Feature Selection and Outliers Detection with Genetic Algorithms and Neural Networks

Agusti Solanas [a,1], Enrique Romero [b,2], Sergio Gómez [a], Josep M. Sopena [c], Rene Alquézar [b] and Josep Domingo-Ferrer [a,3]

[a] *Universitat Rovira i Virgili*
[b] *Universitat Politècnica de Catalunya*
[c] *Universitat de Barcelona*

Abstract. This paper presents a new feature selection method and an outliers detection algorithm. The presented method is based on using a genetic algorithm combined with a problem-specific-designed neural network. The dimensional reduction and the outliers detection makes the resulting dataset more suitable for training neural networks. A comparative analysis between different kind of proposed criteria to select the features is reported. A number of experimental results have been carried out to demonstrate the usefulness of the presented technique.

Keywords. Feature Selection, Genetic Algorithms, Neural Networks,

1. Introduction

Feature selection is a well-known problem and has been deeply studied by the artificial intelligence community. When we deal with a supervised learning task (e.g. training a neural network), this problem consist in selecting a subset of features from a given set of labeled fixed-length feature vectors.

Lots of different techniques have been designed and tested to solve this problem. In [7] the problem is addressed by using the information theory and by maximizing the amount of information that a selected subset of features contains. Cohen et al. [1] proposed a feature selection scheme (i.e. Principal Feature Analysis PFA) based on a modification of the very popular method of Principal Components Analysis (PCA) and applied it to face tracking. In [4] a genetic algorithm is used to select a subset of features for performing classification with fuzzy models. All these methods are focused on the data and the application and cannot be generalized easily.

[1]Correspondence to: Departament d'Enginyeria Informàtica i Matemàtiques. Universitat Rovira i Virgili. Av. Països Catalans, 26. Tarragona 43001. Tel.: +34 977 558270 ; Fax: +34 977 559710; E-mail: agusti.solanas@urv.net.

[2]This work was partly supported by the Spanish Ministry of Education and Science under project DPI2002-03225

[3]This work was partly supported by the Spanish Ministry of Education and Science under project TIC2001-0633-C03-01 "PROPRIETAS".

In this paper we tackle the problem of selecting a subset of features in order to maximize the performance of a given neural network by using genetic algorithms and neural networks. In [10] this problem is addressed by using neural networks only. Neural networks have been widely used over a wide range of different application domains achieving very good results where other techniques simply failed. Although nowadays neural networks are a very common tool for problem solving, the design of a neural network is not an easy task and expert knowledge is usually needed to obtain a good solution. Moreover, when a very well tuned neural network is able, its performance can be dramatically decreased if the data in the training set are not properly selected and precompiled.

If the data set contains elements of a high dimensionality we believe that a dimensional reduction can help to reduce the undesired redundancy of the data. Thus, the problem is to select the dimensions which help to obtain a better performance.

In this article, we focus on the problem of selecting the dimensions (i.e. our features) which cause the neural network to achieve a better result, that is to find the best set of dimensions in which to project the data for training and testing the neural network. At the same time, the dimensional reduction obtained can be applied to several machine learning approaches (i.e. not only neural networks).

The rest of the article is organized as follows. In section two a brief theoretical model is exposed. In section three our algorithms and methodologies are presented. In section four the experimental results are reported. Finally, in section five the article concludes.

2. Model and Objectives

We have a dataset consisting of a number N of different vectors $P = \{\vec{P_1}, \vec{P_2}, \ldots, \vec{P_N}\}$. Each vector $\vec{P_i}$ has a fixed number of dimensions D, where each dimension represents a feature that can be considered as an input variable to a neural network. Thus, a vector is defined as $\vec{P_i} = \{f_1, f_2, \ldots, f_D\}$.

Each vector $\vec{P_i}$ can be classified in a predefined class. Thus, a prototype $\vec{C_i}$ (e.g. the centroid) of each class can be computed.

The main objective of our technique is to automatically reduce the number of dimensions in order to minimize the noise and redundancy of the dataset, while maximizing the neural network performance. Moreover, this technique can be used to detect the most relevant dimensions in the dataset, which is a very difficult task when working with very high dimensionality data.

3. Methodology

Reducing the noise and redundancy of a dataset can be achieved by two different main ways depending on the type and amount of noise of the data (i.e. gaussian noise, impulsional noise).

If the data are very noisy, we propose the use of an outliers detection technique. On the contrary, if the data have been precompiled and are not noisy, we propose to utilize a dimensionality reduction to achieve the objectives exposed in the previous section.

