# Input Variable Generation for Long Term Prediction of EEG Signal Using Genetic Programming

Hossein Soleimani-B.
Control and Intelligent Processing Center of Excellence, Electrical and Computer Eng. Department, University of Tehran, Tehran, Iran
e-mail: h.soleimani@ece.ut.ac.ir

Caro Lucas
Control and Intelligent Processing Center of Excellence, Electrical and Computer Eng. Department, University of Tehran, Tehran, Iran
e-mail: lucas@ipm.ir

## ABSTRACT

Complexity and accuracy of prediction are highly dependent to input variables of the model. Different methods of input variable selection based on correlation analysis and mutual information have been widely used to select minimum number of input variables. Selection of inputs is in fact a linear transformation on available input variables. In this paper, nonlinear transformation is used to generate input variables for the model. Inputs are generated in a way to have maximum dependency to the desired output. These nonlinear transformations are created using genetic programming. Fitness function of genetic programming is defined based on the mutual information between generated input variables and the desired output. Definition of fitness function is done in a way to increase the rate of convergence of the algorithm. Simulations prove the vantage of the proposed method in comparison with the input selection algorithm. Prediction of EEG with the generated input variables leads to less error in comparison with selected inputs.

## Keywords

Input Variable Generation, Mutual Information, Genetic Programming, Long Term Prediction, EEG

## 1. INTRODUCTION

Epilepsy is one of the rifest neurological disorders. It is estimated that approximately one percent of the world population are affected by epilepsy and also 25 percent of this population are not cured with known treatments. Seizures can be aborted by targeted therapy, if an accurate prediction of its occurrence time is available. Seizure prediction consists of two stages, prediction of EEG signal and detection of seizure occurrence. The aim of this paper is to improve the accuracy of EEG prediction, used in seizure prediction.

Input variable selection is one of the most important stages in modeling and prediction. In most of the modeling problems, there are lots of input variables, available. It is obvious that the importance of inputs is different and some of them may not have a significant relevance to the output of the system. In all modeling and prediction problems, models with the least possible level of complexity and minimum number of input variables are preferable.

In order to have a model with minimum level of complexity, there should be a procedure to select most appropriate input variables with less redundancy. Input variable selection should be based on a measure that shows the relevance of each input to the desired output. Correlation analysis is a method that measures the linear relevance between two signals. Since nonlinear dependence of signals is not considered in correlation analysis it is unable to select best inputs. Better selection of input variables is possible using more sophisticated methods such as mutual information. Mutual information has been widely used to select features and input variables in classification and modeling applications [1]-[5]. B. Atoufi and C. Lucas in [6] showed the effectiveness of input variable selection based on mutual information in comparison with correlation analysis.

These methods only select most appropriate input variables between available variables that in prediction applications are the lagged values of time series.

Selection of input variables from a set of available inputs is clearly a special kind of linear transformation. An extension to this process is to use nonlinear transformation to generate inputs from the set of available variables. Input variables generated by the nonlinear transformation can lead to more accurate predictions and models with less complexity. This nonlinear many to one transformation can be found using genetic programming. In this paper the local linear neuro fuzzy model [7] is used as the model for EEG prediction.

This paper is organized as follows. Section 2 describes mutual information and its estimation method. Simple method of input variable selection is also introduced in this section. Section 3 introduces genetic programming and its application in this paper. Main aspects of local linear neuro fuzzy modeling are discussed in section 4. Section 5 presents the simulation results and finally the concluding remarks are discussed in section 6.

## 2. Mutual Information

### 2.1. Definition of Mutual Information

Mutual information is defined based on the Shannon Entropy [8] to measure the dependency of two random signals.

$$\begin{aligned} I(X,Y) &= H(X) - H(X \mid Y) = H(Y) - H(Y \mid X) \\ &= H(X) + H(Y) - H(X;Y) \end{aligned} \tag{1}$$

where *H(X), H(X|Y)* and *H(X;Y)* are the entropy, conditional entropy and joint entropy of *X* and *Y*, respectively. According to [8] entropy and joint entropy are defined as follows:

$$H(X) = -\int_x P_X(x) \log P_X(x) dx \tag{2}$$

$$H(Y) = -\int_y P_Y(y) \log P_Y(y) dy \tag{3}$$

$$H(X;Y) = -\int_x \int_y P_{X,Y}(x,y) \log P_{X,Y}(x,y) dxdy \tag{4}$$

where $P_{X,Y}(x,y)$ and $P_X(x)$ and $P_Y(y)$ are the joint probability density function and marginal probability density functions of *X* and *Y*, respectively. By substituting (2), (3) and (4) into (1) mutual information of *X* and *Y* is obtained.

$$I(X;Y) = \int_x \int_y P_{X,Y}(x,y) \log \frac{P_{X,Y}(x,y)}{P_X(x)P_Y(y)} dxdy \qquad (5)$$

## 2.2. Estimating Mutual Information

In order to compute mutual information, first the probability density functions should be estimated using PDF estimation methods [9], [10] and then mutual information can be computed based on the estimated PDFs. Other methods have been proposed that directly estimate the mutual information. In this paper, mutual information is estimated directly using the method proposed in [11], that estimate entropy based on the average distance to the k-nearest neighbors.

