ, and G-LVQ (1), which is basically, as pointed out above, a genetic optimization of Kohonen's LVQ that, at the same time, discovers the optimal size of the LVQ network. Other algorithms, like backpropagation, were not tested here since previous work on similar problems yielded similar accuracy, but a higher number of parameters to fit (that is, more weights). Usually LVQ is not a good algorithm when the number of classes is small, being usually beaten by Backprop. However, it is usually a high-speed and compact alternative to it, generating classifiers with less parameters which can be trained faster.

LVQ was run 500 times with several preset codebook sizes, using the training file to train and the validation file as test file; since no further selection is made on the training parameters, the test file was not considered necessary. LVQ algorithm has no method to select the codebook size in advance, thus, three different sizes were tested: 2, 4 and 8 levels. Weight vectors and labels were initialized in two different ways:

- random initialization with vectors from the training file, which usually gives good results, but with so few codevectors, it might happen that many codevectors have got the same label.
- or pure random initialization with values in the same range as the training file, and sequential label initialization. I this way, equal probability for all classes is insured.

Only the training and validation files were used;. The gain factor α was set to 0.1, decreasing by α/epochs each step; and they were trained for 1000 epochs. The program was written using S-VQ, (Simple Vector Quantization), a C++ class library for programming vector quantization applications, which is available from the author. The whole test took several minutes on an Silicon Graphics O_2.

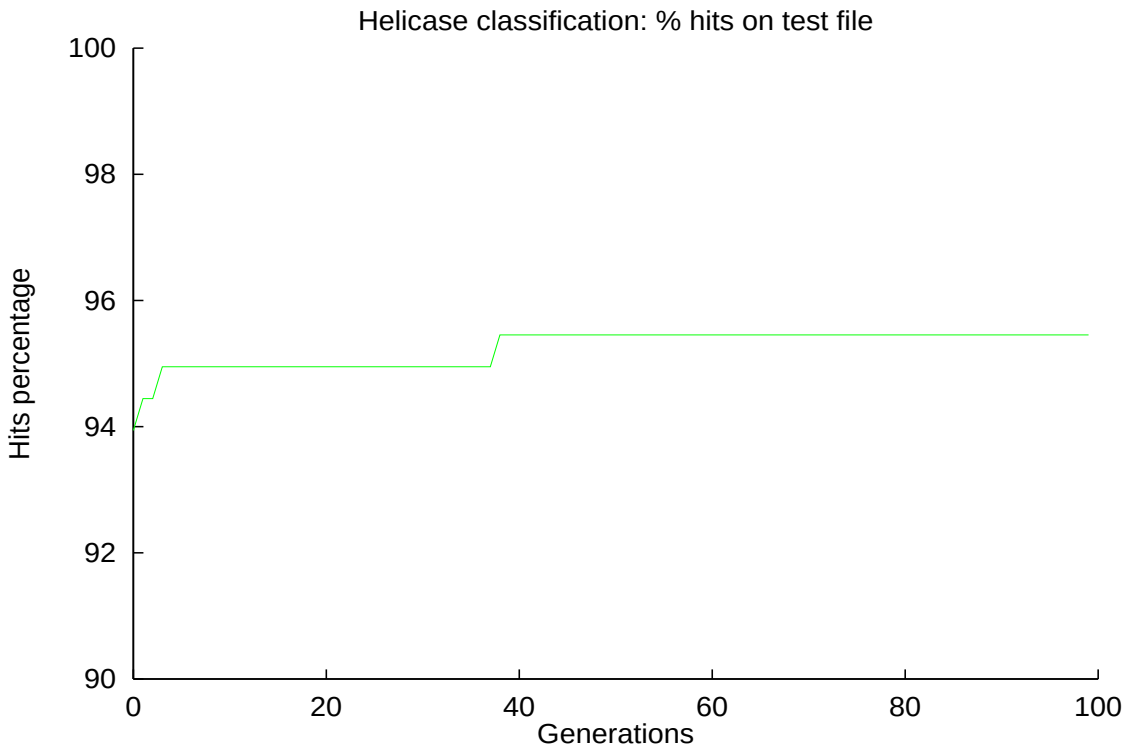
The genetic algorithm used for G-LVQ is a steady-state algorithm, which means that a part of the population is kept in each generation. In this case, 40% of the population were substituted each generation, with the offspring of the 40% best. Population was fixed to 100 networks for each generation. Variable-length chromosome operators were

Table I. Results of evaluating LVQ and G-LVQ on the classification of different views of large T antigen of Simian Virus 40. Parameters for LVQ were as follows: 1000 training steps, and gain parameter set to 0.1; random vector initialization with ordered labels (1) or from the training file (2). In the case of G-LVQ, each network received 1000 training steps.