Reducing noise and redundancy by selecting the most important dimensions is not a trivial task. We address this problem by using a genetic algorithm to encode the selection

Algorithm 1 Detection of Outliers

```
1)  function DetectOutliers (P) return O, C
2)      for all classes C do
3)          for all instances P do
4)              if P_j ∈ C_i then
5)                  C_i = C_i + P_j
6)                  N_i = N_i + 1
7)              end if
8)          end for
9)          C_i = C_i / N_i
10)     end for
11)     P^sorted = OrderDecreasingByDistance(P,C)
12)     for i in (1..λ) do
13)         O_i = P_i^sorted
14)     end for
15) end function DetectOutliers
```

of the dimensions and a neural network to analyze the correctness of the selection. In the next sections we expose our algorithms to perform both tasks.

3.1. Outliers Detection

In a set of real data, the existence of outliers is very common . An outlier is an element in the dataset that is, in average, very different from the others. The outliers exist because of the very nature of the real data. However, they can be considered as errors produced by impulsional noise during the data collection.

When a dataset has an important number of outliers, it is difficult to properly train a neural network because its weights become biased by their influence. Thus, we propose a simple algorithm to detect and erase outliers from the dataset (See Algorithm 1).

The method consists in computing the centroids C of each class by applying the next formula:

$$\vec{C_i} = \frac{1}{N_i} \sum_{j=0}^{N_i} \vec{P_j^i}, \tag{1}$$

where N_i is the number of instances in the class i and $\vec{P_j^i}$ is the instance j in the class i. Once all the centroids C have been obtained, the distance between each centroid and the elements of its class can be computed. Finally, a number λ of instances with the highest distance to their centroids are labeled as outliers.

3.2. Selection of Dimensions with Genetic Algorithms

As we have introduced in the previous sections, the selection of the right dimensions D' (i.e. features) of a given problem can improve the success rate achieved by a trained neural network. To select the most important dimensions we propose a method based on using genetic algorithms as described in [6].

In this section, the basis of our genetic algorithm are exposed. Then, we propose and analyze two different criteria for fitting the chromosomes in the population.

3.2.1. Our Genetic Algorithm

A genetic algorithm is a method for moving from one population of "chromosomes" to a new population by using a kind of "natural selection" together with the genetics-inspired operators of crossover, mutation, and inversion. Each chromosome consists of "genes", each gene being an instance of a particular "allele" [8].

A genetic algorithm depends on some parameters: population size, mutation rate, crossover rate and so on. Although statistical techniques can be used to tune these parameters [2], they are typically initialized by using the recommendations of experts [3][5].

Specifically, our genetic algorithm uses a population of 10 chromosomes of variable length depending on the dimension D of the problem. Each gene in the chromosome can be set to 0 (i.e. not selected dimension) or 1 (i.e. selected dimension). The crossover rate is set up to 0.7 and the mutation rate to 0.2. It uses the well-known roulette-wheel selection algorithm. This selection algorithm is based on weighting the chromosomes of the population using its fitness. Thus, the probability of an element being selected for passing to the next generation is proportional to its fitness. This method tends to converge quickly because it does not help in maintaining the diversity of the population. On the one hand, this is a shortcoming because the solution may be worse. However, it is faster than other methods and it is good enough to demonstrate our ideas.

Finally, we have defined two fitness functions for our genetic algorithm. The first one is based on a geometrical criterion while the second considers the results obtained by a neural network to fit each chromosome. In the next section we discuss in detail both approaches.

3.2.2. Geometrical criterion

The geometrical criterion is based on the intuition presented in [11] that to increase the distance between the centroids and to reduce the dimensionality of the problem is useful to improve the results obtained by a neural network because the patterns can be more easily separated.

Based on this intuition, the chromosomes are evaluated in three steps:

- Projection of all instances from the initial dimension D to a dimension D', where D' equals the number of genes of the chromosome with a one (see Figure 1).
- Computation of the centroids $vecC_i$ in the projected dimension.
- Computation of the average distance between the obtained centroids $\vec{C}_i$

The projection of the instances P consist in erasing the dimensions (i.e. the features) for which the genes of the chromosome equals to zero. Figure 1 shows a projection of a set of instances P of dimension $D = 5$ over a dimension $D' = 2$. Once the instances are projected, they are used to compute the centroids $\vec{C}_i$ by applying Formula 1. Finally, the average distance between all centroids is computed. Thus, the fitness of a chromosome increases when the average distance obtained is bigger, hence, the instances may be easily separated.

