

# MARS: Still an Alien Planet in Soft Computing?

Ajith Abraham and Dan Steinberg<sup>†</sup>

School of Computing and Information Technology  
Monash University (Gippsland Campus), Churchill 3842, Australia  
Email: [ajith.abraham@infotech.monash.edu.au](mailto:ajith.abraham@infotech.monash.edu.au)

<sup>†</sup> Salford Systems Inc  
8880 Rio San Diego, CA 92108, USA  
Email: [dstein@salford-systems.com](mailto:dstein@salford-systems.com)

**Abstract:** The past few years have witnessed a growing recognition of soft computing technologies that underlie the conception, design and utilization of intelligent systems. According to Zadeh [1], soft computing consists of artificial neural networks, fuzzy inference system, approximate reasoning and derivative free optimization techniques. In this paper, we report a performance analysis among Multivariate Adaptive Regression Splines (MARS), neural networks and neuro-fuzzy systems. The MARS procedure builds flexible regression models by fitting separate splines to distinct intervals of the predictor variables. For performance evaluation purposes, we consider the famous Box and Jenkins gas furnace time series benchmark. Simulation results show that MARS is a promising regression technique compared to other soft computing techniques.

## 1. Introduction

Soft Computing is an innovative approach to construct computationally intelligent systems that are supposed to possess humanlike expertise within a specific domain, adapt themselves and learn to do better in changing environments, and explain how they make decisions. Neurocomputing and neuro-fuzzy computing are well-established soft computing techniques for function approximation problems. MARS is a fully automated method, based on a divide and conquer strategy, partitions the training data into separate regions, each with its own regression line or hyperplane [2]. MARS strengths are its flexible framework capable of tracking the most complex relationships, combined with speed and the summarizing capabilities of local regression. This paper investigates the performance of neural networks, neuro-fuzzy systems and MARS for predicting the well-known Box and Jenkins time series, a benchmark problem used by several connectionist researchers. We begin with some theoretical background about MARS, artificial neural networks and neuro-fuzzy systems. In Section 6 we present experimentation setup using MARS and soft computing models followed by results and conclusions.

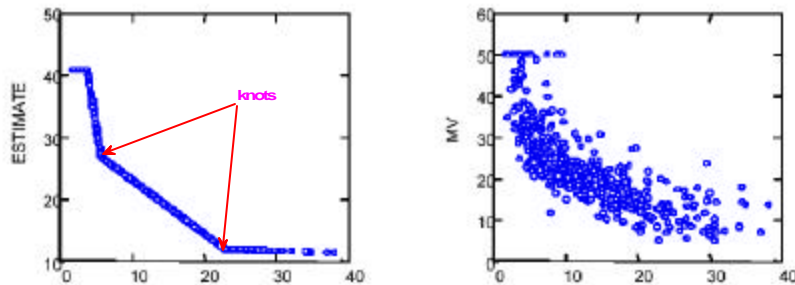
## 2. What are Splines?

Splines can be considered as an innovative mathematical process for complicated curve drawings and function approximation. Splines find ever-increasing application

in the numerical methods, computer-aided design, and computer graphics areas. Mathematical formulae for circles, parabolas, or sine waves are easy to construct, but how does one develop a formula to trace the shape of share value fluctuations or any time series prediction problems? The answer is to break the complex shape into simpler pieces, and then use a stock formula for each piece [4]. To develop a spline the X-axis is broken into a convenient number of regions. The boundary between regions is also known as a knot. With a sufficiently large number of knots virtually any shape can be well approximated. While it is easy to draw a spline in 2dimensions by keying on knot locations (approximating using linear, quadratic or cubic polynomial etc.), manipulating the mathematics in higher dimensions is best accomplished using basis functions.

