

Exploring MARS: An Alternative to Neural Networks

by Will Dwinell

Introduction

Since their inception, neural networks have commanded a significant amount of attention in the world of machine learning – yet neural networks are not the only game in town. While they have been instrumental in solving many difficult real-world problems, keep in mind the wide array of alternatives. One such alternative is Multivariate Adaptive Regression Splines (MARS), developed by statistician Jerome Friedman.

MARS segments the space of possible input cases into rectangular regions that are fit with linear or cubic splines — splines being moderately complex curves. The MARS approach has proven effective at a variety of learning problems and is competitive with neural networks and Non-parametric regressions, such as k-nearest neighbors. After its recent incorporation into the suite of products offered by Salford Systems (www.salford-systems.com), MARS has been further developed into a fine commercial tool (see Figures 1 and 2). MARS is an excellent example of the powerful modeling tools developed by statisticians and is explored in this article.

A standard machine learning benchmarking data set, the Boston Housing data, is used to demonstrate MARS's operation principles. This data, available on-line at the UCI Machine Learning Repository (www.ics.uci.edu/~mlern/MLRepository.html) contains information on 506 examples of housing data from the Boston area consisting of 13 numeric variables and 1 binary variable. One numeric variable, the

median value of owner occupied homes (in thousands of dollars), is usually defined as the target variable and the rest are model inputs. MARS parameters are left at their default settings except that: 1) 2-variable interactions are allowed and 2) 10-fold cross validation is used for testing.

Local, Global and otherwise

The set of all possible input cases make up what is known as the 'input space'. Learning systems create models that span the input space in different ways. Depending on how their models cover the input space,

learning algorithms may be placed on a spectrum from local to global. Local models utilize a large collection of small 'micro-models' that handle specific, small regions of the input space. Global models employ a single, monolithic model covering the entire input space. Very often, local models are fast to train, but slow to recall, whereas global models are slower to train, but quick to recall. More importantly, local models quickly learn to solve the specific types of problems that are presented during training, but are slow to generalize to other types of problems. In contrast, global models are

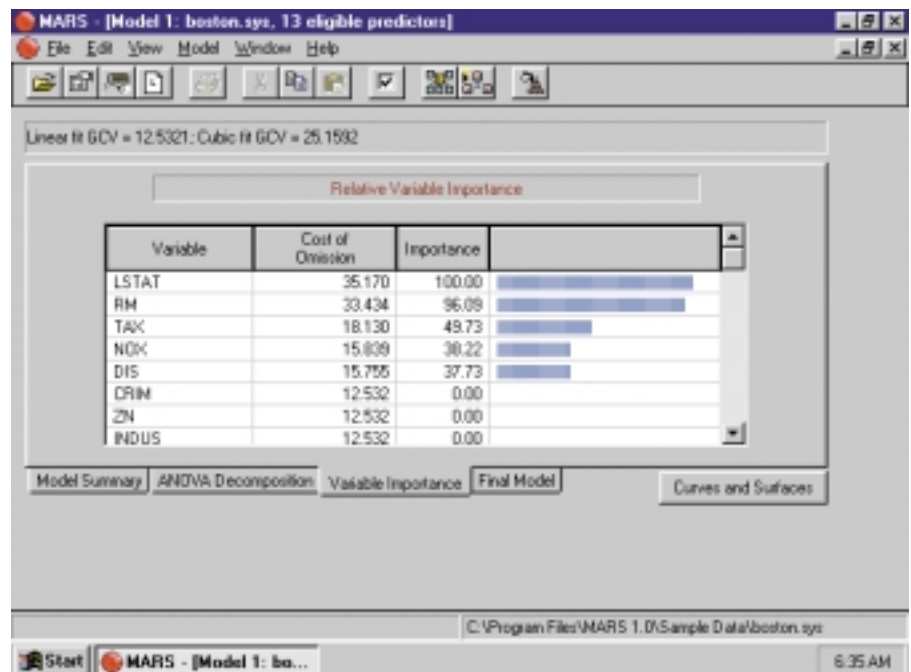


Figure 1: This screenshot from the MARS application shows the Variable Importance ratings for the Boston Housing model.