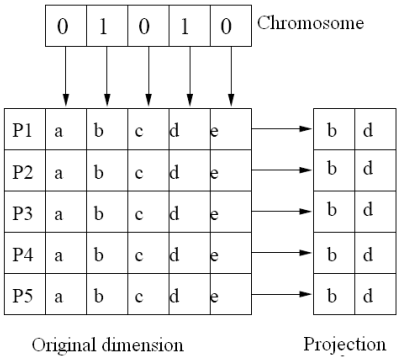

Figure 1

Evaluation of a chromosome. Projection step.

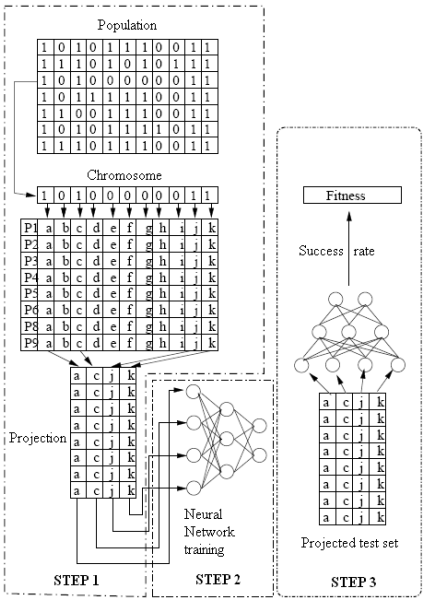

Figure 2

Evaluation scheme with a neural network

3.2.3. Neural Network-based criterion

Using a neural network to evaluate a chromosome represents a different approach. We divide the process in three steps (see Figure 2):

- Projection of the instances P as described in the previous section
- Train the neural network with the projected training set
- Evaluate the success rate of the neural network with the projected test set

The projection step is identical to the one described in the previous section. Next, in the second step, a neural network is trained by using the projection of the training set. The utilized neural network is a multilayer perceptron consisting of an input layer, a hidden layer and an output layer. The number of neurons in the input layer equals the number of dimensions of the projected dataset. On the other layers, the number of neurons are designed according to the problem being solved. Once the network is trained, it is tested by using the projected test set and some measurements are taken:

- Success rate in the test set (1 experiment)
- Success rate in the test set (10-fold cross-validation)

The fitness assigned to the chromosome equals the success rate obtained by the neural network in the projected test set. The most important shortcoming of this approach compared to the previous one is its computational cost. In this case, to evaluate a chromosome we need to train and test a neural network. Thus, the computational cost is clearly higher. However, we consider that this process of feature selection is an off-line pre-process and, hence, the computational cost is not a critical factor.

4. Experimental Results

In this section we describe the experiments that we have carried out over two problems taken from the UCI Machine Learning Repository [9]: The *Hepatitis* problem and the *Ionosphere* problem. We have designed different multilayer perceptrons for each problem to test our methods over them.

Using the hepatitis problem, we have tested the influence of the outliers and the dimensional reduction over the performance of the neural network. The ionosphere problem does not have a noisy dataset, hence, we have focused on the dimensional reduction only.

When testing the neural network based on the "wrapping" method we apply two different test techniques: A simple test and a 10-fold cross validation test[1]. A simple test consists in training the neural network only once using 90% of the instances to train and 10% to test. A 10-fold crossvalidation test consists in training the neural network and test it 10 times re-selecting randomly the instances of the train and test groups with each test step and obtaining the average success rate.

4.1. Hepatitis problem

The hepatitis problem consists in determining whether a patient with hepatitis will survive. To perform this task, a dataset of 155 instances is given. Each instance has 20 dimensions (i.e. 19 features and a class attribute) and each dimension gives information about: class, age, sex, antivirals, fatigue, malaise, anorexia, liver big, liver firm, spleen palpable, spiders, ascites, varices, bilirubin, alk phosphate, sgot, albumin, protime and histology. The instances are divided in two groups: in the first group there are 32 instances and the patient dies. In the second group there are 123 instances and the patient survives.

To solve this problem we use a multilayer perceptron consisting of: an input layer of 19 neurons (i.e. one neuron for each feature), a hidden layer of 20 neurons and an output layer of 2 neurons. When the designed neural network is applied with no modification it obtains a success rate of 79.3% and we call it the base line. The results obtained by applying our methods are shown in Table 1.

