Computing and Informatics, Vol. 26, 2007, 33–43

EVOLVING GENERALIZED EUCLIDEAN DISTANCES FOR TRAINING RBNN

José M. Valls, Ricardo Aler, Oscar Fernández

Avenida de la Universidad
30. 28911 Leganés (Madrid), Spain
e-mail: jvalls@inf.uc3m.es, ricardo.aler@uc3m.es

Manuscript received 23 March 2006; revised 5 September 2006
Communicated by Vladimír Kvasnička

Abstract. In Radial Basis Neural Networks (RBNN), the activation of each neuron depends on the Euclidean distance between a pattern and the neuron center. Such a symmetrical activation assumes that all attributes are equally relevant, which might not be true. Non-symmetrical distances like Mahalanobis can be used. However, this distance is computed directly from the data covariance matrix and therefore the accuracy of the learning algorithm is not taken into account. In this paper, we propose to use a Genetic Algorithm to search for a generalized Euclidean distance matrix, that minimizes the error produced by a RBNN.

Keywords: Generalized distances, evolving distances, radial basis neural networks, genetic algorithms

1 INTRODUCTION

Radial Basis Neural Networks (RBNN) [1, 2] are originated from the use of radial basis functions, in the solution of the real multivariate interpolation problem [3, 4]. As the Multilayer Perceptron (MLP) they can approximate any regular function [5]. Due to its local behavior and to the linear nature of its output layer, their training is faster than MLP training [5] and this fact makes them useful for a wide variety of applications. The most used radial basis functions are Gaussian functions, defined by Equation (1).

$$\phi_m(x_k) = e^{-\frac{\|\mathbf{c_m}-\mathbf{x_k}\|^2}{2\sigma_m^2}} \tag{1}$$

where $\phi_m(x_k)$ represents the activation function for neuron m when an input pattern x_k is presented. The vector $\mathbf{c_m}$ is the center of the neuron m, and σ_m is its deviation or width.

One of the problems of RBNN is the symmetrical nature of their activation function, making that the activation of a neuron when a pattern is presented, only depends on the Euclidean distance from this pattern to the neuron center. This implies that all attributes have the same relevance for the learning task. This could be solved by altering the metric used in the activation function, so that, for instance, differences between values of relevant attributes are given more importance than less relevant ones. The use of non-Euclidean metrics in the context of RBNN was suggested by [6, 7, 8].

The Mahalanobis distance addresses this problem. It is a metric used in statistics in order to normalize different attributes and take into account the correlations among them. This distance is computed according to Equation (2).

$$d_{ij} = \left[(\mathbf{x_i}-\mathbf{x_j})^T\mathbf{S}^{-1}(\mathbf{x_i}-\mathbf{x_j})\right]^{1/2} = \left[(\mathbf{x_i}-\mathbf{x_j})^T\mathbf{M}^T\mathbf{M}(\mathbf{x_i}-\mathbf{x_j})\right]^{1/2} \tag{2}$$

where d_{ij} is the Mahalanobis distance between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, S is the variance-covariance matrix of all vectors in the data set and $\mathbf{M}$ is the so-called Mahalanobis matrix [9, 10, 11]. This distance has been used to improve prediction accuracy in learning systems that use distances [12]. However, the Mahalanobis distance is independent of the learning system used and of the error produced on the training data, because it is computed from the points in the dataset only (more specifically, it is computed from the variance-covariance matrix of the dataset).

To see this more clearly, Equation (3) displays the Mahalanobis distance between two vectors (x_1, y_1) and (x_2, y_2) in two-dimensional space.

$$d_{12}^2 = \frac{1}{(1-r^2)}\left[\frac{(x_1-x_2)^2}{\sigma_1^2} + \frac{(y_1-y_2)^2}{\sigma_2^2} - 2r\frac{(x_1-x_2)(y_1-y_2)}{\sigma_2\sigma_2}\right] \tag{3}$$

where r is Pearson's linear correlation coefficient and σ_1 and σ_2 are the respective standard deviations. In the simplest case where $r = 0$ (no correlation between the attributes), the Mahalanobis distance simply normalizes attributes x and y by dividing by their respective variance. Now, let us suppose that attribute x is highly correlated with the class to be predicted (i.e. very relevant for learning) whereas attribute y is not. This implies that attribute x should have a stronger weight in the distance function. But, clearly, the Mahalanobis distance does not take this into account, because neither the class nor the error are considered in the Mahalanobis matrix.