### 3. Multivariate Adaptive Regression Splines (MARS)

The MARS model is a spline regression model that uses a specific class of basis functions as predictors in place of the original data. The MARS basis function transform makes it possible to selectively blank out certain regions of a variable by making them zero, allowing MARS to focus on specific sub-regions of the data. MARS excels at finding optimal variable transformations and interactions, as well as the complex data structure that often hides in high -dimensional data [3].



**Figure 1.** MARS data estimation using splines and knots (actual data on the right)

Given the number of predictors in most data mining applications, it is infeasible to approximate the function  $y=f(x)$  in a generalization of splines by summarizing  $y$  in each distinct region of  $x$ . Even if we could assume that each predictor  $x$  had only two distinct regions, a database with just 35 predictors would contain more than 34 billion regions. Given that neither the number of regions nor the knot locations can be specified a priori, a procedure is needed that accomplishes the following:

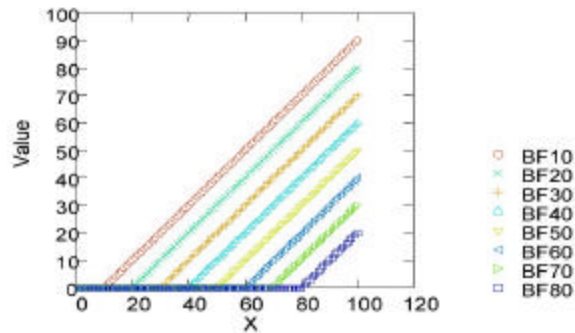
- judicious selection of which regions to look at and their boundaries, and
- judicious determination of how many intervals are needed for each variable

A successful method of region selection will need to be adaptive to the characteristics of the data. Such a solution will probably reject quite a few variables (accomplishing

variable selection) and will take into account only a few variables at a time (also reducing the number of regions). Even if the method selects 30 variables for the model, it will not look at all 30 simultaneously. Such simplification is accomplished by a decision tree (e.g., at a single node, only ancestor splits are being considered; thus, at a depth of six levels in the tree, only six variables are being used to define the node).

### MARS Smoothing, Splines, Knots Selection and Basis Functions

A key concept underlying the spline is the knot. A knot marks the end of one region of data and the beginning of another. Thus, the knot is where the behavior of the function changes. Between knots, the model could be global (e.g., linear regression). In a classical spline, the knots are predetermined and evenly spaced, whereas in MARS, the knots are determined by a search procedure. Only as many knots as needed are included in a MARS model. If a straight line is a good fit, there will be no interior knots. In MARS, however, there is always at least one "pseudo" knot that corresponds to the smallest observed value of the predictor. Figure 1 depicts a MARS spline with three knots.



**Figure 2.** Variations of basis functions for  $c = 10$  to  $80$

Finding the one best knot in a simple regression is a straightforward search problem: simply examine a large number of potential knots and choose the one with the best  $R^2$ . However, finding the best pair of knots requires far more computation, and finding the best set of knots when the actual number needed is unknown is an even more challenging task. MARS finds the location and number of needed knots in a forward/backward stepwise fashion. A model which is clearly overfit with too many knots is generated first, then, those knots that contribute least to the overall fit are removed. Thus, the forward knot selection will include many incorrect knot locations, but these erroneous knots will eventually, be deleted from the model in the backwards pruning step (although this is not guaranteed).

Thinking in terms of knot selection works very well to illustrate splines in one dimension; however, this context is unwieldy for working with a large number of

variables simultaneously. Both concise notation and easy to manipulate programming expressions are required. It is also not clear how to construct or represent interactions using knot locations. In MARS, Basis Functions (BFs) are the machinery used for generalizing the search for knots. BFs are a set of functions used to represent the information contained in one or more variables. Much like principal components, BFs essentially re-express the relationship of the predictor variables with the target variable. The hockey stick BF, the core building block of the MARS model is often applied to a single variable multiple times. The hockey stick function maps variable  $X$  to new variable  $X^*$ :

$$\max(0, X - c), \text{ or} \\ \max(0, c - X)$$

where  $X^*$  is set to 0 for all values of  $X$  up to some threshold value  $c$  and  $X^*$  is equal to  $X$  for all values of  $X$  greater than  $c$ . (Actually  $X^*$  is equal to the amount by which  $X$  exceeds threshold  $c$ ). The second form generates a mirror image of the first. Figure 2 illustrates the variation in BFs for changes of  $c$  values (in steps of 10) for predictor variable  $X$ , ranging from 0 to 100.