Algorithm		Error+StdDev	Lowest	Net size
LVQ(1)	2-level	21 ± 15	6.57	2
	4-level	15 ± 5	7.07	4
	8-level	14 ± 5	6.57	8
LVQ (2)	4-level	28 ± 17	8.6	4
	8-level	16 ± 5	9.1	8
G-LVQ	20 generations	18 ± 2	7.58	2.3 ± 0.4
	50 generations	13 ± 3	8.18	2.2 ± 0.3
	100 generations	15 ± 5	9.09	2.2 ± 0.3

used, with 1% probability for the gene-duplication operator, and 0.5 % probability for the gene-elimination operator. Bit-flip mutation and 2-point crossover were applied with 10% probability. These rates did not change during execution. The GA was run for several generations, using a vectorial fitness as shown in (1); the main criterium for optimization was minimization of misclassification rate, followed by length. Accuracy and length evolution during a tipical run are shown in figure 4; usually each of those quantities is optimized in turn; first accuracy is optimized, until a good codebook is found; then, while keeping the same accuracy, G-LVQ optimizes length. This is not set in advance, and is only due to the order defined on the fitness triplets. In this case, in generation 40, a codebook with more codevectors but better accuracy is found; later on, keeping accuracy, a trimmed-down version of the same codebook is found, and selected as the winner. G-LVQ was run 20 times with the same parameter settings, and different random initialization. Running the test took less than an hour on an Silicon Graphics O_2.

Results for different parameter settings and initializations are shown in table 4. The best average results for the LVQ algorithm are for the biggest codebooks tested, 8 levels, and random initialization. However, with many random initializations, a small error, around 7%, can be expected from this algorithm. Initializing randomly from the training sample yields worse results since the labels are not assigned as the probability distribution function would advise.