[13] used Euclidean weighted norms (diagonal matrices) that were computed during the training process. That is, unlike the Mahalanobis distance, the training error was considered for computing the weights. Similar techniques were reviewed

for other lazy techniques in [14]. However, only diagonal matrices (attribute weighting) were used. [15] presents some preliminary work about using full generalized Euclidean distances, which involve symmetrical matrices (non-diagonal matrices).

In this paper we extend previous work. We address the problem of finding the generalized Euclidean distance that minimizes the error of a RBNN for classification and regression tasks. We propose to use a generalized Euclidean distance in the activation function of the RBNN, so that, for instance, different attributes are treated differently according to their relevance. This will be achieved by a genetic algorithm [16] whose individuals are generalized Euclidean distance matrices and whose fitness function depends on the prediction accuracy attained by the network using the matrix.

2 DESCRIPTION OF THE METHOD

In this paper we use a standard Genetic Algorithm (GA) [16] to evolve distance matrices. A GA is a kind of heuristic search. The algorithm maintains a set of candidate solutions (or population of individuals) and applies the search operators on them (also called genetic operators: mutation and crossover). The search is guided by a heuristic (or fitness) function. We have used a standard generational GA with elitism and tournament selection. Matrices in the individuals are coded by representing each of their components in binary format. The fitness function is computed by training a RBNN on a set of training data and determining the training error. Training a RBNN involves to determine the centers, the widths, and the weights. We have chosen the usual hybrid approach [1]. Thus, the centers are calculated in an unsupervised way using the K-means algorithm, which is randomly initialized, to classify the input space. The neurons widths are obtained as the geometric mean of the distances from each neuron center to its two nearest centers. Finally, the weights are estimated in a supervised way to minimize the mean square error measured over the training set.

Thus, the GA tries to find the distance matrix that minimizes the RBNN training error. The number of hidden neurons is fixed from the start.

In particular, we wish to evolve matrix M to compute a generalized Euclidean distance between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, according to Equation (4).

$$d_{ij} = [(\mathbf{x_i} - \mathbf{x_j})^T \mathbf{M}^T \mathbf{M} (\mathbf{x_i} - \mathbf{x_j})]^{1/2}. \tag{4}$$

In order to determine the appropriate $\mathbf{M}$ matrix by using GA, individuals must be properly encoded. In order to reduce the number of components of $\mathbf{M}$ to be evolved, we have chosen matrix $\mathbf{M}$ to be either diagonal or symmetrical. In that case, only the diagonal and the upper half of the matrix coefficients must be encoded to a binary representation in order to build the chromosome of the individual. The diagonal case corresponds to a diagonally weighted Euclidean distance, and it is equivalent to have every attribute weighted by a factor (see Equation (5)). The second case (the symmetrical matrix) is a fully weighted Euclidean distance.

$$d(A,B) = \sqrt{(\sum_{i=1}^{i=d} m_{ii} * (A_i - B_i)^2)} \tag{5}$$

Each matrix element is a real number that must be encoded to a binary representation with a fixed number of bits, following a fixed-point representation with a single bit for the sign. Hence, the chromosome is a string of bits formed by the binary representation of each matrix element belonging to the diagonal or the upper half of the matrix; i.e., if the matrix is

$$\mathbf{M} = \begin{pmatrix} \mathbf{m}_{11} & \mathbf{m}_{12} & \dots & \mathbf{m}_{1d} \\ m_{21} & \mathbf{m}_{22} & \dots & \mathbf{m}_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ m_{d1} & m_{d2} & \dots & \mathbf{m}_{dd} \end{pmatrix}$$

the corresponding string chromosome will be

$$\{B(m_{11}), B(m_{12}), \dots, B(m_{1d}), B(m_{22}), B(m_{23}), \dots, B(m_{dd})\}$$

where $B(m_{ij})$ is the binary representation of m_{ij}

Each individual represents a matrix $\mathbf{M}$ that will determine the distance function to be used for the neurons activation (see Equation (2)). The goal of this work consists of improving the accuracy or the RBNN; hence, if the network error is small it means that the corresponding distance function is good; thus, the individual representing of the M matrix must have a big fitness value. The fitness function chosen in this work is given in Equation (6).

$$fitness = -\log_2 E \tag{6}$$

where E is the mean squared error committed by the network on the training data. It has been chosen so that fitness increases when error decreases. This function also manages to amplify differences between individuals whose error is close to zero. This is important to increase evolutionary pressure in the latest stages of GA-evolution, when all individuals are very good.

