

Induction and Polynomial Networks

John F. Elder IV
Donald E. Brown

IPC-TR-92-009
8/20/92

Institute for Parallel Computation
and
Department of Systems Engineering
University of Virginia
Charlottesville, VA 22901
804-924-5393

Internet: jfe7w@virginia.edu

Acknowledgement

This work was supported in part by the Jet Propulsion
Laboratory under grant number 95722.

TABLE OF CONTENTS

1. INTRODUCTION.....	1
2. INDUCTIVE MODELING.....	2
2.1. Representative Inductive Techniques.....	3
2.2. Induction Hierarchy.....	5
3. DATA VISUALIZATION.....	8
3.1. Combining Induction and Visualization.....	10
3.2. Projection Pursuit.....	10
3.3. Remarks.....	13
4. MODEL SELECTION.....	14
4.1. Model Accuracy.....	16
4.2. Significance Tests.....	18
4.3. ANOVA Model Selection.....	19
4.4. Explicit Complexity Penalties.....	21
4.5. Minimum Description Length.....	22
4.6. Other Complexity Criteria.....	24
4.7. Implications of the Computer Age.....	27
4.8. Remarks.....	29
5. POLYNOMIAL NETWORK ALGORITHMS.....	30
5.1. The Group Method of Data Handling (GMDH).....	30
5.2. The Polynomial Network Training Routine (PNETTR).....	35
5.3. The Algorithm for Synthesis of Polynomial Networks (ASPN).....	37
5.4. The Analogy with Evolution.....	39
5.5. Potential Improvements.....	41
6. EXAMPLE ASPN SESSIONS.....	44
6.1. Task 1 -- Moon Ballistics.....	44
6.2. Task 2 -- The Inverse Modeling (Control) Problem.....	55
6.3. Task 3 -- Impact Estimation in the Atmosphere.....	58
6.4. Remarks.....	62
7. SUMMARY AND CONCLUSIONS.....	63
REFERENCES.....	65

TABLES

Table 1.	Anscomb's (1973) Quartet.....	8
Table 2.	Criteria Scores for the Most Accurate Model of Each Size, K	26
Table 3.	Greedy Counter-Example for Regression.....	33
Table 4.	Greedy Counter-Example for CART	34
Table 5.	Moon Cannon Data	45
Table 6.	Complete ASPN Control File (Task 1).....	46
Table 7.	Model Description File "moon0.mod" (Task 1).....	48
Table 8.	C Code Generated by IMP for a Network Model (Task 3)	51
Table 9.	Statistics File Generated by IMP for a Network Model (Task 3).....	53
Table 10.	Network Performance Summary.....	61

FIGURES

Figure 1.	Three Types of Inductive Models.....	2
Figure 2.	A Quartet of Data Series (after Anscomb, 1973).....	9
Figure 3.	1-Dimensional Projections of Sample 2-Dimensional Data.....	11
Figure 4.	3-Dimensional Views of a Projection Pursuit "Score Surface"	12
Figure 5.	Example of Model Overfit.	15
Figure 6.	Model Selection Example.....	20
Figure 7.	Three-Layer GMDH Network	31
Figure 8.	Model Accuracy vs. Complexity.....	32
Figure 9.	(Sub-optimal) CART Tree for Data of Table 4 ($XOR(b,c)$).....	34
Figure 10.	Binary Tree Implementing $XOR(b,c)$	35
Figure 11.	Sample ASPN Network.	38
Figure 12.	AIM Interface for MacIntoshtm.....	40
Figure 13.	ASPN Screen at End of Task 1.....	47
Figure 14.	γ vs. X (Task 2)	56
Figure 15.	Model Error vs. Estimate (Task 2)	57
Figure 16.	Task 3 Data Regions.	59
Figure 17.	Network to Estimate Final Velocity, Position, Time, and Angle (Task 3).....	60

1. INTRODUCTION

The aim of inductive modeling is to infer general laws from specific cases -- a process central to the scientific method. Researchers must often summarize observations about a phenomenon into a coherent picture (a theory, or model of the underlying “data generating machinery”) which can be tested for explanatory power on new cases. To perform well on data not seen during “training”, such models need appropriate structure and complexity; that is, they must be powerful enough to approximate the known data, but simple enough to successfully generalize to new cases.