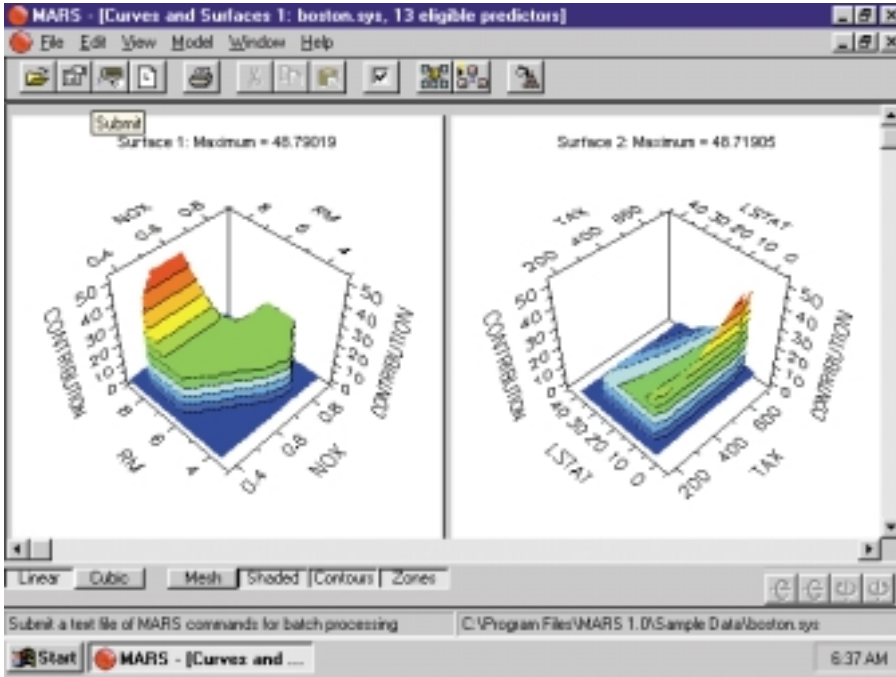


Figure 2: This screenshot from MARS displays a 3-dimensional graph of the model output against various pairs of input variables.

general and handle most problems somewhat well, but may have more difficulty learning to handle specific, quirky cases.
 During learning, MARS splits the

input space into regions of varying sizes — building a local model for each region. The size of these regions may be as small or as large as deemed necessary by MARS, allowing it to adjust its level of ‘localness’ as

appropriate, even within a single model. The greater the density of training cases and the greater the complexity of the relationship between inputs and the output, the smaller the regions may become. In this way, MARS adapts itself across the input space to squeeze maximum performance from the available training data. To some extent, conventional feedforward neural networks (being global models) are forced to spread their attention much more evenly across all cases. For learning problems that present a varying density of examples, MARS may be better able to adapt its solutions than feedforward neural networks.

MARS performs input selection automatically. When a MARS’s model was built of the Boston Housing data, it selected 5 of the 13 input variables for inclusion. In comparison, feedforward neural networks can learn to effectively ignore useless inputs, but they do not explicitly exclude them from the model. This gives MARS two advantages over neural networks for certain types of data:

- Resulting model code will be smaller, tighter and presumably run faster
- Explicit selection of useful inputs aids in human understanding of the data and the model.

1/2 Pg Ad
 Salford
 PU 13.6, Pg. 8, #6

Additionally, MARS explicitly defines the boundaries of the various regions it constructs, providing the user with a very good idea of where important changes in the data occur.

Size and Speed

While learning times vary with the learning task, MARS generally constructs models at speeds comparable to those of neural network tools. On a 233 MHz AMD K6 PC with 64 megabytes of RAM, MARS took less than a minute to model the Boston Housing data. Remember that with 10-fold cross-validation, 11 full models need to be built: 10 cross-validation models to ascertain the optimal model settings plus the final model. Models are ordinarily fairly compact, although this naturally will depend on the data. The MARS model of the Boston Housing data, containing 12 basic functions (regions), could be implemented in approximately 15 lines of code in C or another high-level language. The actual code

for this model, as pulled directly from MARS is:

$$\begin{aligned} \text{BF1} &= \max(0, \text{LSTAT} - 6.070); \\ \text{BF2} &= \max(0, 6.070 - \text{LSTAT}); \\ \text{BF3} &= \max(0, \text{RM} - 6.425); \\ \text{BF4} &= \max(0, 6.425 - \text{RM}); \\ \text{BF5} &= \max(0, \text{NOX} - 0.647) * \text{BF3}; \\ \text{BF8} &= \max(0, 1.386 - \text{DIS}); \\ \text{BF9} &= \max(0, \text{TAX} - 187.000) * \text{BF1}; \\ \text{BF10} &= \max(0, \text{TAX} - 296.000) * \text{BF2}; \\ \text{BF11} &= \max(0, 296.000 - \text{TAX}) * \text{BF2}; \\ \text{BF12} &= \max(0, \text{RM} - 6.216) * \text{BF8}; \\ \text{BF13} &= \max(0, 6.216 - \text{RM}) * \text{BF8}; \\ \text{BF14} &= \max(0, \text{LSTAT} - 20.620) * \text{BF4}; \\ \\ \text{Y} &= 23.529 - 0.349 * \text{BF1} + 11.713 * \text{BF3} - \\ &209.030 * \text{BF5} \\ &+ 127.078 * \text{BF8} - .818217\text{E-}03 * \text{BF9} + 0.024 * \\ &\text{BF10} \\ &+ 0.036 * \text{BF11} - 219.330 * \text{BF12} - 63.201 * \text{BF13} \\ &+ 0.297 * \text{BF14}; \end{aligned}$$

LSTAT, RM, NOX, DIS and TAX are the model inputs. The Basic Function (BF) variables are used to cover various regions of input space and hold intermediate calculations in the model. Although they are numbered from 1 to 14, only 12 are used. The model output, Y, is a function of these basic functions. In terms of compute time, this is a fairly Spartan model: only 12 conditionals, 13 assignments, 21 additions and 17 multiplications are needed. It may be possible to pare this down even farther (by hand) if more conditionals were used to limit the floating point operations to only those needed in any given circumstance — the setting of certain basic functions to zero is intended to turn them “off”. Even a modest neural network requires many more additions and multiplications for a problem of this size.

Recall speed is also an issue in modeling: we are not just concerned with how quickly our learning system learns, but also how quickly it can recall (answer a

Other Alternatives

Aside from MARS, there are a great many nonlinear modeling tools available. The statistical field alone has produced a host of such techniques. While an exhaustive list of non-neural techniques is not practical, consider adding any of the following to your toolbox:

- Adaptive Linear Networks
- ALISA
- ARIMA
- Box-Jenkins
- C4.5
- CART
- CHAID
- genetic programming
- k-nearest neighbors
- kernel smoothers
- LOESS / LOWESS
- local linear models
- logistic regression
- orthogonal series estimators
- projection pursuit regression
- robust regression
- Savitzky-Golay smoothing
- splines
- Tukey's smoothers

A good statistical reference for such techniques is *Applied Non-parametric Regression*, by Hardle, published by Cambridge University Press, ISBN 0-521-42950-1. It covers a number of these methods from a statistical perspective. While rigorous, most of the book should be accessible to PC AI readers who spend any amount of time working with neural networks.

Also consider moving beyond plain vanilla backprop-trained feedforward neural networks. There are dozens if not hundreds of neural architectures. Even within the realm of feedforward neural networks there are many variations, such as jump connections and alternative training algorithms (simulated annealing, competitive genetic algorithms, and so forth). Again, a complete list cannot be provided, but consider the following neural architectures:

- CMAC
- CPM
- Dystal
- functional link networks (FLN)
- generalized regression neural networks (GRNN)
- higher-order neural networks (HONN)
- LVQ
- Logicon Projection Network
- polynomial networks (GMDH)
- probabilistic neural networks (PNN)
- RAM neuron discriminators
- RCE
- recurrent neural networks
- self-organizing map (SOM)
- sparse distributed matrices (SDM)

My article, “Alternative Neural Architectures”, which appeared in the Q3, 1998 issue of the Gordian Institute Electronic Newsletter (on the Web at www.gordianknot.com/virtualoffice/news1.htm#neural) covers several alternative neural systems and their use. An excellent exploration of alternative neural networks is

available in *Advanced Methods in Neural Computing*, by Wasserman, published by VNR, ISBN 0-442-00461-3.

Specialized algorithms, developed for solving specific types of problems, will usually outperform a general-purpose neural network. As an example, consider anomaly detection, in which the goal is to identify ‘unusual’ cases — those that are unlike any found in the training set. This important task, often referred to as deviation detection and one-class learning, is found in fields such as statistical quality control, medical diagnosis, machine performance monitoring, and fraud detection. Some algorithms, like artificial immune systems, are readily applicable to this problem. Since all the training data comes from the class considered ‘normal’ or ‘usual’, training a neural network requires transforming the problem — perhaps by adding artificial cases to the training set to be labeled as ‘unusual’. Consider the difficulty in doing this, however, since randomly generated cases might actually be similar to ‘normal’ data and confuse the neural network.