Assuming $z_i = (x_i, y_i)$ $i = 1, 2,.. ,N$ are $N$ data points of a random variable $Z = (X,Y)$ with the density function $P_{X,Y}(x,y)$ where $X$ and $Y$ are random vectors in $R^n$. Distance of each data point, $z_i$, to other points is computed using maximum norm.

$$\left\| z_i - z_j \right\|_\infty = \max\{\left\| x_i - x_j \right\|_2, \left\| y_i - y_j \right\|_2\}, \quad j = 1,2,...,N \qquad (6)$$

$z_{k(i)} = (x_{k(i)}, y_{k(i)})$ is defined as the k-th nearest neighbor of $z_i$ with respect to the maximum norm, where k is a fixed positive integer. Distance of each data point from its k-th nearest neighbor is defined as follows:

$$\varepsilon_i / 2 = \left\| z_i - z_{k(i)} \right\|_\infty \qquad (7)$$

$$\varepsilon^x{}_i / 2 = \left\| x_i - x_{k(i)} \right\|_2, \quad \varepsilon^y{}_i / 2 = \left\| y_i - y_{k(i)} \right\|_2 \qquad (8)$$

$n_i{}^x$ and $n_i{}^y$ are the number of data points with $\left\| x_i - x_j \right\|_2 \le \varepsilon_i^x / 2$ and $\left\| y_i - y_j \right\|_2 \le \varepsilon_i^y / 2$. Finally the mutual information of X and Y is estimated.

$$\hat{I}(X;Y) = \psi(k) - \frac{1}{k} - \frac{1}{N} \sum_{i=1}^{N} [\psi(n_i^x) + \psi(n_i^y)] + \psi(N) \qquad (9)$$

where $\psi$ is the Digamma function.

In order to have a correct estimation, k should be selected carefully. Using a small value for *k*, the estimator will have a large variance and a small bias, while large values of *k* lead to a small variance and a large bias [12]. In this paper the value of *k* is selected to be 6.

## 2.3. Input Variable Selection

Battiti in [1] proposed an algorithm to select most relevant input variables among available features. The aim of this algorithm is to select a set of input variables that have maximum relevance to the output and minimum redundancy among each other. This algorithm is as follows:

1) *Initialization:* Set L to 'initial set of n inputs' and S to 'empty set'. T is the output of the model.
2) *Computation of the mutual information with the output:* Compute *I(T;l)* for each input *l ϵ L.*
3) *Choice of the first input:* Find the input *l* that maximize *I(T;l).* Set $L \leftarrow L - \{l\}$ and $S \leftarrow \{l\}$
4) *Greedy selection:* Repeat until desired number of input variables are selected:
   a. *Computation of mutual information between variables:* For all couple of variables *l ϵ L* and *s ϵ S* compute *I(l;s).*
   b. *Selection of the next input:* choose the input *l* that maximize $I(T;l) - \frac{\beta}{|S|} \sum_{s \in S} I(l;s)$; Set $L \leftarrow L - \{l\}$ and $S \leftarrow S \bigcup \{l\}$
5) Output the set of S containing the selected input variables.

Redundancy between selected input variables can be controlled with β. If β decreases, the importance of dependency between selected inputs decreases.

## 3. Genetic Programming

Genetic programming (GP) is an evolutionary algorithm that searches for optimum structure of a system [13]. GP evolves systems that are represented by tree structures. Every tree consists of terminal nodes that are input variables of the system and functional nodes. A useful toolbox of genetic programming has been developed in [14].

GP, similar to genetic algorithm has genetic operators to build new individuals. Crossover, mutation and swap mutation are some of genetic operators that are used in GP. Crossover operator randomly selects two subtrees from parents and swaps them to create two children. Mutation operator substitutes a randomly selected subtree of an individual with a new randomly created node-branch. In swap mutation two random subtrees are chosen from an individual and swapped.

