Feature Selection Using Artificial Neural Networks

Conference Paper · October 2008

DOI: 10.1007/978-3-540-88636-5_34 · Source: DBLP

5 authors, including:

Sergio Ledesma
Universidad de Guanajuato
122 PUBLICATIONS **969** CITATIONS
SEE PROFILE

Gustavo Cerda-Villafana
Universidad de Guanajuato
27 PUBLICATIONS **97** CITATIONS
SEE PROFILE

Juan Gabriel Avina-Cervantes
Universidad de Guanajuato
167 PUBLICATIONS **1,569** CITATIONS
SEE PROFILE

Donato Hernández
Universidad de Guanajuato
41 PUBLICATIONS **331** CITATIONS
SEE PROFILE

Feature Selection Using Artificial Neural Networks

Sergio Ledesma, Gustavo Cerda, Gabriel Aviña, Donato Hernández, and Miguel Torres

Dept. of Electrical & Computer Engineering. University of Guanajuato. Salamanca, Gto. 36700, México
{selo,gcerdav,avina,donato,mtorres}@salamanca.ugto.mx

Abstract. Machine learning is useful for building robust learning models, and it is based on a set of features that identify a state of an object. Unfortunately, some data sets may contain a large number of features making, in some cases, the learning process time consuming and the generalization capability of machine learning poor. To make a data set easy to learn and understand, it is typically recommended to remove the most irrelevant features from the set. However, choosing what data should be kept or eliminated may be performed by complex selection algorithms, and optimal feature selection may require an exhaustive search of all possible subsets of features which is computationally expensive. This paper proposes a simple method to perform feature selection using artificial neural networks. It is shown experimentally that genetic algorithms in combination with artificial neural networks can easily be used to extract those features that are required to produce a desired result. Experimental results show that very few hidden neurons are required for feature selection as artificial neural networks are only used to assess the quality of an individual, which is a chosen subset of features.

1 Introduction

Feature selection is commonly associated with the curse of dimensionality that affects machine learning and optimization problems solved by backwards induction. Feature selection is important because it may reduce the number of features of a data set, and hence, speed up the learning process while making the data set easier to understand and manage, see [7]. Feature selection may be performed by trial and error of all possible subsets of the original data set, this, of course, is very inefficient and there are several systematic methods to handle the selection process, see [2]. We proposed the use of artificial neural networks (ANN) in combination with genetic algorithms (GA) to assist the search process and select from the data set those most relevant features.

This paper is organized as follows. In Section 2, background information about genetic algorithms and ANN training is reviewed. In Section 3, a simple method to perform feature selection is presented. In Section 4, simulation experiments are performed to verify our results. Finally, Section 5 presents some conclusions and direction for future work.

A. Gelbukh and E.F. Morales (Eds.): MICAI 2008, LNAI 5317, pp. 351–359, 2008.

2 Background Information

Generally, inductive learning methods work best when supplied with features of a state that are relevant to its evaluation, rather than with just raw state description [10]. Hence, feature selection plays an important role when extracting information from a data set. Unfortunately, features may be correlated and have to be analyzed by groups rather that individually, making the process of feature selection complicated. As an result, in [8], a theory of mutual information that demonstrates the relationship of four selection schemes: maximum dependency, mRMR, maximum relevance, and minimal redundancy is presented; a specific application of this approach is offered in [3].

2.1 The Genetic Algorithm

The genetic algorithm (GA) is a general optimization method that has been applied to many problems including neural network training [9]. Additionally, it has been pointed out in [5] that because the usefulness of the genetic algorithm depends most highly on the representation of the solution, any number of numerical and symbolic problems may be optimized. As it will be shown later, feature selection problems may be solved using genetic algorithms by analyzing the features in groups and reducing highly the number of subsets to evaluate.

2.2 Artificial Neural Networks

They are made out of neurons and are excellent to extract information from a data set even under noisy conditions. During a process, called training, the network's weights are adjusted until the actual network's output and a desired output are as close as possible, see [10]. In general, artificial neural networks can be used to map an input to a desired output, classify data or learn patterns. Hence, artificial neural networks can also be used to assess the quality of an individual (from a genetic algorithm pool), and therefore, perform indirectly feature selection.