A wide variety of inductive modeling algorithms exist for practical use; these range from artificial neural networks (ANNs) and decision trees, to kernel and spline techniques. The variety of induction methods indicates both the importance of the field from an application perspective and the inherent difficulty in performing induction tasks. This chapter aims to 1) survey and categorize leading techniques for inducing estimation, classification, and control models from sample data, 2) suggest fruitful enhancements to and combinations of the methods, and 3) demonstrate the power of a particular induction technique – polynomial networks. This chapter moves from a broad overview of induction techniques to general purpose enhancements to these techniques, and finally to a particularly effective but less well-known approach, polynomial networks, that developed out of the neural network and adaptive systems communities.

The remainder of this chapter is organized as follows. Section 2 provides a brief survey of the leading techniques, and organizes them into a hierarchy according to the level of “decisions” they leave to the computer. Sections 3 and 4 contain our suggested enhancements to inductive modeling. In particular, Section 3 describes some of the principle hazards of inferring useful models from data, and proposes a simple way to improve model induction by integrating visual information. Section 4 discusses model selection and demonstrates the performance of a number of complexity-penalty metrics on an example problem. The last two sections concentrate on networks of polynomial nodes, such as created by the Group Method of Data-Handling, GMDH (Ivakhnenko, 1968; Farlow, 1984), the Polynomial Network Training Algorithm, PNETTR-IV (R. Barron, Mucciardi, Cook, Craig, and A. Barron, 1984) and the Algorithm for Synthesis of Polynomial Networks, ASPN (Elder; 1985, 1989). Section 5 describes each of these approaches to the construction of polynomial networks and concludes with suggested enhancements to the methods. Section 6 provides examples of the use of polynomial networks. Finally, Section 7 contains conclusions and suggestions for future research in induction.

2. INDUCTIVE MODELING

The goal of inductive modeling is to efficiently and robustly model high-dimensional data. As shown in Figure 1, given any two components of the set {Inputs, System State, Outputs} one can potentially induce the third. For example, a *training* data base of inputs, \mathbf{X} , and system states, \mathbf{S} , can be employed to estimate the ensuing outputs, \mathbf{Y} , (for unseen cases) in a *forecast* or prediction model. Likewise, it is a *design* problem to obtain a system, \mathbf{S} , for given inputs and outputs \mathbf{X} and \mathbf{Y} ; and a *control* task to seek the best \mathbf{X} for a given \mathbf{S} and \mathbf{Y} .

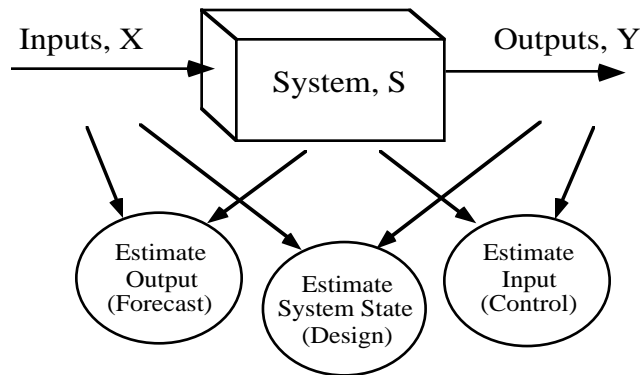


Figure 1. Three Types of Inductive Models

Induction techniques can be put into two broad classes:

- 1) **Consensus** methods, which parametrically summarize data points, and
- 2) **Contributory** methods, which retain the training data, and may employ one or every training case in the estimate for a new case.¹

With consensus methods, the training data are employed only to obtain a model (and are then discarded); for contributory techniques, at least a subset of the data are retained for on-line operation of the model. Consensus methods are quite efficient in their representation of the model but generally have to be retrained in the presence of new data. Contributory methods typically require more storage space for their implicit model, but are quickly updated with new data.

¹Contributory methods are most often referred to as *nonparametric* techniques, though some parameters are usually involved; e.g., bin or kernel widths, number of neighbors, type of wavelet, etc.

Section 2.1 provides a brief overview of representative inductive techniques and classifies them as consensus, contributory or a combination of both. Section 2.2 provides a more complete categorization, placing techniques in an induction hierarchy based on the amount of decision making performed by the computer.

