A First Attempt at Constructing Genetic Programming Expressions for EEG Classification

César Estébanez, José M. Valls, Ricardo Aler, and Inés M. Galván

Universidad Carlos III de Madrid,
Avda. de la Universidad, 30, 28911, Leganés (Madrid), Spain
{cesteban, jvalls, aler, igalvan}@inf.uc3m.es

Abstract. In BCI (Brain Computer Interface) research, the classification of EEG signals is a domain where raw data has to undergo some preprocessing, so that the right attributes for classification are obtained. Several transformational techniques have been used for this purpose: Principal Component Analysis, the Adaptive Autoregressive Model, FFT or Wavelet Transforms, etc. However, it would be useful to automatically build significant attributes appropriate for each particular problem. In this paper, we use Genetic Programming to evolve projections that translate EEG data into a new vectorial space (coordinates of this space being the new attributes), where projected data can be more easily classified. Although our method is applied here in a straightforward way to check for feasibility, it has achieved reasonable classification results that are comparable to those obtained by other state of the art algorithms. In the future, we expect that by choosing carefully primitive functions, Genetic Programming will be able to give original results that cannot be matched by other machine learning classification algorithms.

1 Introduction

Within the Machine Learning field, there are many domains where the main difficulty is not to determine the proper algorithm to be applied, or even selecting the most relevant attributes. In these cases, the attributes available in raw data are not the most significant for classification, and new attributes have to be constructed. This is usually called feature induction or constructive induction [1, 2, 3]. The brain computer interface (BCI) is a domain where raw data has to undergo some preprocessing, so that the right attributes for classification are obtained. In BCI research, several transformational techniques have been used to obtain high classification accuracy (over 90%): Principal Component Analysis [4], the Adaptive Autoregressive Model [5], FFT or Wavelet Transforms [6], etc. Intuitions, empirical results, and knowledge about the domain is what lead researchers towards using these methods. However, it would be useful to automatically build significant attributes appropriate for every kind of signal and classification task.

Genetic Programming (GP) is an evolutionary technique for evolving symbolic programs [7]. Most research has focused in evolving functional expressions,

but using loops and recursion have also been considered [8]. Evolving circuits are also among the successes of GP [9]. In this paper, we intend to use GP to evolve expressions that project original data, which is expressed in coordinates of a space with N dimensions, to a new space with M dimensions (where $M << N$). Here, we consider two-class classification problems. We expect that by finding the right projection, it will be possible to separate data linearly (approximately) in the projected space.. The secondary goal is to project data into a smaller space: This way, the dimensionality of the problem is reduced and a new set of attributes is obtained. This new emergent-attributes represent the relevant information needed for classification. They also can reveal non-observable relationships between the original attributes, improving the understanding of the problem. Fitness of each projection is determined by computing the degree of linear separation of data in the projected space. This has been implemented as a linear perceptron. Our motivation for using GP is that the set of primitives GP uses for building hypothesis is a parameter of the algorithm. Other machine learning algorithms work with pre-defined primitives like node-comparisons in ID3 or neurons in ANN. GP allows to include the most relevant primitives for the domain, although this selection requires sometimes a long trial-and-error process.

The final aim of our research is to obtain results that cannot be obtained by directly applying other machine learning methods. This paper is a first shot at using GP for this purpose. Therefore, results obtained in this paper correspond to the most straightforward and simple approach, that will test the feasibility of GP, and can be used as a base to compare (and improve) future research.

This paper is organized as follows. Section 2 introduces Genetic Programming. Section 3 describes our approach. Section 4 reports the experiments carried out. And finally, Section 5 draws the main results of the paper and prepares for future research.

2 Genetic Programming

Genetic Programming (GP) is an evolutionary technique designed to generate programs automatically [7]. It has three main elements:

- A population of individuals. In this case, the individuals are computer programs.
- A fitness function. It is used to measure the goodness of the computer program represented by the individual.
- A set of genetic operators. In GP, the basic operators are reproduction, mutation, and crossover.