In the following, the sequential structure of the proposed method is summarized.

---

1. Create the initial population. A set of random chromosomes is generated. These chromosomes represent different distance functions to be used in the Radial Basis Functions of the networks.
2. Evaluate the fitness of each element of the current population. In order to perform this point, RBNN with a fixed number of hidden neurons are trained using the distance function determined by each individual of the population. Training errors of these networks are used to calculate the fitness of each individual.
3. Apply genetic operators to the population in order to create the next generation.

4. If the number of generations is lower than the maximum, go to step 2.
5. Return the highest fitness matrix.

3 EMPIRICAL RESULTS

The purpose of this section is to validate empirically our approach. Four sets of experiments will be carried out. First, a synthetic domain, where the solution is known, will be posed to the system. Next, three more domains will be tested: the Mackey-Glass and the Venice Lagoon time series, and a classification problem, the Ripley data set.

3.1 Synthetic Domain

This domain follows a bi-variate Gaussian shape ($\mu = (0.5, 0.5)$, $\sigma^2 = 0.002$). However, instead of the Euclidean distance, a generalized Euclidean distance with a symmetrical matrix $\mathbf{M}$ will be used instead (see Equation (2) and matrix (7)). In Euclidean space, the result is a rotated and stretched gaussian (i.e. non-symmetrical). In short, the goal is to approximate the function given by Equation (8), where $\mathbf{M}$ is given by Equation (7).

$$\mathbf{M} = \begin{pmatrix} 0.2 & 0.75 \\ 0.75 & 1.0 \end{pmatrix} \tag{7}$$

$$e^{-\frac{(\mathbf{x}-0.5)^T \mathbf{M}^T \mathbf{M}(\mathbf{x}-0.5)}{2*0.002}} \tag{8}$$

Obviously, a standard RBNN with a single neuron centered on (0.5, 0.5) will not be able to correctly learn this function, because the standard activation function uses a Euclidean distance which is symmetrical. However, our GA should be able to learn the matrix $\mathbf{M}$ used to generate the domain. In order to get a proof-of-concept using this simple problem, we trained our system using a single neuron centered on $(0.5, 0.5)$ with a $\sigma^2 = 0.008$ (four times the σ^2 used to generate the domain). The GA was run using the parameters shown in Table 1.

Generations	30
Tournament size	2
Population size	20
Elitism	1
Crossover probability	0.6
Mutation probability	0.03

Table 1. Parameters of the genetic Algorithm for the Gaussian domain

In addition, 3 bits were used for the integer part, and 5 bits for the fractionary part. Only symmetrical matrices were allowed. After 30 generations, the following

matrix was obtained (see Equation (9)), which approached the function very well (it achieved a $5.867x10^{-5}$ error).

$$M = \begin{pmatrix} 0.46875 & 1.50000 \\ 1.50000 & 2.00000 \end{pmatrix} \tag{9}$$

Matrix 9 does not match matrix 7 (the one used to generate the domain), although it can be seen that their components approximately double the ones in the domain matrix. In any case, it is the activation functions that must be the same, in order for the 1 neuron RBNN to approximate perfectly the function. That is, the following equality has to be satisfied (see Equation 10):

$$(1/\sigma_1^2)[M_1^T M_1] = (1/\sigma_2^2)[M_2^T M_2] \tag{10}$$

where σ_1^2 and M_1 refer to the parameters used to generate the domain, σ_2^2 is the variance of the neuron, and M_2 is the matrix obtained by the genetic algorithm. This equality is almost satisfied, as Equation (11) shows.

$$\begin{pmatrix} 301.25 & 450 \\ 450 & 781.25 \end{pmatrix} \sim \begin{pmatrix} 283.2 & 421.8 \\ 421.8 & 757.8 \end{pmatrix} \tag{11}$$

It is interesting to remark that even though the σ^2 of the neuron (0.008) was not the same than the one used to generate the domain ($\sigma^2 = 0.002$), the GA managed to fit the domain function by appropriately escalating the components of the evolved matrix.