MARS generates basis functions by searching in a stepwise manner. It starts with just a constant in the model and then begins the search for a variable-knot combination that improves the model the most (or, alternatively, worsens the model the least). The improvement is measured in part by the change in Mean Squared Error (MSE). Adding a basis function always reduces the MSE. MARS searches for a pair of hockey stick basis functions, the primary and mirror image, even though only one might be linearly independent of the other terms. This search is then repeated, with MARS searching for the best variable to add given the basis functions already in the model. The brute search process theoretically continues until every possible basis function has been added to the model.

In practice, the user specifies an upper limit for the number of knots to be generated in the forward stage. The limit should be large enough to ensure that the true model can be captured. A good rule of thumb for determining the minimum number is three to four times the number of basis functions in the optimal model. This limit may have to be set by trial and error.

#### **4. Artificial Neural Network (ANN)**

ANN is an information-processing paradigm inspired by the way the densely interconnected, parallel structure of the mammalian brain processes information. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons [7]. Learning typically occurs by example through training, where the training algorithm iteratively adjusts the connection weights (synapses). These connection weights store the knowledge necessary to solve specific problems. A typical three-layer feedforward neural network is illustrated in Figure 3. Backpropagation (BP) is one of the most famous training algorithms for multilayer perceptrons. BP is a gradient descent technique to minimize the error  $E$  for a

particular training pattern. For adjusting the weight ( $w_{ij}$ ) from the  $i$ -th input unit to the  $j$ -th output, in the batched mode variant the descent is based on the gradient  $\nabla E$  ( $\frac{\partial E}{\partial w_{ij}}$ ) for the total training set:

$$\Delta w_{ij}(n) = -\alpha * \frac{\partial E}{\partial w_{ij}} + \alpha * \Delta w_{ij}(n-1)$$

The gradient gives the direction of error  $E$ . The parameters  $\alpha$  and  $\beta$  are the learning rate and momentum respectively.

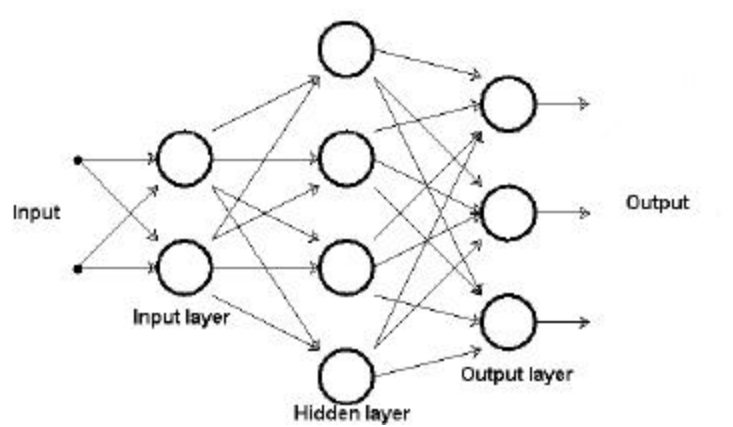


Figure 3. Typical three-layer feedforward network architecture

## 5. Neuro-Fuzzy (NF) System

We define a NF [6] system as a combination of ANN and Fuzzy Inference System (FIS) [9] in such a way that neural network learning algorithms are used to determine the parameters of FIS. As shown in Table 1, to a large extent, the drawbacks pertaining to these two approaches seem largely complementary.