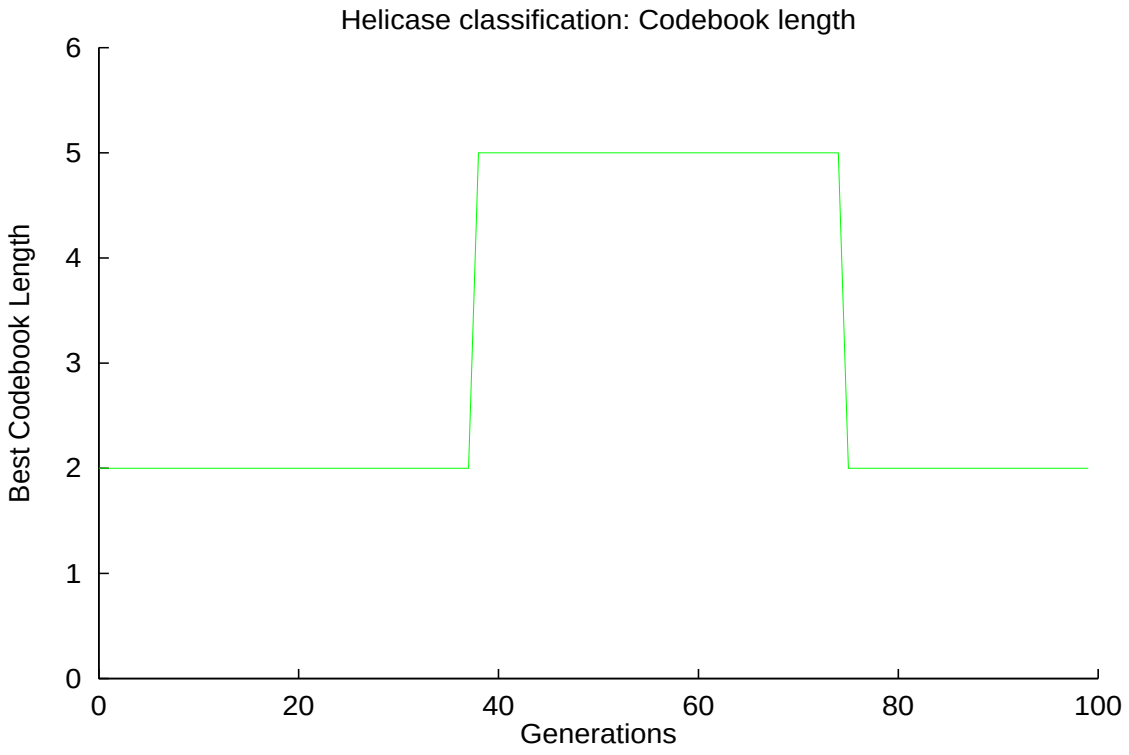


Figure 3. Evolution of the accuracy of G-LVQ in terms of hits percentage and code length during a typical run. At around generation 40, a codebook with better accuracy but more codevectors is found; later on during the simulation a codebook with the same accuracy but smaller length takes its place.

As should be expected, G-LVQ outperforms LVQ, and besides, finds codebooks that are much more compact. Usually found codebooks have only 2 codevectors 4; sometimes 3 or even 5. Among them, an intermediate number of generations (50) yields the best results. This is probably due to generalization problems: training for too many generations overfits the codebooks to the training and test sets, which increases error rate on the validation set.

In the best G-LVQ case, average error is 1 point lower than for the best LVQ case, and standard deviation is smaller too. However, the lowest error found is usually higher than for LVQ. This is probably due to the lack of a generalization check using a test file for LVQ; the former algorithm is only checked with the validation file, and not the test file. And, in any case, in these experiments, we were looking for best average results, not absolute best, since average results give an estimation of the expected accuracy when the algorithms are run a few times, not hundred of times. With respect to this, it can be affirmed that G-LVQ outperforms LVQ if only one or a few initializations are performed.

In any case, G-LVQ achieves the lowest error with codebooks that are much smaller than LVQ, and besides, assigns labels automatically, not needing a heuristic procedure to set them.

Codevectors in the best codebook found give an idea of how a top and side view should look like (after preprocessing, of course), as is shown in figure 4; in this case, it confirms that a low-density center is required to identify an image as a side view, while top views show a center with higher density than its surroundings. These codevectors would represent a kind of balanced or consensus image of each class.

In any case, error is quite high, and 1 out of 10 samples are usually misclassified. This could be due to the following reasons:

- samples are not well labeled, and
- preprocessing applied obviously loses some information.

Another observed that during tests was that training error was quite different to validation error, being this much higher, which might confirm the erroneous labeling of some vectors in the training and/or test samples.

5. Discussion and future work

This paper shows very promising results in the task of automatic classification of electron microscopy imagest of biological particles.

So far, this problem had not been approached using neural classification algorithms, so that it is difficult to offer a comparison with

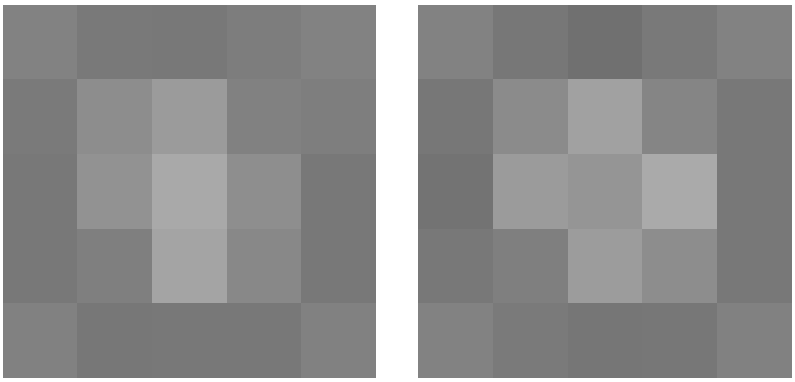


Figure 4. Trained codebook obtained in one of the G-LVQ runs; first one is labeled as a side view, and the second one as a top view. The genetic algorithm has assigned 1 vector to each class. This particular run obtained a 9% error on the validation sample. The main difference seems to be in the density of the center of the image, which is lower for side views.

other approaches; that is the reason why two neural algorithms have been compared. Besides, initial tests with other neural algorithms, like backpropagation, give a much higher classification error, even with the training file.

Original images had to be preprocessed to obtain vectors with less dimensions; at the same time, this preprocessing was checked to prove that not all information that made classes different was lost.

Results show that a 8-level LVQ codebook, initialized randomly with equal number of labels assigned to classes, gives the best results, achieving an average 14% error, and 6.57% best-case error on a training file/test file setup.