3.2 The Mackey-Glass Domain

The Mackey-Glass time series is widely regarded as a benchmark for comparing the generalization ability of RBNN [17, 18, 19]. The task for the RBNN is to predict the value of the time series at point $x[t + 50]$ from the earlier points ($x[t], x[t - 6]$, $x[t - 12], x[t - 18]$). It is a chaotic time series created by Equation (12):

$$\frac{dx(t)}{dt} = -bx(t) + a\frac{x(t-\tau)}{1 + x(t-\tau)^{10}} \tag{12}$$

1 474 patterns were generated for the Mackey-glass series, and values were normalized in $(0, 1)$. First, we ran some preliminary experiments in order to determine the number of neurons required. The minimum error was obtained with about 25 neurons. Also, these preliminary tests showed that 400 learning cycles and a 0.002 learning rate were reasonable values in this domain.

We tested two configurations of the system: allowing only diagonal matrices, and allowing general symmetrical matrices. Table 2 summarizes the parameters used. Two bits were used for the integer part, and three for the fractionary part.

Table 3 displays the results comparing the performance of a RBNN using a standard Euclidean distance and evolved distances. 5-fold crossvalidation results are

Generations	50
Tournament size	2
Population size	15
Elitism	1
Crossover probability	0.7
Mutation probability	0.01

Table 2. Parameters of the genetic algorithm for the Mackey-Glass problem

shown for both a diagonal matrix and a general symmetrical matrix. Improvements of 32.3 % and 6.4 % (respectively) can be observed. In this domain, using a diagonal matrix seems to be better than using a symmetrical matrix. In order to get a better understanding of results in this domain, we observed the values of the components of the matrices evolved by the GA. As we used a 5-fold crossvalidation procedure, 5 matrices were evolved. We observed that none of the components outside the diagonal are significantly different than 0 (taking into account the 5-folds, the median for these components is very close to 0). This means that for this domain, a symmetrical matrix does not give any advantage over a diagonal matrix. As a symmetrical matrix has many more parameters to be adjusted, it is more difficult for the GA to get the correct result.

Distance used	Test Error	Improvement (%)
RBNN Euclidean	0.015116	
RBNN GA diagonal	0.010237	32.3 %
RBNN GA symmetrical	0.014145	6.4 %

Table 3. Comparison of results between Euclidean and evolved distances (5-fold crossvalidation)

3.3 The Venice Lagoon Time Series Domain

This real world time series represents the behavior of the water level at Venice lagoon. Unusual high tides result from a combination of chaotic climatic elements with the more normal, periodic, tidal systems associated with a particular area. The prediction of high tides has always been the subject of intense interest, not only from a human point of view, but also from an economic one, and the water level of Venice Lagoon is a clear example of these events [20, 21].

The goal in this work is to predict only the next sampling time and a nonlinear model using the six previous sampling times, i.e. data of the six previous hours, may be appropriate. Thus, the function to be learned is:

$$x(t) = f(x(t-1), x(t-2), x(t-3), x(t-4), x(t-5), x(t-6)) \tag{13}$$

A data set of 4 000 points corresponding to the water level measured each hour has been extracted from available data (water level of Venice Lagoon between 1 980

and 1 994 sampled every hour). This set has been chosen in such a way that both stable situations and high water situations appear represented in the set. High-water situations are considered when the level of water is not lower than 110 cm. Values are normalized in $(0, 1)$. As in the previous domain, we ran some preliminary experiments in order to determine the number of neurons required, corresponding to the minimum error to RBNN with 25 neurons. Also, the same learning rate and number of cycles were used. The GA parameters and the number of bits used for the binary representation are the same that in the previous domain (see Table 1).

Both configurations of the system are tested, using a 5-fold crossvalidation procedure, corresponding to the diagonal and the symmetrical matrices, and their performance is compared with standard RBNN, when only Euclidean distances are used. Table 4 shows the results. Significant improvements for both configurations can be observed. When diagonal matrices are evolved, the RBNN performance improves a 44.5 % with respect to RBNN with Euclidean activation functions. The improvement reaches a 51 % when symmetrical matrices are used instead.

