A Neural Evolutionary Classification Method for Brain-Wave Analysis

Antonia Azzini and Andrea G.B. Tettamanzi

Università degli Studi di Milano,
Dipartimento di Tecnologie dell'Informazione,
via Bramante 65, I-26013, Crema, Italy
{azzini, tettamanzi}@dti.unimi.it

Abstract. This paper presents an approach to the joint optimization of neural network structure and weights which can take advantage of backpropagation as a specialized decoder. The approach is applied to binary classification of brain waves in the context of brain-computer interfaces.

Keywords: Neural Networks, Classification, Brain-Computer Interfaces, Evolutionary Algorithms.

1 Introduction

The evolutionary approach that implements the conjunction of evolutionary algorithms (EAs) with neural networks (NNs) is a more integrated way of designing ANNs since it allows all aspects of NN design to be taken into account at once and does not require expert knowledge of the problem. Some EAs have implemented a search over the topology space, or a search for the optimal learning parameters or weight setting.

The primary motivation for using evolutionary techniques to establish the weighting values rather than traditional gradient descent techniques such as backpropagation (BP) [5], lies in the trapping in local minima and in the non-differentiability of the function. For this reason, rather than adapting weights based on local improvement only, EAs evolve weights based on the whole network fitness. An interesting area of evolutionary NNs is the combination of architecture and weight evolution in order to find an optimal network architecture and to train the network on a given data set. The advantage of combining these two basic elements of a NN is that a completely functioning network can be evolved without any intervention by an expert.

2 The Neuro-genetic Approach

The approach is designed to be able to take advantage of the backpropagation (BP) algorithm if that is possible and beneficial; however, it can also do without it. This research was tested by an industrial application [1] for the design of

F. Rothlauf et al. (Eds.): EvoWorkshops 2006, LNCS 3907, pp. 500–504, 2006.

neural engine controllers, with particular attention to reduced power consumption and silicon area. In this work we apply our neuro-genetic approach to brain wave signal processing, in particular as a classification algorithm in the analysis of P300 Evoked Potential. We restrict our attention to Multi-Layer Perceptrons (MLPs), a specific subset of the feedforwards NNs which is powerful enough to be general, while at the same time allowing for a compact representation.

2.1 The Evolutionary Representation

The initial population is seeded with random networks initialized with different hidden layer sizes, using two exponential distributions to determine the number of hidden layers and neurons for each individual, and a normal distribution to determine the weights and bias values. For all weights matrices and bias we also define matrices of variance, that will be applied in conjunction with evolutionary strategies in order to perturb network weights and bias. Variance matrices will be initialized with matrices of all ones. In both cases, unlike other approaches like [6], the maximum size and number of the hidden layers is neither pre-determined, nor bounded, even though the fitness function may penalize large networks. Each individual is encoded in a structure in which we maintain basic information about string codification of topology and weights and bias matrices. This kind of structure is defined together with all parameters algorithms in [1]. The values of all these parameters are affected by the genetic operators during evolution, in order to perform incremental (adding hidden neurons or hidden layers) and decremental (pruning hidden neurons or hidden layers) learning.

2.2 Fitness

Like indicated in [1] the fitness is proportional to the value of the mse and to the cost of the considered network. It is defined as

$$f = \lambda kc + (1 - \lambda)\text{mse}, \tag{1}$$

where $\lambda \in [0, 1]$ is a parameter which specifies the desired trade-off between network cost and accuracy, k is a constant for scaling the cost and the mse of the network to a comparable scale, and c is the overall cost of the considered network, defined as

$$c = \alpha N_{hn} + \beta N_{syn}, \tag{2}$$

where N_{hn} is the number of hidden neurons, and N_{syn} is the number of synapses. The mse depends on the *Activation Function*, that calculates all the output values for each single layer of the neural network. In this work we use the *Sigmoid Transfer Function*. To be more precise, two fitness values are actually calculated for each individual: the fitness f, used by the selection operator, and a test fitness $\hat{f}$. $\hat{f}$ is calculated according to Equation 1 by using the mse over the test set. When BP is used, i.e., if $bp = 1$, $f = \hat{f}$; otherwise ($bp = 0$), f is calculated according to Equation 1 by using the mse over the training and test sets together.

2.3 Genetic Operators

The genetic core of the algorithm is described by the following pseudo-code:

1. Select from the population (of size n) $\lfloor n/2 \rfloor$ individuals by truncation and create a new population of size n with copies of the selected individuals.
2. For all individuals in the population:
   (a) Mutate the weights and the topology of the offspring.
   (b) Train the resulting network using the training and test sets if $bp = 1$.
   (c) Calculate f and $\hat{f}$.
   (d) Save the individual with lowest $\hat{f}$ as the best-so-far individual if the $\hat{f}$ of the previously saved best-so-far individual is higher (worse).
3. Save statistics.

The *selection* strategy used by the algorithm is truncation: starting from a population of n individuals, the worst $\lfloor n/2 \rfloor$ (with respect to f) are eliminated. The remaining individuals are duplicated in order to replace those eliminated. Finally, the population is randomly permuted. Two types of *mutation* operators are used: a general random perturbation of weights, applied before the BP learning rule, and three *mutation* operators which affect the network architecture. The weight mutation is applied first, followed by the topology mutations, as follows:

1. Weight mutation: all the weight matrices and the biases are perturbed by using equations based on variance matrices and evolutionary strategies applied to the number of synapses of the entire neural network. These equations are described in detail in [1]. After this perturbation has been applied, neurons whose contribution to the network output is negligible are eliminated: a variable threshold is defined, depending on a norm (in our case L_∞) of the weight vector for each node, and the relevant average and standard deviation of the norms of the considered layer.
2. Topology mutations: these operators affect the network structure (i.e., the number of neurons in each layer and the number of hidden layers). In particular, three mutations can occur, which are described in detail in [1]:
   (a) Insertion of one hidden layer with probability p^+_{layer};
   (b) Deletion of one hidden layer with probability p^-_{layer};
   (c) Insertion of a neuron with probability p^-_{layer}.