3 Proposed Method

There are several existing methods to perform feature selection, see the references in [8]. These methods are based on several criteria: maximum relevance, minimum redundancy, etc. We propose the use of the genetic algorithm and the mean-squared error obtained during the training of a neural network to perform feature selection.

This paper focuses on two typical feature selection problems: mapping and classification. In a mapping problem, a set of input values must produce ("map") a set of desired output values. The feature selection problem consists on reducing the number of variables in the input set while producing the same output. Those values that can be removed from the input set may not contain useful information

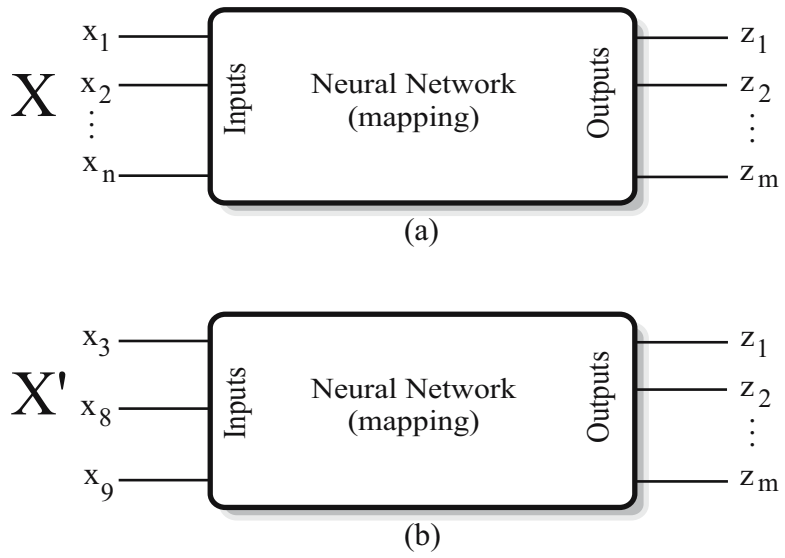

Fig. 1. An artificial neural network used for mapping: (a) using all features, (b) using a subset of features

to produce the desired output or may contain redundant information that is already contained in other input variable (feature) or other set of inputs. This can be observed at Figure 1.a that shows a neural network that maps the input set $X = \{x_1, x_2, \cdots, x_n\}$ into the output set $Z = \{z_1, z_2, \cdots, z_m\}$.

On a classification problem, an input set (feature set) $X = \{x_1, x_2, \cdots, x_n\}$, allows identifying the class that these input values belong. Figure 2.a shows an artificial neural network used as a classifier; the input set $X = \{x_1, x_2, \cdots, x_n\}$ let the neural network identify the class of each element of the input set. In this case, a feature selection problem consists on computing a new input set that is a subset of X, so that only those features that presumably contain useful information to identify the class are considered.

3.1 Coding Feature Selection Using a Genetic Algorithm

Genetic algorithms are non-deterministic algorithms that their success depends strongly on how the problem is coded or mapped into the domain of GA. A feature selection problem may map naturally to a set of binary numbers as it will be explained next.

Consider the input set $X = \{x_1, x_2, \cdots, x_n\}$ with n features, and suppose that there is a subset X' with k features taken from the set X, i.e., $X' = \{x_3, x_8, x_9\}$. Thus, X', a GA individual, can be coded as $0010000110\cdots$, where the first zero indicates that x_1 is not included in X', the third 1 indicates that x_3 is included in X', and so on. That is, those features that included in the subset X' are represented by one, and those features that left out are represented by zero. This simple coding