Distance used	Test Error	Improvement (%)
RBNN Euclidean	0.056584	
RBNN GA diagonal	0.031407	44.5 %
RBNN GA symmetrical	0.027726	51.0 %

Table 4. Comparison of results between Euclidean and evolved distances for the Venice Lagoon time series (5-fold crossvalidation)

3.4 The Ripley Data Domain

This artificially generated dataset has been used in [25]. Each pattern has two real-valued coordinates and a class which can be 0 or 1. Each class corresponds to a bimodal distribution that is a balanced composition of two normal distributions. Covariance matrices are identical for all the distributions and the centers are different. One of the issues that make this domain interesting is the big overlap existing between both classes. Due to this strong overlap, RBNN usually obtain poor results with this domain. Here, we are interested in improving the performance of classical RBNN using generalized Euclidean distances.

As in the previous domains, we ran some preliminary experiments in order to determine the number of neurons required, corresponding the minimum error to RBNN with 10 neurons. Also, the same learning rate and number of cycles were used. The GA parameters and number of bits used in the binary representation are also identical (see Table 1).

Table 5 shows the classification rates achieved by the different kinds of RBNN. It is possible to observe that the results corresponding to RBNN using both the diagonal matrix and the general one are better that the ones obtained when a purely Euclidean distance is used.

Distance used	Classif. Rate
RBNN Euclidean	73.6 %
RBNN GA diagonal	75.6 %
RBNN GA symmetrical	76.0 %

Table 5. Comparison of results between Euclidean and evolved distances for the Ripley data set (5-fold crossvalidation)

4 CONCLUSIONS

One of the problems of RBNN is the symmetrical nature of their activation function: the activation of a neuron only depends on the Euclidean distance from the input pattern to the neuron center, without taking into account the importance of different attributes; and, in general, there is no guarantee that the Euclidean distance is the best suited for a particular classification or regression problem. This issue can be approached by altering the metric used in the activation function. The learning method presented in this work uses a generalized Euclidean distance function which is determined in such a way that it minimizes the error of the network. This is achieved by a genetic algorithm whose individuals are generalized distance matrices and whose fitness function depends on the prediction accuracy attained by the network.

Our GA approach has been tested on four domains. The first one is a simple synthetic domain that helps understand the system. The rest of the domains are: the Mackey-Glass and the Venice Lagoon time series and a classification problem, the Ripley data set. It has been shown that using both diagonal and symmetrical evolved matrices improves prediction accuracy over a purely Euclidean distance.

Although RBNN have been used as the learning element in this paper, our GA-based method is relevant for any other machine learning technique where distances are involved. For instance, the family of nearest neighbor algorithms, support vector machines, or locally weighted regression methods would be clear candidates for our approach.

REFERENCES

[1] Moody, J. E.—Darken, C.: Fast Learning in Networks of Locally Tuned Processing Units. Neural Computation, Vol. 1, 1989, pp. 281–294.

[2] Ghosh, J.—Nag, A.: An Overview of Radial Basis Function Networks. R. J. Howlett and L. C. Jain (Eds.), Physica Verlag, 2000.

[3] Broomhead, D. S.—Lowe, D.: Multivariable Functional Interpolation and Adaptative Networks. Complex Systems, Vol. 2, 1988, pp. 321–355.

[4] Powell, M.: The Theory of Radial Basis Function Approximation in 1990. Advances in Numerical Analysis, Vol. 3, 1992, pp. 105–210.

[5] Park, J.—Sandberg, I. W.: Universal Approximation and Radial-Basis-Function Networks. Neural Computation, Vol. 5, 1993, pp. 305–316.

[6] Musavi, M. T.—Ahmed, W.—Chan, K. H.—Faris, K. B.—Hummels, D. M.: On the Training of Radial Basis Function Classifiers. Neural Networks, Vol. 5, 1992, pp. 595–603.

[7] Poggio, T.—Girosi, F.: Regularization Algorithms for Learning That Are Equivalent to Multilayer Networks. Science, Vol. 247, 1990, pp. 978–982.

[8] Poggio, T.—Girosi, F.: Networks for Approximation and Learning. Proceedings of the IEEE 1990, Vol. 78, 1990, No. 9, pp. 1481–1497.

[9] Atkenson, C. G.—Moore, A. W.—Schaal, S.: Locally Weighted Learning. Artificial Intelligence Review, Vol. 11, 1997, pp. 11–73.

[10] Tou, J. T.—Gonzalez, R. C.: Pattern Recognition Principles. Addison-Wesley, 1974.

[11] Weisberg, S.: Applied Linear Regression. New York: John Wiley and Sons, 1985.