The GP algorithm enters into a cycle of fitness evaluation and genetic operator application, producing consecutive generations of populations of computer programs, until a good enough individual is found. Every genetic operator has a probability of being applied each time we need to generate an offspring individual for the next generation. Also, GP has many other parameters, the most

important ones being the size of the population (M) and the maximum number of generations (G).

3 The Approach

We will learn from a set E of n examples expressed in a space U of N dimensions. Our objective is to be able to represent the examples in the space V, of M dimensions, and in which the examples will be linearly separable.

As we already have seen, Our method have two different applications: on one hand, the improvement of classification tasks by means of a transformation of the dataset; on the other hand, the reduction of dimensionality by constructing new attributes that are as good, at least, as the original ones.

Our method uses standard GP to evolve individuals made of M subtrees (as many of dimensions of the projected space V). Then, data is projected from U to V by means of applying the individual to the original data. Fitness is computed by measuring the degree of linear separation after the projection. The system stops if a 100% linear separation has been achieved or if the maximum number of generations is reached. Otherwise, the system outputs the individual that separated better the training data.

For the implementation of our application, we have used Lilgp 1.1, the software package for Genetic Programming developed in Michigan State University by Douglas Zongker and Bill Punch, members of the group GARAGe (Genetic Algorithms Research and Applications Group) (http://garage.cse.msu.edu/).

Terminal and Function Set

In our problem, terminal set will be formed by the attributes of the problem expressed in coordinates of U (u_0,u_1...,u_N), and by Ephemeral Random Constants [7].

The set of functions to use is difficult to determine: it must be sufficient for, along with the set of terminals, being able to express the solution to the problem, but they must not be too many as for uselessly increase the search space. Of course, for different domains, different terminal and function sets will be more appropriate. In this case, we have tried with just arithmetic functions $(+, -, *, /)$.

In the future, we would like to explore better grounded, domain-oriented sets of functions, like FFT or wavelet transforms.

GP Individuals

Instead of having individuals work with vectorial data and return a vector of M dimensions, every individual will contain M subtrees, using the same set of functions and terminals, that will be ran independently. Thus, a projection consists of a series of trees labelled (v_0,v_1...,v_M) that represent combinations of all the terminals (u_0,u_1...,u_N) and functions.

Genetic Operators

In our approach, we use the three genetic operators typically used in GP. Reproduction chooses an individual and copies it verbatim into the new population;

Mutation chooses an individual and a subtree, then the subtree is deleted and replaced with a randomly generated subtree; and Crossover selects two individuals, a subtree is selected on each individual, and the subtrees are swapped.

The Fitness Function
We already have introduced the basic mechanism of the fitness function. It takes the examples expressed in space U, project them using the GP individual, and obtain the examples expressed in space V of M dimensions. Next, a classification algorithm is applied to the projected data. In this case, we have choosen to apply a Simple linear Perceptron. The Perceptron is run for 500 cycles (experimentally we have checked that this is more than enough). If the SP converges, the projection would be producing a linear separation of the data and it would be the solution to the problem. If the SP does not converge, the fitness assigned to the individual is the number of examples that the SP has been able to correctly classify in the best cycle. We choose the punctuation of the best cycle because if projected data is not linearly separable, the SP will oscillate. Storing the best value guarantees stability of the fitness value. This way, fitness is gradual enough and has the resolution necessary to be able to exert a real selective pressure.

If the method does not find a linear (or nearly linear) classification of the examples in the space V, we can change the number of dimensions desired for V and launch the method again. As we already have stated, our main goal is to find a linear separation of examples and, in order to achieve this, it is possible for our application to increase M to values greater than N. However, in BCI research, the original number of dimensions N is usually very large, so it does not make sense to increase them in the new space V. This is the reason for not documenting in this paper the possibility of a M value greater than N.

4 Experiments

This section describes our first experiments using the NIPS 2001 Brain Computer Interface Workshop dataset [10].[1] This dataset was recorded from a normal subject during a no-feedback session. The subject sat in a normal chair, fingers in the standard typing position at the computer keyboard. The task of the subject was to press with the index and little fingers the corresponding keys in a self-chosen order and timing.

The classification task is to create a classifier to predict which of the two fingers the user intends to use (before pressing the key).