Table 1. Complementary features of ANN and FIS

ANN	FIS
Black box	Interpretable
Learning from scratch	Making use of linguistic knowledge

In our simulation, we used ANFIS: Adaptive Network Based Fuzzy Inference System [5] as shown in Figure 5, which implements a Takagi Sugeno Kang (TSK) fuzzy

inference system (Figure 4) in which the conclusion of a fuzzy rule is constituted by a weighted linear combination of the crisp inputs rather than a fuzzy set.

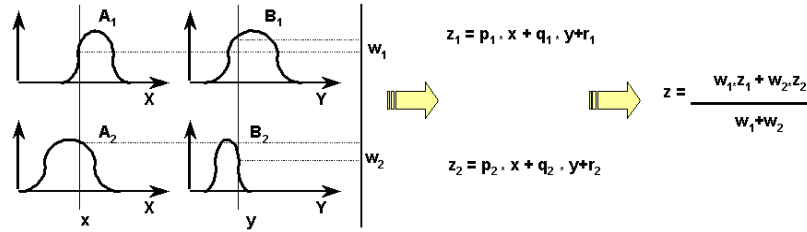


Figure 4. TSK type fuzzy inference system

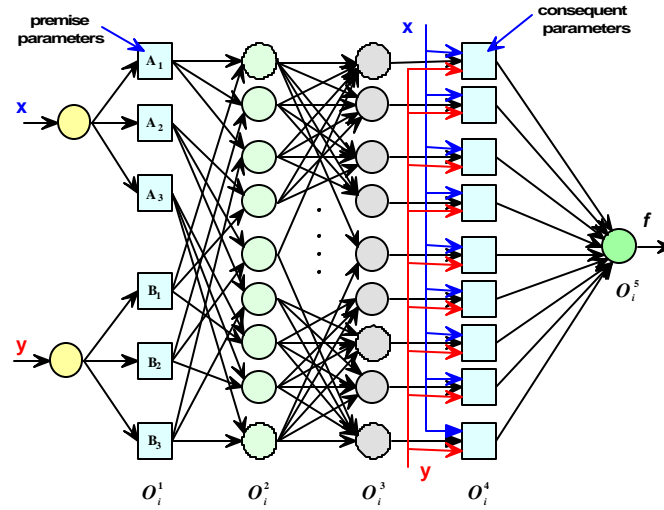


Figure 5. Architecture of the ANFIS

Architecture of ANFIS and the functionality of each layer is as follows:

**Layer-1** Every node in this layer has a node function

$$o_i^1 = m_{A_i}(x), \text{ for } i=1, 2$$

or

$$o_i^1 = m_{B_{i-2}}(y), \text{ for } i=3, 4, \dots$$

$O_i^1$  is the membership grade of a fuzzy set A ( = A<sub>1</sub>, A<sub>2</sub>, B<sub>1</sub> or B<sub>2</sub>), specifies the degree to which the given input  $x$  (or  $y$ ) satisfies the quantifier A. Usually the node function can be any parameterized function.. A gaussian membership function is specified by two parameters  $c$  (membership function center) and  $\sigma$  (membership function width)

$$\text{gaussian}(x, c, \sigma) = e^{-\frac{1}{2} \left( \frac{x-c}{\sigma} \right)^2}$$

Parameters in this layer are referred to premise parameters.

**Layer-2** Every node in this layer multiplies the incoming signals and sends the product out. Each node output represents the firing strength of a rule.

$$O_i^2 = w_i = \mathbf{m}_{A_i}(x) \times \mathbf{m}_{B_i}(y), i = 1, 2, \dots .$$

In general any T-norm operators that perform fuzzy AND can be used as the node function in this layer.