[12] Babiloni, F.—Bianchi, L.—Semeraro, F.—del R-Millan, J.—Mourino, J.—Cattini, A.—Salinari, S.—Marciani, M. G.—Cincotti, F.: Mahalanobis Distance-Based Classifiers Are Able to Recognize EEG Patterns by Using Few EEG Electrodes. In Engineering in Medicine and Biology Society, 2001, Proceedings of the 23rd Annual International Conference of the IEEE, Vol. 1, pp. 651–654.

[13] Randolph-Gips, M. M.—Karayiannis, N. B.: Reformulated Radial Basis Function Neural Networks With Adjustable Weighted Norms. International Journal of Intelligent Systems, Vol. 18, 2003, pp. 1065–1085, Wiley Periodicals Inc.

[14] Wettschereck, D.—Aha, D. W.—Mohri, T.: A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. Artificial Intelligence Review, Vol. 11, 1997, Nos. 1–5, pp. 273–314.

[15] Valls, J. M.—Aler, R.—Fernández, O.: Using a Mahalanobis-Like Distance to Train Radial Basis Neural Networks. IWANN 2005, Lecture Notes in Computer Science 3512, Proceedings of the 8th International Work-Conference on Artificial Neural Networks IWANN 2005, pp. 257–263, 2005.

[16] Holland, J. H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, 1975.

[17] Leonardis, A.—Bischof, H.: An Efficient MDL-Based Construction of RBF Networks. Neural Networks, Vol. 11, 1998, pp. 963–973.

[18] Orr, M. J. L.: Introduction to Radial Basis Neural Networks. Technical Report, Centre for Cognitive Science, University of Edinburgh, 1996.

[19] Yingwei, L.—Sundararajan, N.—Saratchandran, P.: A Sequential Learning Scheme for Function Approximation Using Minimal Radial Basis Function Neural Networks. Neural Computation, Vol. 9, 1997, pp. 461–478.

[20] Moretti, E.—Tomasin, A.: Un Contributo Matematico All-Elaborazione Previsionale dei Dati di Marea a Venecia. Boll. Ocean. Teor. Appl., Vol. 1, 1984, pp. 45–61.

[21] Michelato, A.—Mosetti, R.—Viezzoli, D.: Statistical Forecasting of Strong Surges and Aplication to the Lagoon of Venice. Boll. Ocean. Teor. Appl., Vol. 1, 1983, pp. 67–83.

[22] Tomasin, A.: A Computer Simulation of the Adriatic Sea for the Study of Its Dynamics and for the Forecasting of Floods in the Town of Venice. Comp. Phys. Comm., Vol. 5, 1973, p. 51.

[23] Vittori, G.: On the Chaotic Features of Tide Elevation in the Lagoon Venice. Proc. of the ICCE-92, 23rd International Coference on Coastal Engineering, pp. 4–9, 1992.

[24] Zaldvar, J. M.—Gutiérrez, E.—Galván, I. M.—Strozzi, F.—Tomasin, A.: Forecasting High Waters at Venice Lagoon Using Chaotic Time Series Analysis and Nonlinear Neural Networks. Journal of Hydroinformatics, Vol. 2, 2000, pp. 61–84.

[25] Ripley, B. D.: Pattern Recognition and Neural Networks. Cambridge: Cambridge University Press, 1996.

José M. Valls received his Ph. D. in computer science at Universidad Carlos III of Madrid (Spain) in 2004. He joined the Computer Science Department at the same university in 1998, being Associate Professor since 2004. He is enrolled in the Neural Networks and Evolutionary Computation Laboratory of this university. His current research focuses on the application of neural networks, evolutionary computation and other soft computing techniques to engineering problems.

Ricardo Aler is Associate Professor at Universidad Carlos III Computer Science Department. He has researched in several areas, including automatic control knowledge learning, genetic programming, and machine learning. He has also participated in international projects about automatic machine translation and optimising industry processes. He holds a Ph. D. in computer science from Universidad Politécnica de Madrid (Spain) and a M. Sc. in decision support systems for industry from Sunderland University (UK). He graduated in computer science at Universidad Politécnica de Madrid.

Oscar Fernández graduated in computer science and statistics at Universidad Carlos III of Madrid (Spain). He has worked in several companies as consultant and systems engineer. His research interest is mainly focused at data mining, data analysis and its integration in corporative systems.