For validation purposes, we have divided the dataset into a training and a test set, containing 412 (80%) and 102 (20%) instances respectively. Due to the long runs required by GP, crossvalidation was not feasible. We took the raw data and selected the 20 last instants of every channel. Allegedly, these instants are more relevant for the classification because they are closer to the time the person makes the decision. Therefore, as there are 27 channels, the initial space has $N = 27 * 20 = 540$ dimensions. The goal is to project this data to a space

[1] http://liinc.bme.columbia.edu/competition.htm

Table 1. Summary of experiments carried out

Experiment	Population size	Generations	Max Nodes
GP1	1000	500	200
GP2	3000	500	70
GP3	3000	500	40

with $M = 3$ dimensions where data can be classified linearly. Table 1 summarizes the 3 experiments carried out in this paper. It displays details about the population size, the maximum number of generations, and the maximum size allowed for the evolved expression (the latter is meant to limit overfitting).

Table 2. Summary of best/median test results for experiments GP1, GP2, and GP3

Experiments	% Test
GP1 (best)	87.5/83.175
GP2 (best)	94.2/86.54
GP3 (best)	94.2/87.5
SMO	93.3
Simple logistics	96.2
ANN	95.1

For every experiment in Table 1, the GP system was run 6 times. The best and median results for every experiment on the test set are summarized in Table 2. Other machine learning approaches have also been tested on the same data. SMO (support vector machine) and Simple Logistic Regression come from the Weka package, using default parameters. They both performed very well in relation to other Weka algorithms. ANN is a backpropagation neural network.

In summary, the best GP individual (94.2%) obtains comparable results to the other systems tested. This result is positive, considering that this is preliminary research, and that we have obtained an expression that projects from 540 to 3 dimensions, where almost linear classification can be carried out.

5 Conclusions

In this paper, we have applied Genetic Programming to evolve functions that project data from spaces with N dimensions to spaces with M dimensions. The main goal is that in the M-space, projected data can be classified linearly (or close to). The secondary goal is to project data into a smaller space, so $M << N$. This approach has been applied to classification of EEG data coming from a Brain Computer Interface competition.

Here, we have applied a straightforward configuration, where the primitives used by GP are arithmetic functions. Results are comparable or inferior to other

machine learning methods, but the dimensionality of the problem has been reduced from 540 to only 3. Yet, even for this simple approach and for the important reduction of the problem, we have obtained a projection that classifies linearly the projected data with an accuracy of 94.2%. This shows that the approach has some merit, although our goal of evolving original expressions, better than what can be achieved by traditional machine learning methods, or by human expertise, has not been achieved. In the future, we expect that by carefully choosing primitive functions, known to perform well in the EEG domain, results will be much improved.

Acknowledgements

This article has been financed by the Spanish founded research MCyT project TRACER, Ref:TIC2002-04498-C05-04M.

References

1. Fawcett, T., Utgoff, P.: A hybrid method for feature generation. In: Proceedings of the Eighth International Workshop on Machine Learning, Evanston, IL (1991) 137–141
2. Kramer, S.: Cn2-mci: A two-step method for constructive induction. In: Proceedings of ML-COLT'94. (1994)
3. Pfahringer, B.: Cipf 2.0: A robust constructive induction system. In: Proceedings of ML-COLT'94. (1994)
4. Hoya, T., Hori, G., Bakardjian, H., Nishimura, T., Suzuki, T., Miyawaki, Y., Funase, A., Cao, J.: Classification of single trial eeg signals by a combined principal + independent component analysis and probabilistic neural network approach. In: Proceedings of the fourth International Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003). (2003) 197–202
5. Schlogl, A.: "The Electroencephalogram and the Adaptive Autoregressive Model: theory and applications". Shaker Verlag (2000)
6. Felzer, T.: On the possibility of developing a brain-computer interface (bci). Technical report, Technical University of Darmstadt, Germany (2001)
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
8. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge Massachusetts (1994)
9. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers (2003)
10. Blankertz, B., Curio, G., Müller, K.R.: Classifying single trial eeg: Towards brain computer interfacing. In: Advances in Neural Inf. Proc. Systems 14 (NIPS 01). (2002)