Selecting correct probabilities of genetic operators is an important concern in evolutionary algorithms. Mutation is the ability of the algorithm to explore new points in search space while crossover operator helps the algorithm to exploit points with better fitness function based on individuals in the population. The evolutionary algorithm can converge to global optimum point if there is a good balance between exploration and exploitation ability. In this paper crossover and mutation probabilities are tuned adaptively during the algorithm based on the method proposed in [15]. Operator probabilities are adjusted based on the performance and their effectiveness in generating individuals with better fitness. A credit is assigned to each offspring based on its fitness in comparison with the best and worst fitness of the population of its parents. Probability of each operator is computed as the average of credits of individuals that are created using that operator.

In genetic programming the complexity of individuals may grow without any significant improvement in fitness function. This phenomenon is known as bloat. The simplest method of avoiding bloat is to set a maximum depth size on trees [13]. Bloat can also be avoided by setting a measure of complexity into the fitness of each individual [16]. Using this method, the fitness of the individual is decreased with the growth of its depth. Using dynamic maximum tree depth instead of fixed limit is another way that effectively controls bloat [17]. This method actually has two kinds of limit on the depth size; dynamic and static limits. In this method, depth of each individual can grow if there is an improvement in fitness function until the depth reaches the static limit. When the depth of individuals reaches to the static limit, they cannot grow any further. In this paper dynamic depth size method is used. Dynamic and static limits on depth size are set to 10 and 30, respectively.

In this paper genetic programming is used to find a nonlinear transformation from available input variables to create a set of inputs that have maximum relevance to the desired output. The procedure described in part 3 of section 2 is done while in each iteration**,** an input variable is created using genetic programming. Fitness function for genetic programming can be defined as follows.

$$F = I(T;l) - \frac{1}{|S|} \sum_{s \in S} I(l;s) \qquad (10)$$

while *l* is the created feature and s is the set of previously generated input variables. In order to increase the rate of convergence of the algorithm and creating trees with a predefined minimum level of complexity the fitness function (10) is revised as follows.

$$F = I(T;l) - \frac{1}{|S|}\sum_{s\in S} I(l;s) + \frac{\min(d,10)}{10} \qquad (11)$$

where *d* is the depth of the individual. It is obvious that fitness value is increased with increasing the depth size of individuals. Using this fitness function, the algorithm will converge quickly to trees with at least depth size equals to 10. Increasing depth size of trees to more than 10 does not have any direct effect in fitness function. After guiding the algorithm to generate trees with a minimum level of complexity it naturally selects trees that lead to greater mutual information. In fact, this kind of definition of fitness function increases the rate of convergence of the algorithm.

Selection of termination criterion is another important part of genetic programming. Generated input variables are used in neurofuzzy networks that should have a good level of generalization. In other words networks that are trained with created input variables should have a good performance for both training and test datasets. So the nonlinear transformation that is created using GP should lead to input variables with high dependence between input and output variables for both test and training datasets. In this algorithm stop criterion is selected such that the nonlinear function resulted from the evolution have a good generalization for the test dataset. Every decrement in fitness function for test data should be compensated in next 10 iterations and if it does not decrease to a level less than its value before the increment, the algorithm will be terminated.

## 4. Neuro Fuzzy Modeling

The local linear model tree (LoLiMoT) proposed in [7] is a neuro fuzzy model with one hidden layer and a linear neuron in the output layer. Neurons in hidden layer divide the input space into small linear subspaces. The output of the locally linear neurons in hidden layer is a weighted sum of input variables, multiplied by the normalized value of validity function.

$$\hat{y}_i = \omega_{i_0} + \omega_{i_1} u_1 + \omega_{i_2} u_2 + \dots + \omega_{i_p} u_p \qquad (12)$$

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i \phi_i(u) \qquad (13)$$

where u = [$u_1$, $u_2$, …, $u_p$] is the input of the network and M is the number of neurons in hidden layer. The validity functions are typically selected as normalized Gaussians.

$$\phi_i(u) = \mu_i(u) \Big/ \sum_{j=1}^{M} \mu_j(u) \qquad (14)$$

$$\mu_i(u) = \exp\{-\frac{1}{2}(\frac{(u_1 - c_{i_1})^2}{\sigma_{i_p}^2} + \dots + \frac{(u_p - c_{i_p})^2}{\sigma_{i_p}^2})\} \qquad (15)$$

The fundamental approach with the locally linear neuro fuzzy model is dividing the input space into small linear subspaces with fuzzy validity functions.