2.1. Representative Inductive Techniques

By far the most popular consensus technique is Linear (least-squares) Regression (LR). Usually, all the equation terms (i.e., the full model structure) are specified by the analyst and only the parameters are estimated by the computer. This is also the case for conventional Artificial Neural Networks (ANNs), and for logistic regression networks (e.g., Class; Cellucci and Hess, 1990). However, while the LR parameters can be found in a single step (involving matrix inversion or singular-value decomposition; e.g., (Golub and Van Loan, 1989)), the nonlinear ANN and logistic parameters require use of a much slower search technique. For ANNs, the local gradient descent of *back-propagation* (Werbos, 1974), is typically employed, and logistic weights are often set by a Levenburg-Marquardt search technique (e.g., Press, Flannery, Teukolsky, and Vetterling, 1988). In what amounts to a combination of LR and ANN approaches, networks of regression nodes are employed in the family of GMDH techniques which includes PNETTR-IV and ASPN (described in Section 5 and employed on example problems in Section 6). Polynomial networks allow the model structure to grow out of the data, rather than being set by an analyst beforehand -- a significant strength of the method. Other adaptive tools include: Classification and Regression Trees, CART (Breiman, Friedman, Olshen, and Stone, 1984) which employ piecewise constants for estimation; and Hinging Hyperplanes (Breiman, 1991b), using sums of multi-dimensional ramp functions.

The CART algorithm induces classification and estimation trees in two stages: iterative forward building, followed by iterative reverse pruning. During the former, its goal is to find successive binary splits of the local data until representatives of only one class (or one response level, for estimation) remain in each *leaf* (terminal node) of the tree. After this *greedy* (i.e., one-step optimal) successive refinement strategy creates the tree, the branches are pruned back (also greedily) to reduce complexity, using *cross-validation* or testing data to govern the trade-off between complexity and accuracy (see Section 4).

CART creates piecewise constant estimation surfaces, but those formed by polynomial and sigmoidal methods are smooth (i.e., differentiable), though nonlinear. Spline techniques, like Multiple Adaptive Regression Splines, MARS (Friedman, 1988, 1991) or the *PI* method (Breiman, 1991a), can adaptively construct smooth surfaces which typically demonstrate more *local* response

to “bumps” in the data. That is, a polynomial equation is *global*; a change to affect the equation in one location affects it everywhere (to varying degrees). A piecewise collection of splines however, can be adjusted more locally.

Contributory techniques also embody this property of *local effects*. A classic example is the Nearest Neighbor (NN) algorithm, wherein the output for a new case is estimated to be the output of the known case closest by some measure (e.g., Euclidean distance) in input space. (The k -NN method just combines, e.g., averages, the outputs of the k nearest cases.²) If closer neighbors are to be weighted more heavily than distant ones, then *kernel* estimation (Parzen, 1962; Specht, 1967) is appropriate. There, a unit kernel function, $h()$ (such as a rectangle, triangle, or Gaussian node) of a given breadth is centered over each known case, i , and is evaluated at the location of interest (the new case, j) to supply the estimate

$$Y_j \approx \frac{\sum w_i Y_i}{\sum w_i}, \quad i = 1, 2, \dots, N \quad (1)$$

for N training cases, where $w_i = h(\|X_i - X_j\|)$ is the relative weight. For example, a rectangular kernel can be thought of as a k -NN approach, with k depending on location. In classification, the relative sum of weights of each class at a location provides an estimate of its conditional probability; and, in density estimation, it forms the surface itself. This last is analogous to dribbling unit piles of sand (for 2-dimensional Gaussian kernels) over each case location to build up the density estimation surface. Kernels lead therefore, to a type of *histogram* with bins not pre-established, but following the data. Indeed, *averaged shifted histogram* techniques have been employed (Scott, 1985) which have properties intermediate between the smoothness of kernels and the speed of histograms.

Radial basis functions, RBFs (e.g., Broomhead and Lowe, 1988) are a contributory technique with a consensus aspect. Radially-symmetric functions (similar to kernels, although they may increase with distance) are centered over each training case. Yet, w_i values are not used directly (as with kernels), but become variables in an LR data base. That is, each training output, Y_j , is modeled as a weighted sum of a constant and the $N-1$ distance functions $h(\|X_i - X_j\|)$, $i \neq j$. As the number of constraints, N , matches the degrees of freedom available, there is an exact model (zero training error) which is used to interpolate new cases. In practice however, the variables are

²Other refinements to the Nearest Neighbor approach include *editing* (iteratively removing misclassified training cases) and *condensing* (removing training cases completely surrounded by others of its class) (Fukunaga, 1990).

often too numerous for model robustness, and the training data is pared (e.g., Chen, Cowan, and Grant, 1991); i.e., features are removed to reduce collinearity (e.g., Belsley, 1991; Elder, 1990).