Table 1. Results of the Hepatitis Problem

Experiment	% Success	Deviation	number of Dimensions
Base line	79.3	± 9.21	19
Erasing Outliers	83.3	± 9.40	19
Geometrical Criterion	80.0	± 9.94	5
Wrapping (1 train)	84.4	± 9.53	8.6
Wrapping (10-fold train)	87.6	± 8.57	6.3

The obtained results let us conclude that:

- Erasing the outliers significantly improves the success rate when working with noisy datasets.

[1] Both using back-propagation

- The genetic algorithm based on the geometrical criterion is very effective in reducing the dimensionality of the problem but does not help to increase the success rate of the neural network.
- The wrapping method clearly increases the success rate of the network while, at the same time, reducing the dimensionality of the problem.

These results prove that the proposed methods are useful to reduce the dimensionality of the problem while maintaining or increasing the success rate.

4.2. Ionosphere problem

The ionosphere problem consists in determining whether an electronic pattern exists in the ionosphere. The dataset is a collection of radar measurements. This radar dataset was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this database are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal [9]. Thus, we have 34 features and the class attribute.

In this case, we do not consider the outliers and the geometrical criterion because the data are not noisy. Thus, we pay attention to the wrapping methods in order to confirm the behavior obtained in the previous problem. We use a multilayer perceptron consisting of: an input layer with 34 neurons, a hidden layer with 40 neurons and an output layer of 2 neurons.

Table 2. Results of the Ionosphere Problem

Experiment	% Success	Deviation	Number of Dimensions
Base line	90.57	± 4.62	34
Wrapping (1 train)	90.68	± 5.16	11
Wrapping (10-fold train)	92.86	± 4.31	14

We observe that the base line of the problem is very high (i.e. $\approx 90\%$) and is very difficult to improve the result. However, our method slightly improves the success rate of the neural network and dramatically reduces the dimensionality of the problem (i.e. more than a 50%). Thus, our method proves to be useful to increase the success rate of the neural network while improving its efficiency (i.e. to reduce the diensionality of the problem).

5. Conclusions

We have presented a set of methods to increase the efficiency of a neural network. On the one hand, we have presented a simple method for detecting outliers on noisy datasets. On the other, we propose a complete architecture based on genetic algorithms for selecting the features in a dataset, which improves the performance of a neural network.

The experimental results have shown the correctness of the presented technique and demonstrate the usefulness of our methods to properly reduce the dimensionality of the problem, while improving the neural network success rate and efficiency.

Although the presented methods are complete, they can be extended in the following ways:

- Obtain an automatic method to define the number of outliers λ.
- Use a more complex genetic algorithm to improve the quality of the solution.
- Combine the extraction of outliers with the dimensional reduction.

References

[1] I. Cohen, Q. Tian, X. S. Zhou, and T. S. Huang. Feature selection using principal feature analysis. In *Proceedings of the International Conference of Image Processing*, pages 0–0, Rochester, New York, September 2002.

[2] A. Czarn, C. MacNish, K. Vijayan, B. Turlach, and R. Gupta. Statistical exploratori analysis of genetic algortihms. *IEEE Transactions on Evolutionari Computation*, pages 405–421, August 2004.

[3] K. A. DeJong. *Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, Dept. Comput. Commun. Sci., University of Michigan, 1975.

[4] C. Emmanouilidis, A. Hunter, J. MacIntyre, and C. Cox. Multiple - criteria genetic algorithms for feature selection in neurofuzzy modeling. In *Proceedings of the IJCNN'99, the International Joint Conference on Neural Networks*, pages 4387–92, Washington, USA, July 1999.

[5] B. Freisleben and M. Härtfelder. Optimization of genetic algorithms by genetic algorithms. In *In Proc. Int. Conf. Artificial Neural Networks and Genetic Algorithms*, pages 392–399, Viena, Italy, 1993.

[6] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[7] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning (ML)*, pages 284–292, Bari, Italy, July 1996.

[8] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT press, 1999.

[9] University of California Irvine. http://www.ics.uci.edu/ mlearn/MLSummary.html.

[10] E. Romero, J. M. Sopena, G. Navarrete, and R. Alquézar. Feature selection forcing overtraining may help to improve performance. In *Proceedings International Joint Conference on Neural Networks, IJCNN-2003*, volume 3, pages 2181–2186, Portland, Oregon, 2003.

[11] A. Solanas. *Genetical Selection of features*. Master thesis, Dept. Comput. Sci. Math. , University Rovira i Virgili, 2002.