Avoid becoming locked into a single solution. Keep in mind that no learning system is best for all problems and that a different learning system provides important design trade-off. Having more tools rather than less gives the modeler greater flexibility in solving real problems. Insight into the nature of the learning task at hand is often gained by comparing performance of diverse modeling tools, especially ones which select and discard inputs automatically.

question about a new case). Some researchers have made a persuasive case for the idea that recall speed is more important than learning speed since (in most applications) the system will learn only once but will recall many times. With no large weight matrices to multiply, the recall speed of MARS's models is generally fast - at least in the experiments I performed.

As a test of the recall speed of MARS's models, I ported the Boston Housing model to BASIC for the Atari 800 home computer. This 1979 vintage computer has an 8-bit microprocessor running at 1.77 MHz and 48 kilobytes of RAM. Even running in interpreted BASIC on a computer with no floating point hardware, the model executed fast enough (under half of a second) that, in an interactive setting, the computer would end up waiting for the user.

Conclusion

While neural networks are in vogue at the moment, it is worth any analyst's time and energy to investigate other modeling techniques. Although some neural network alternatives, such as MARS, are built on a very rigorous statistical basis — some ad hoc methods can be very useful. In some

circles, techniques born in the statistical community are preferred — not for technical but political reasons. Right or wrong, there are those that prefer methods which enjoy a certain 'celebrity status', having been invented by prominent statisticians. While having nothing to do with the technical merits of the technologies in question, this may be an important selling point for consultants who must deal with their client's prejudice against newer technologies like neural networks or fuzzy logic.

The literature records a variety of successful applications of MARS, especially in time-series work. MARS deals with high dimensional problems in a straightforward manner, is deterministic and can handle missing values. While MARS won't outperform neural networks every time, it is worth adding to one's toolbox.

Further Reading

As an example of how MARS has been used, see *Time Series Prediction*, edited by Weigend and Gershenfeld, ISBN 0-201-62602-0, which includes an interesting article, "Modeling Time Series by Using Multivariate Adaptive Regression Splines (MARS)", by Lewis, Ray and Stevens.

For another treatment of local and global modeling systems, see my article "Modeling Methodology 4: Localizing Global Models", which appeared in the May/June issue, 1998 of *PC AI* Vol. 12.3.

For a more thorough discussion of algorithm selection issues in machine learning, see my article "Modeling Methodology 3: Algorithm Selection", which was published in the Mar./Apr., 1998 issue of *PC AI* Vol. 12.2.



Will Dwinell is an artificial intelligence consultant who lives and works in southeastern Pennsylvania. He can be reached at predictor@compuserve.com.



Coming Attractions

Next Few Issues

Themes

- Fuzzy Logic
- Intelligent Knowledge Management
- Intelligent Process Control
- Intelligent Web Applications
- Neural Networks
- Object Oriented Development

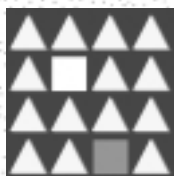
Topics

- AI Languages - Prolog and LISP
- Applications of AI
- Business Forecasting
- Data Mining and Knowledge Discovery
- Expert Systems
- Intelligent Searching
- Intelligent Web Utilities
- Knowledge Management and AI
- Natural Language
- Modeling and Simulation
- Object Design Issues in Business Process Modeling

And much, much more

Practical Data Mining Tools

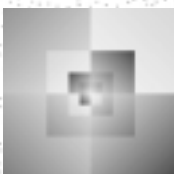
Businesses need clean data



WizRule[®]
to improve your data quality

- detects errors and anomalies (cases to be investigated) in the data
- reveals all the known and unknown rules in the content of the data
- discovers relations that may exist among unrelated fields

Businesses need the ability to predict



WizWhy[®]
to refine strategic decisions

- reveals rules, trends and interesting phenomena in the data
- predicts the outcome for new cases based on the discovered rules
- summarizes the data textually and graphically

for a **FREE** demo & detailed information contact

WizSoft[®]

(516) 393-5841 ♦ info@wizsoft.com ♦ www.wizsoft.com