All linear parameters of the model are estimated using least squares optimization. The parameter vector consist of M(p+1) elements

$$\omega = [\omega_{1_0}\omega_{1_1}\dots\omega_{1_p}\omega_{2_0}\omega_{2_1}\dots\omega_{2_p}\dots\omega_{M_0}\omega_{M_1}\dots\omega_{M_p}]^T \qquad (16)$$

Regression matrix *X* for *N* samples is:

$$X = [X_1 \quad X_2 \quad \dots \quad X_N]$$

$$X_i = \begin{pmatrix} \phi_i(\underline{u}(1)) & u_1(1)\phi_i(\underline{u}(1)) & \cdots & u_p(1)\phi_i(\underline{u}(1)) \\ \phi_i(\underline{u}(2)) & u_1(2)\phi_i(\underline{u}(2)) & \cdots & u_p(2)\phi_i(\underline{u}(2)) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_i(\underline{u}(N)) & u_1(N)\phi_i(\underline{u}(N)) & \cdots & u_p(N)\phi_i(\underline{u}(N)) \end{pmatrix} \qquad (17)$$

Using least square optimization, the estimation of parameter vector is obtained as follows:

$$\hat{\omega} = (X^T X)^{-1} X^T y \qquad (18)$$

One of the most important vantages of this model is that the number of local linear models (LLM) in hidden layer is selected during the training process. In each iteration the worst local linear model is selected to be divided to two LLMs. All possible divisions in p dimensional input space are evaluated and the best division is selected. After creation of new LLM the fuzzy validity functions of the network should be updated. Centers are set to centers of new hyper-rectangles and standard deviations are set to a proportion of the size of hyper-rectangles as follows.

$$\sigma_{ij} = \frac{1}{3}\Delta_{ij} \qquad (19)$$

where $\Delta_{ij}$ is the extension of hyper-rectangle of i-th LLM in dimension $u_j$.

Creation of new LLMs is continued until a termination criterion is met. Termination criterion is defined in a way to prevent the network to be over-trained. Overtraining is a situation where the trained network becomes an expert for training data and the index of error for test data increases. Termination criterion is defined in a similar way of stop condition in GP that was described in section 3.

Index of error in this paper is normalized mean square of error (NMSE) that is defined as follows.

$$\text{NMSE} = \frac{\sum_{i=1}^{n}(y-\hat{y})^2}{\sum_{i=1}^{n}(y-\bar{y})^2} \qquad (20)$$

where $y$, $\hat{y}$ and $\bar{y}$ are observed, predicted and average of observed data points, respectively.

## 5. Simulation Results

The effectiveness of the proposed method is investigated in prediction of EEG signal. In this paper the dataset for 3rd international workshop on epileptic seizure prediction is used. EEG signals are available from [18]. First channel of EEG of second patient is selected to create the test and training datasets. 20 previous values of this time series are selected to be potential input variables to predict the value of time series in 1000 steps ahead. Prediction is done using three different methods.

In first method, all of 20 lags of time series are used as the input variables of the predictor. This method needs complex computations to design neuro fuzzy network.

In second method input variables are selected from the set of available inputs based on the method described in part 3 of section 2. Table I shows the selected input variable in each iteration and the NMSE on test and training datasets that is resulted using selected input variables. This table also shows the optimum number of LLMs before the network becomes overt-rained.

Table I. Results of feature selection method using mutual information

| Number of input variables | Input variable | NMSE of Training data | NMSE of Test data |
|---|---|---|---|
| 1 | 6 | 0.2159 | 0.2343 |
| 2 | 2 | 0.1882 | 0.1899 |
| 3 | 20 | 0.1841 | 0.1856 |
| 4 | 17 | 0.167 | 0.1771 |
| 5 | 11 | 0.1656 | 0.1777 |
| 6 | 16 | 0.1643 | 0.1759 |
| 7 | 8 | 0.1617 | 0.1851 |
| 8 | 5 | 0.1612 | 0.1846 |
| 9 | 1 | 0.1602 | 0.1849 |
| 10 | 7 | 0.1588 | 0.1859 |
| 11 | 14 | 0.1674 | 0.1859 |
| 12 | 12 | 0.1665 | 0.186 |
| 13 | 18 | 0.1745 | 0.1987 |
| 14 | 15 | 0.1507 | 0.1887 |
| 15 | 9 | 0.1576 | 0.1857 |
| 16 | 3 | 0.1571 | 0.1864 |
| 17 | 13 | 0.1568 | 0.1861 |
| 18 | 4 | 0.1621 | 0.1913 |
| 19 | 19 | 0.1646 | 0.1913 |
| 20 | 10 | 0.1559 | 0.1781 |

The process of input selection in this algorithm is terminated similar to the termination of LoLiMoT training. It is seen in table I that after selection of 6th input variable, the minimum NMSE for test data is achieved so this six input variables are selected as the final inputs.