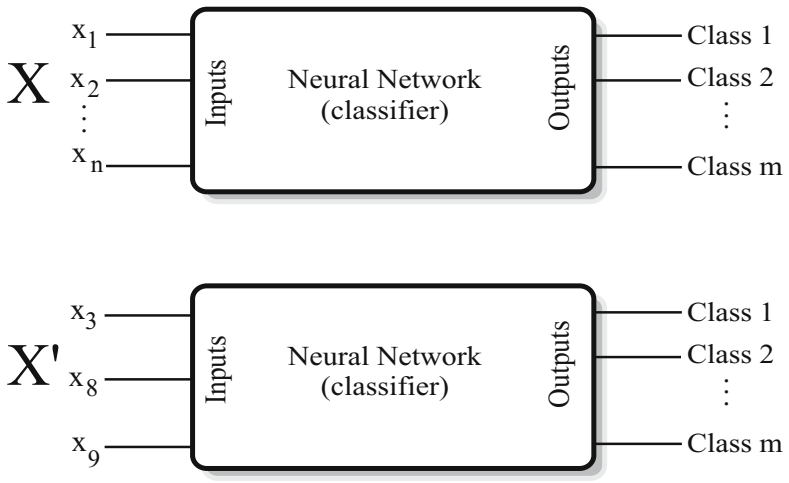

Fig. 2. Artificial neural network used for classification: (a) using all features, (b) using a subset of features

scheme may be used as long as the number of elements of X' be equal to the number of feature to select (k). Finally, it is important to note that each subset of X, a subset of features, may represent an individual with a specific fitness level.

3.2 Initial Pool

A genetic algorithm starts by creating an initial pool of individuals. Some implementations of GA require the creation of more individuals than the size of the initial pool and select only those more fitted individuals. However, when using neural networks for feature selection, it is not appropriate to create more individuals than required as assessing the fitness for each individual is time consuming. More about this will be addressed on subsection 3.4.

To create the initial pool, it is necessary to create a set of individuals; this is a two step process. First, an individual is created by randomly setting k of its bits to one. Second, we check if this individual is already in the pool; if it is, then a new individual is created; if it is not, then this individual is added to the pool. This process continues until the initial pool is full with different individuals.

Once the initial pool is filled, each individual must be evaluated to assess its fitness. Those most fitted individuals will be the parents of the next generation. This will be discussed at the end of this section.

3.3 Reproduction and Mutation

From one generation to the next, the success of GA depends highly on how the individuals are reproduced and mutated. For some implementations of GA,

Table 1. Reproduction used for feature selection

Parent-1 bit	Parent-2 bit	Child bit
0	0	0
0	1	1 with a probability of 0.1
1	0	1 with a probability of 0.1
1	1	1

reproduction and mutation are two separated steps; first the individuals are combined using the probability of crossover, then the new individuals are mutated using the probability of mutation. On the other hand, when using GA for feature selection, these two operations must be performed together as the number of features from generation to generation must remain constant.

Once the individuals from the current generation are evaluated, a list of possible parents is created. This list includes all individuals of the current generation, and a fitness level associated to each individual. Combining the probability of crossover and the fitness level of each individual two parents are chosen. Every time an individual is selected as a parent, its probability of selecting him again is reduced. After this, two new children are produced from these two parents. The process continues until the new generation is completely created.

In order to produce the children, the following algorithm was used. Suppose there are two parents X_1' and X_2', thus a bit position is randomly generated and the children bit is set using the rules on Table 1. Basically, if both parents include that feature, the child's bit is set to one; if non of the parents include that feature, this bit is set to zero; and if only one of the parents include this feature, this bit is set to one with a probability of 0.1. Right after this, the gene mutation occurs: a bit position is randomly generated and the child bit on this position is mutated with a given probability. This process continues until the child contains k bits set to one (k features). Once the first child has been created, a second child is created using the same mechanism just described.

After carefully observing Table 1, it can be seem odd to set a bit to 1 with a probability of 0.1 when only one of the parents has that feature. GA is based on diversity and fitness; a careful balance must exist between these two. If this probability is set to 0.5 or more the algorithm may not favor features that are present on both parents. If this probability is set to a value close to 0, some good features will die too soon killing diversity.

3.4 Fitness of an Individual

In order to choose the parents for the new generation and improve the new pool of individuals, it is very important to assign a fitness level to each individual. This process can be tricky as some criteria must be used.