All three topology mutation operators are designed so as to minimize their impact on the behavior of the network; in other words, they are designed to be as little disruptive (and as much neutral) as possible.

3 Application to Brain-Wave Analysis

3.1 Brain-Computer Interfaces and Problem Description

Brain Computer Interfaces (BCI) represent a new communication option for those suffering from neuromuscular impairment that prevents them from using

conventional input devices, such as mouses, keyboards, joysticks, etc. Exploiting the residual functions of the brain, BCI may allow those patients to communicate. One of the most utilized electrical activities of the brain for BCI is the P300 Evoked Potential wave. This wave is a late-appearing component of an Event Related Potential (ERP) which can be auditory, visual or somatosensory. The idea of Donchin's solution [3] is that the patient is able to generate this signal without any training. This is due to the fact that the P300 is the brain's response to an unexpected or surprising event and is generated naturally. Donchin developed a BCI system able to detect an elicited P300 by signal averaging techniques (to reduce the noise) and used a specific method to speed up the overall performance. Donchin's idea has been adopted and further developed by Beverina and colleagues of ST Microelectronics [2]. We have applied the neuro-genetic approach described in Section 2 to the same dataset of P300 evoked potential used by Beverina and colleagues for their approach on brain signal analysis based on support vector machines.

3.2 Experimental Protocol and Results

The dataset provided by Beverina and colleagues consists of 700 negative cases and 295 positive cases. The feature are based on wavelets, morphological criteria and power in different time windows, for a total of 78 real-valued input attributes and 1 binary output attribute, indicating the class (positive or negative) of the relevant case. In order to create a balanced dataset of the same cardinality as the one used by Beverina and colleagues, for each run of the evolutionary algorithm we extract 218 positive cases from the 295 positive cases of the original set, and 218 negative cases from the 700 negative cases of the original set, to create a 436 case training dataset; for each run, we also create a 40 case test set by randomly extracting 20 positive cases and 20 negative cases from the remainder of the original dataset, so that there is no overlap between the training and the test sets. This is the same protocol followed by Beverina and colleagues. For each run of the evolutionary algorithm we allow up to 25,000 network evaluations (i.e., simulations of the network on the whole training set), including those performed by the backpropagation algorithm. 100 runs of the neuro-genetic approach with parameters set to their defaults values were executed with $bp = 0$ and $bp = 1$, i.e., both without and with backpropagation. The results obtained are presented in Table 1.

Table 1. Error rates of the best solutions found by the neuro-genetic approach with and without the use of backpropagation, averaged over 100 runs.

bp	training				test			
	false positives		false negatives		false positives		false negatives	
	avg	stdev	avg	stdev	avg	stdev	avg	stdev
0	93.28	38.668	86.14	38.289	7.62	3.9817	7.39	3.9026
1	29.42	14.329	36.47	12.716	1.96	1.4697	2.07	1.4924

Due to the way the training set and the test set are used, it is not surprising that error rates on the test sets look better than error rates on the training sets. That happens because, in the case of $bp = 1$, the performance of a network on the test set is used to calculate its fitness, which is used by the evolutionary algorithm to perform selection. Therefore, it is only networks whose performance on the test set is better than average which are selected for reproduction. The best solution has been found by the algorithm using backpropagation and is a multi-layer perceptron with one hidden layer with 4 neurons, which gives 22 false positives and 29 false negatives on the training set, while it commits no classification error on the test set. The results obtained by the neuro-genetic approach, without any specific tuning of the parameters, appear promising. To provide a reference, the average number of false positives obtained by Beverina and colleagues with support vector machines are 9.62 on the training set and 3.26 on the test set, whereas the number of false negatives are 21.34 on the training set and 4.45 on the test set [4].

4 Conclusion and Future Works

We illustrated an evolutionary approach to the joint design of neural network structure and weights which can take advantage of BP as a specialized decoder. The approach has been applied to the analysis of brain waves and compared to a mature approach based on support vector machines which has been presented in [2]. The comparison shows that our approach has some potential, even though, unsurprisingly, it does not attain the same levels of accuracy. Further work on this problem will include an in-depth study for parameters tuning and data set up, in order to improve the accuracy of classification.

References

1. A. Azzini, M. Lazzaroni, and G.B. Tettamanzi. A neuro-genetic approach to neural network design. In F. Sartori, S. Manzoni, and M. Palmonari, editors, *AI*IA 2005 Workshop on Evolutionary Computation*. AI*IA, Italian Association for Artificial Intelligence, September 20 2005.
2. F. Beverina, G. Palmas, S.Silvoni, F. Piccione, and S. Giove. User adaptive bcis: Ssvep and p300 based interfaces. *PsychNology Journal*, 1(4):331–354, 2003.
3. E. Donchin, K.M. Spencer, and R. Wijesinghe. The mental prosthesis: assessing the speed of a p300-based brain-computer interface. *IEEE Transactions on Rehabilitation Engineering*, 8(2):174–179, June 2000.
4. Giorgio Palmas. Personal communication, November 2005.
5. D. E. Rumelhart, J. L. McClelland, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
6. X. Yao and Y.Liu. Towards designing artificial neural networks by evolution. *Applied Mathematics and Computation*, 91(1):83–90, 1998.