In third method, genetic programming is used to generate input variables. Every available input variable is normalized to lie between -1 and 1. Population size of genetic programming is set to 100. Other parameters of genetic programming are set as described in section 3. Table II shows the results of applying this method.

Table II. Results of prediction with input variables generated using GP

| Number of input variables | NMSE of Test data |
|---|---|
| 1 | 0.1746 |
| 2 | 0.1647 |

Table II shows that using two input variables that are created using genetic programming leads to better results than using seven input variables that are selected based on mutual information.

Table III shows the result of comparison between three different methods used for EEG prediction.

Table III. Comparison of three methods for EEG prediction

| Method | Number of input variables | NMSE of Test data |
|---|---|---|
| Using all input variables | 20 | 0.1781 |
| Input selection based on MI | 6 | 0.1759 |
| Input generation using GP | 2 | 0.1647 |

Table III shows that as it is stated before, using all available input variables decreases the efficiency of the model. It is also shown that generating input variable using genetic programming performs better than the simple selection of input variables that is a special kind of linear transformation.

## 6. Conclusion

In this paper a new method of feature conditioning is proposed to predict the EEG signal. Input variables are created using a nonlinear transformation on lagged values of the time series. Genetic programming is used to find the nonlinear transformation. Fitness function of GP is defined based on mutual information of generated input variable and the desired output. Simulations prove the vantage of this method over the input selection method.

## REFERENCES

[1] R. Battiti, "Using mutual information for selecting features in supervised neural network," *IEEE Trans. On Neural Networks*, vol. 5, pp. 537-550, 1994

[2] A. Al-Ani, M. Deriche, "An optimal feature selection technique using the concept of mutual information," *Int. Symposium on Signal Processing and its Applications (ISSPA)*, Kuala lumpar, Malaysia, 2001, pp. 477-480.

[3] N. Kwak, C. H. Choi, "Input feature selection for classification problems," *IEEE Trans. Neural Networks*, vol. 13, pp. 143-159, 2002.

[4] M. M. Rezaei Yousefi, M. Mirmomeni, C. Lucas, "Input variable selection using mutual information for neuro fuzzy modeling with application to time series forecasting," in *proc. International Joint Conference on Neural Networks,* Orlando FL, 2007, pp. 1121-1126.

[5] A. H. Vahabie, M. M. Rezaei Yousefi, B. N. Araabi, C. Lucas, "Mutual information based input selection in neuro-fuzzy modeling for short term load forecasting of iran national power system," in *proc IEEE International Conference on Control and Automation*, Guangzhou, 2007, pp. 2710-2715.

[6] B. Atoufi, C. Lucas, A. Kalhor, M. M. Rezaei Yousefi, "Channel selection in EEG prediction: linear and nonlinear approach," , unpublished

[7] O. Nelles, *Nonlinear System Identification,* Springer Verlag, Berlin, 2001

[8] C. E. Shannon, "A mathematical theory for communication," *The Bell System Technical*, vol. 27, pp. 379–423, 623–656, 1948.

[9] D.W. Scott, *Multivariable Density Estimation: Theory, Practice, and Visualization,* New York: John Wiley, 1992.

[10] T. Lan, D. Erdogmus, U. Ozertem, Y. Huang, "Estimating mutual information using Gaussian mixture density model for feature ranking and selection", in *proc. International Joint Conference on Neural Networks*, Vancouver, 2006, pp. 5034 - 5039

[11] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physics. Review* E., vol. 69, 066138, 2004.

[12] H. Stogbauer, A. Kraskov, S. A. Astakhof, P. Grassberg, "Least dependent component analysis based on mutual information", *Physics. Review* E., vol. 70, 066123, 2004

[13] J. R. Koza, *Genetic Programming: On Programming of Computers by means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[14] E. William, J. Northern, "Genetic programming lab (GPLab) tool set version 3.0," in *proc. IEEE Region 5 Conference*, Kansas City, 2008, pp. 1-8

[15] L. Davis, "Adaptive operator probabilities in genetic algorithms," in *proc. 3rd International conference on genetic algorithms*, George Mason University, United States, pp. 61-69, 1989

[16] H. Etemadi, A. A. Anvary Rostamy, H. F. Dehkordi, "A genetic programming model for bankruptcy prediction: empirical evidence from Iran," *Expert Systems with Applications*, vol. 36, pp. 3199-3207, 2009

[17] S. Silva, J. Almeida, "Dynamic maximum tree depth – a simple technique for avoiding bloat in tree-based GP", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* vol. 2724, pp. 1776-1787, 2003

[18] http://epilepsy.uni-freiburg.de/freiburg-seizure-prediction-project/eeg-database