**Layer-3** Every  $i$ -th node in this layer calculates the ratio of the  $i$ -th rule's firing strength to the sum of all rules firing strength.

$$O_i^3 = \overline{w}_i = \frac{w_i}{w_1 + w_2}, i = 1, 2, \dots .$$

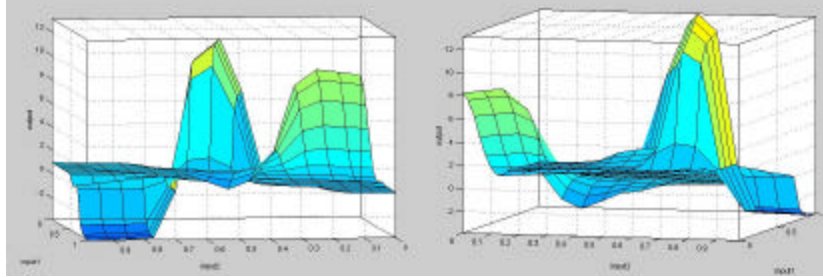
**Layer-4** Every node  $i$  in this layer has a node function

$$O_i^4 = \overline{w}_i f_i = \overline{w}_i ( p_i x + q_i y + r_i ),$$

where  $\overline{w}_i$  is the output of layer3, and  $\{p_i, q_i, r_i\}$  is the parameter set. Parameters in this layer will be referred to as consequent parameters.

**Layer-5** The single node in this layer labeled  $\hat{O}$  computes the overall output as the summation of all incoming signals:  $O_1^5 = \text{Overall output} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$ .

ANFIS makes use of a mixture of backpropagation to learn the premise parameters and least mean square estimation to determine the consequent parameters. A step in the learning procedure has two parts: In the first part the input patterns are propagated, and the optimal conclusion parameters are estimated by an iterative least mean square procedure, while the antecedent parameters (membership functions) are assumed to be fixed for the current cycle through the training set. In the second part the patterns are propagated again, and in this epoch, backpropagation is used to modify the antecedent parameters, while the conclusion parameters remain fixed. This procedure is then iterated.



**Figure 6.** 3D view of Gas furnace time series training data (I/O relationship)

## 6. Experimental Setup Using Soft Computing Models and MARS

**Gas Furnace Time Series Data:** This time series was used to predict the CO<sub>2</sub> (carbon dioxide) concentration  $y(t+1)$  [10]. In a gas furnace system, air and methane are combined to form a mixture of gases containing CO<sub>2</sub>. Air fed into the gas furnace is kept constant, while the methane feed rate  $u(t)$  can be varied in any desired manner. After that, the resulting CO<sub>2</sub> concentration  $y(t)$  is measured in the exhaust gases at the outlet of the furnace. Data is represented as  $[u(t), y(t), y(t+1)]$ . The time series consists of 292 pairs of observation and 50% was used for training and remaining for testing purposes. Figure 6 shows the complexity of the input / output relationship in two different angles. Our experiments were carried out on a PII, 450MHz Machine and the codes were executed using MATLAB and C++.

- **ANN training**

We used a feedforward neural network with 1 hidden layer consisting of 24 neurons (tanh-sigmoidal node transfer function). The training was terminated after 6000 epochs. Initial learning rate was set at 0.05.

- **ANFIS training**

In the ANFIS network, we used 4 Gaussian membership functions for each input parameter variable. Sixteen rules were learned based on the training data. The training was terminated after 60 epochs.

- **MARS**

We used 5 basis functions and selected 1 as the setting of minimum observation between knots. To obtain the best possible prediction results (lowest RMSE), we sacrificed the speed (minimum completion time).