Consider again Figure 1.a, in this case the artificial neural network is trained by applying it a specific input, and observing the actual network output. The training process uses the mean squared error (mse) measured between the actual

network and the desired output. The mse is typically computed using all ANN outputs and all training cases. Consequently, the learning process is managed by the value of the mse. When the mse reaches a desired value, the neural network has been trained and the ANN can be used to solve real life problems.

Figure 1.a shows an artificial neural network that is being trained using all features from the data set. Figure 1.b shows an artificial neural network that is being trained using only a subset of features.

Because of their fancy cleverness to discover hidden patterns and structures (even under noisy conditions), we propose the use of artificial neural networks to evaluate the fitness of an individual. That is, a given subset of features is used for training an artificial neural network; if the subset of features contains features that help the network to map the desired output, the resulting mse will be small. On the other hand, if this subset does not have information to get the desired output, the resulting ANN mse will be large.

Once the ANN learning process has completed, the individual's mse must be converted so that a very small mse maps to a high fitness level; the exponential function was used for this purpose see [6]. Finally, a list of parents for the next generation must be created using each individual's fitness. The parents with high fitness levels will have a high expected frequency of being selected from this list.

Consider now Figure 2 that shows an artificial neural network used for classification. Figure 2.a shows an artificial neural network that is being trained for classification using all the features in the data set; while Figure 2.b shows an artificial neural network that is being trained for classification using only a subset of features. For this case, the mse can be calculated using the fact that only one output neuron must be active. If the ANN activation function is the $logsig(x)$, the active value should be approximately 0.9 and the inactive value 0.1. On the other hand, if the ANN activation function is the $tanh(x)$, the active value should be approximately 0.9 and the inactive value -0.9.

4 Simulation Results

On machine learning, some practitioners recommend splitting the available data into two sets: one for training and another for validation. However, in some real life problems, there are awfully few data available to perform training and validation. Besides, it is possible that these two sets are very different or extremely small; making difficult to reach any valid conclusion from the data analysis. One typical solution, for short data set analysis, is to use the leave-one-out cross validation (LOOCV), which is a $K - fold$ cross validation with K being equal to the number of observations in the data set. Leave-one-out cross-validation consists on using only a single observation from the original data set for validation, and leaving the remaining observations for training. This process goes over until each observation in the data set is used once for validation.

LOOCV is useful for classification problems and has been used in some previous papers to test feature selection. Unfortunately the sets, used there, have relatively very few training cases are not useful to train artificial neural networks

without easily incurring in over-fitting. Consider for example the lung data set used in [3] with 325 features, 7 classes and only 73 training cases; even the simplest neural network (with very few hidden neurons) would commit overfitting. Similarly, the data sets of [1] or [4] are not most favorable for ANN learning. Therefore, some synthetic data sets are created and tested to validate our results.

4.1 Data Set with Uncorrelated Features

To evaluate the usefulness of the proposed method, we carried out experiments on a synthetic data set for a mapping problem with 60 input features, 5 outputs and 200 training cases. The input values, in this case the features, where a set of random values $X = \{x_1, x_2, \cdots, x_n\}$ uniformly distributed in $[-\pi, \pi]$ while the output values were synthesize as shown in table 2. Note that the desired output of the mapping problem was computed using only: x_{10}, x_{20}, x_{30}, x_{40}, and x_{50}; while the other input features were not used at all to compute the output. In other words, the problem consisted on selecting only 5 features from the 60 contained in the original data set while producing the desired output.

Once the data set was synthesize, a multi-layer feed-forward artificial neural network with five inputs, five outputs, and 9 hidden neurons was created and trained following the genetic algorithm to perform feature selection by the proposed method. During training, the ANN required 500 epochs or less to reach an mse of 0.0001. Because of the network size, finding the fitness of an individual may take some seconds when running on a computer Pentium 4 at 3.2 GHz.