These results are improved by using a genetic optimization of the LVQ codebook training algorithm, called G-LVQ by the authors. In this case, the best results are achieved with an intermediate number of generations, namely 50 generations, and steady state selection, giving a 13% average error and 8on a training/test/validation file setup. This proves again that, without being much slower (an order of magnitude or so), G-LVQ achieves much better results than LVQ in the accuracy and parsimony sense. At the same time, G-LVQ is a fully automatic codebook-design procedure, which does not need heuristic initialization procedures for the weights, labels or size.

The results achieved so far are very relevant to the biological application presented in this work, in that it opens new possibilities by allow-

ing us to automatically process tens of thousands of images with a high sucess rate. G-LVQ is a fast and accurate way of generating classifiers for any set of images, and, besides, does not need much parameter-tuning by hand, if any.

These results can be improved in several ways; first, with a more careful relabelling, and second, by testing and cross-checking several preprocessing methods. Future work will go along this line, and will include jackknife procedures for assessing the capabilities of these and other algorithms for automatic classification. Besides, other preprocessing procedures more adapted to the problem can also be tested.

Acknowledgements

This work has been supported in part by CICYT (Spain) grants number BIO-95-0768 and BIO96-0895.

References

J. J. Merelo and A. Prieto. G-LVQ, a combination of genetic algorithms and lvq. In N.C.Steele D.W.Pearson and R.F.Albrecht, editors, *Artificial neural Nets and Genetic Algorithms*, pages 92–95. Springer-Verlag, 1995.

T. Kohonen. The self-organizing map. *Procs. IEEE*, 78:1464 ff., 1990.

J. J. Merelo, A. Prieto, and F. Morán. A GA-optimized neural network for classification of biological particles from electron-microscopy images. In Cabestany Prieto, Mira, editor, *Proceedings IWANN 97*, number 1240 in LNCS. Springer-Verlag, 1997.

C. San Martin, C. Gruss, and J.M. Carazo. Six molecules of sv40 large T antigen assemble in a propeller-shaped particle around a channel. *Journal of Molecular biology*, page in press, 1997.

Jose-Jesus Fernández and Jose-Maria Carazo. Analysis of structural variability within two-dimensional biological crystals by a combination of patch averaging techniques and self-organizing maps. *Ultramicroscopy*, 65:81–93, 1996.

R. Marabini and J.M. Carazo. Pattern recognition and classification of images of biological macromolecules using artificial neural networks. *Biophysics Journal*, 66:1804–1814, 1994.

W. Schiffman, M. Joost, and R. Werner. Optimization of the backpropagation algorithm for training multilayer perceptrons. Technical report, University of Koblenz, Institute of Physics, 1994.

F. Murtagh and M. Hernández Pajares. The Kohonen self-organizing map method: An assesment. Technical report, European Space Agency/ UPC, 1993.

R. Huang. Systems control with the genetic algorithm and the nearest neighbour classification. *CC-AI*, 9(2-3):225–236, 1992.

Jr. James D. Kelly and Lawrence Davis. Hybridizing the genetic aglorithm and the k nearest neighbors classification algorithms. In Richard K. Belew and Lashon B. Booker, editors, *ICGA91*, San Mateo, CA, 1991. Morgan Kaufmann.

S. P. Brooks. Markov chain monte carlo and its application. *The Statistician*, 1998.

Bradley Efron and Robert J. Tibshirani. *An introduction to the bootstrap*. Chapman & Hall, 1993.

Ethem Alpaydim. GAL: Networks that grow when they learn and shrink when they forget. Technical Report TR-91-032, International Computer Science Institute, May 1991.

Enrique Monte, D. Hidalgo, J. Mariño, and I. Hernáez. A vector quantization algorithm based on genetic algorithms and LVQ. In *NATO-ASI Bubión*, page 231 ff., 1993.

Juan Carlos Pérez and Enrique Vida. Constructive design of LVQ and DSM classifiers. In J. Mira, J. Cabestany, and A. Prieto, editors, *New Trends in Neural Computation*, pages 334–339, 1993.

Address for correspondence: J. J. Merelo, E-mail: `jmerelo@kal-el.ugr.es`
Depto. Electrónica y Tecnología de las
Computadoras, Facultad de Ciencias,
Campus Fuentenueva, s/n,
18071 Granada (Spain)