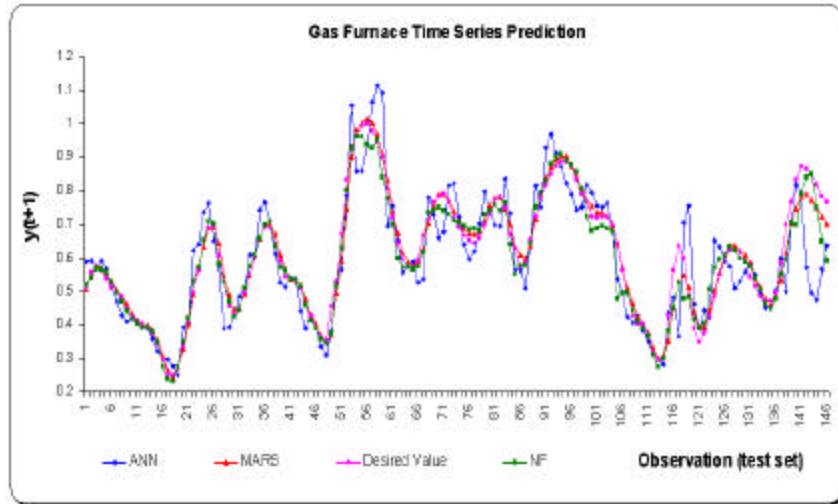


Figure 7. Gas furnace series prediction using soft computing models and MARS

- **Performance and results achieved**

Figure 7 illustrates the test results achieved for the gas furnace time series. Table 2 summarizes the comparative performances of the different soft computing models and MARS in terms of performance time, training error and testing error obtained.

Table 2. Results showing performance comparison between MARS and soft computing models for gas furnace series prediction

Model	Root Mean Squared Error		BFlops	Epochs	Training time (seconds)
	Training Set	Test Set			
MARS	0.0185	0.0413	-	-	1
ANN	0.0565	0.0897	0.5802	6000	250
NF	0.0137	0.0570	0.0005	60	40

\*Computational load in billion flops (BFlops)

## 7. Conclusion

In this paper we have investigated the performance of MARS and compared the performance with artificial neural networks and neuro-fuzzy systems (ANFIS), which are well-established function approximators. Our experiments to predict the

benchmark time series reveal the efficiency of MARS. In terms of both RMSE (test set) and performance time, MARS outperformed the soft computing models considered.

MARS can no longer be considered an alien planet considering the performance depicted in Table 2. It will be interesting to study the robustness of MARS compared to neural networks and neuro-fuzzy systems. Choosing suitable parameters for a MARS network is more or less a trial and error approach where optimal results will depend on the selection of parameters. Selection of optimal parameters may be formulated as an evolutionary search [8] to make MARS fully adaptable and optimal according to the problem.

## References

- [1] Zadeh LA, *Roles of Soft Computing and Fuzzy Logic in the Conception, Design and Deployment of Information/Intelligent Systems*, Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, O Kaynak, LA Zadeh, B Turksen, IJ Rudas (Eds.), pp1-9, 1998.
- [2] Friedman, J. H, *Multivariate Adaptive Regression Splines*, Annals of Statistics, Vol 19, 1-141, 1991.
- [3] Steinberg, D, Colla, P. L., and Kerry Martin (1999), *MARS User Guide*, San Diego, CA: Salford Systems, 1999.
- [4] Shikin E V and Plis A I, *Handbook on Splines for the User*, CRC Press, 1995.
- [5] Jang J S R, *Neuro-Fuzzy Modeling: Architectures, Analyses and Applications*, PhD Thesis, University of California, Berkeley, July 1992.
- [6] Abraham A and Nath B, *Designing Optimal Neuro-Fuzzy Systems for Intelligent Control*, The Sixth International Conference on Control, Automation, Robotics and Vision, (ICARCV 2000), December 2000.
- [7] Abraham A and Nath B, *Optimal Design of Neural Nets Using Hybrid Algorithms*, In proceedings of 6<sup>th</sup> Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000), pp. 510-520, 2000.
- [8] Fogel D, *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*, 2<sup>nd</sup> Edition, IEEE press, 1999.
- [9] Cherkassky V, *Fuzzy Inference Systems: A Critical Review*, Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, Kayak O, Zadeh LA et al (Eds.), Springer, pp.177-197, 1998.
- [10] Box G E P and Jenkins G M, *Time Series Analysis, Forecasting and Control*, San Francisco: Holden Day, 1970.