It is important to mention that the number of hidden neurons is critical. If the number of hidden neurons is small, the resulting *mse* will be the same no matter what features are applied to the input. Therefore, the number of hidden neurons should be kept to the maximum just before committing over-fitting. This means that the number of training cases directly affects the number of hidden neurons. The software *NeuralLab* that is available at *Wikipedia* under "neural network software" provides the appropriate number of hidden neurons to avoid over-fitting. It is also possible to estimate the maximum number of hidden neurons once the number of training cases is given, because in order to compute the network weights (or unknowns) a minimum number of equations (training cases) is required.

The initial population was set to 100 individuals, and once the algorithm started, it was after only 8 generations by the time the algorithm selected: x_{10}, x_{20}, x_{30}, x_{40}, and x_{50} (the only features that were used to compute the output).

Table 2. Synthetic data set with uncorrelated features

Output	Value
1	$\sin(x_{10})$
2	$\sin(x_{20})$
3	$\cos(x_{30})$
4	$\cos(x_{40})$
5	$\cos(x_{50})$

Finally, note that the proposed method allowed a reduction from $5,461,512$ (as computed in Equation 1) to only 800 evaluations to select those features required to get the desired output.

$$\binom{60}{5} = 5,461,512 \tag{1}$$

4.2 Data Set with Correlated Features

To make the problem more difficult, we repeat the same experiments of subsection 4.1 but using correlated outputs as shown in Table 3. For this case, one of the outputs was computed using two inputs making more difficult to solve this feature selection problem.

Table 3. Synthetic data set with independent features

Output	Value
1	$\sin(x_{10})$
2	$\sin(x_{20})$
3	$\cos(x_{30} + x_{40})$
4	$\cos(x_{40})$
5	$\cos(x_{50})$

Once the data set was synthesize, an artificial neural network with five inputs, five outputs, and 9 hidden neurons was created and trained just as before to perform feature selection. The initial population was set to 100 individuals. However and because of the added correlation in the inputs, it took 10 generations for the proposed algorithm to select: x_{10}, x_{20}, x_{30}, x_{40}, and x_{50}.

5 Summary

Feature selection simplifies some mapping and classification problems by reducing the number of features to analyze. The proposed algorithm does not focus on the relevance or redundancy of the features; instead it passes this job to an artificial neural network, because they have excellent skills to discover hidden patterns yet under noisy conditions.

We have shown that a genetic algorithm can be used to assist the search for those feature that will produce the desired results. A GA individual is a subset of features built from the main set of features. Each individual may be evaluated using the mean squared error obtained during the training of an artificial neural network. The mean squared error can be, then, used to select those more fitted individuals to create a new generation of individuals. The experimental results show that the proposed method allows extracting those features that are required to get a desired output. The experimental results were performed using uncorrelated and correlated features. Last, note that artificial neural network

training is time consuming, and it would interesting to incorporate other artificial intelligence techniques to reduce the total processing time of this algorithm.

References

1. Alizadeh, A.A.: Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature 403, 503–511 (2000)
2. Bonev, B., Escolano, F., Carzola, M.A.: A Novel Information Theory Method for Filter Feature Selection. In: Gelbukh, A., Kuri Morales, Á.F. (eds.) MICAI 2007. LNCS (LNAI), vol. 4827, pp. 431–440. Springer, Heidelberg (2007)
3. Ding, C., Peng, H.: Minimum redundancy feature selection from microarray gene expression data. Journal of Bioinformatics and Computational Biology 3(2), 185–205 (2005)
4. Garber, M.E., Troyanskaya, O.G., et al.: Diversity of gene expression in adenocarcinoma of the lung. PNAS USA 98(24), 13784–13789 (2001)
5. Jones, M.T.: AI Application Programming, Charles River Media, 2nd edn., pp. 229–261 (2005)
6. Masters, T.: Practical Neural Network Recipes in C++, pp. 135–164. Academic Press, Inc., London (1993)
7. Peng, H., Ding, C., Long, F.: Minimum redundancy maximum relevance feature selection. IEEE Intelligent Systems 20(6), 70–71 (2005)
8. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(8), 1226–1238 (2005)
9. Reed, R.D., Marks II, R.J.: Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, pp. 185–195. The MIT Press, Cambridge (1999)
10. Russel, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, pp. 109–134. Prentice-Hall of India, Englewood Cliffs (2006)