Wavelets are a recent contributory technique having useful theoretical properties and the ability to describe functions with rapid level transitions (e.g., Bock, 1992). Another estimation method capable of representing abrupt surface changes, *Delaunay planes*, can be taken from a recent optimization algorithm (Elder [this volume], Section 6). The expected value of the unknown function, known only at a set of training locations, is represented by a piecewise planar surface. Each plane is defined by a simplex of training points (a $d+1$ point set in d -dimensions; e.g., a triangle in two-space) connected according to the “empty circumsphere” rule of Delaunay triangulation (the circumscribing sphere for each simplex encloses no other point). The variance of this estimate (assuming a random field, or fractal surface) is Gaussian, and grows from a minimum at the known points to peak at uncharted locations -- forming a piecewise quadratic “variance canopy” which arches over the expectation planes. Unlike global methods, wherein each training point affects the estimate (to some degree) at every location, use of the piecewise planar surface can insulate one region from another, as points outside a given simplex have no effect on its interior.

A wide range of other algorithms for automatically inducing models from sample data are emerging. A sample of further promising methods includes *locally weighted regression* procedures (e.g., Cleveland and Devlin, 1988; Fan, 1991); *stochastic* models (Sacks, Schiller and Welch, 1989); *regression trees*³(Chaudhuri, Huang, Loh, and Yao, 1990), *slicing inverse regression* (Duan and Li, 1991); *Fit-Short* (an adaptive kernel method; Kozek and Schuster, 1991), and *composite* models (Skeppstedt, Ljung and Millnert, 1992). Many of the techniques differ significantly in the (often implicit) assumptions made about the data, and in the demands on (or opportunities for) the user to direct the analysis. Their diverse strengths suggest that a collection of several of these semi-automatic methods could be simultaneously employed on difficult problems (as suggested for visual search in Section 3).

2.2. Induction Hierarchy

To classify the growing list of viable induction methods (Section 2.1), and to clarify the appropriateness of a technique for a given task, we propose an induction hierarchy. We scale this hierarchy by the increasing degree to which a technique leaves modeling “decisions” to the computer:

³Regression Trees embody the type of method combination (which is bound to increase): they use *decision trees* to hierarchically separate the data into disjoint regions, each of which is fit by separate *LR polynomial models* and then, for a particular location, employ sums of sub-model estimates, weighted according to nearness (like *kernels*).

- 1) *Heureka*: The analyst specifies the entire model. (Like the legend of Athena from Zeus, it springs forth fully formed from one's head -- terms, weights, and all.)
- 2) *Knobby*: One specifies the structure (e.g., inputs and terms) and the computer searches for parameter settings (weights) which best fit the data. This is the most common category of models, and includes LR and most ANNs.⁴
- 3) *Plastic*: The analyst specifies the model class (polynomials, for example) and initiates a search for both structure and weights. This class can be further subdivided into:
 - a) *Stretchable*. The structure can grow in only one input dimension (say, the order of a polynomial in one variable); so, the decision concerns the best cutoff point. Other examples: determining the number of useful terms for a power series expansion, or the frequencies of a low-pass spectral approximation.
 - b) *Accretable/Erodible*. The structure can grow/diminish in several dimensions. Examples include CART, and adaptive regression techniques, such as "greedy" forward term selection (add the term which most helps), greedy backward term elimination (remove the term least aiding the model), and their step-wise combination (e.g., Draper and Smith, 1966). Also included here are clustering techniques (e.g., Everitt, 1974) which attempt to create class structure out of only the data and some measure of distance or similarity.
 - c) *Malleable*. The model is a composition of functions drawn from a given family (e.g., GMDH polynomials); thus, the form is very flexible and is capable of growing in large *chunks*.⁵ Malleable techniques are capable of building new candidate features when lower-order versions of those features prove useful.
4. *General*: The analyst provides little more than representative data, and the algorithm searches across model families (sets of basis functions) and perhaps even across error metrics. No implementation of such an ambitious method is known, though the Minimum Description Length (MDL) model selection criterion (Rissanen, 1978) and other recent means of complexity regulation (A. Barron, 1991; Faraway, 1991) may be capable of judging between candidate models generated by such a search procedure.

⁴Entropy Nets (Sethi, 1990), are ANNs which are not merely "knobby", as they obtain their structure from a preliminary run of a hierarchical classifier (e.g., a decision tree program such as CART).

⁵Thereby, the *short-term optimality* of greedy growth is extended more to the medium-term.

A further distinction can be made between *closed* techniques, for which the library of potential terms must be specified by the analyst beforehand, and *open* methods, which effectively expand their libraries of potential terms as the model grows. The former includes *branch and bound* regression (e.g., Furnival and Wilson, 1974), which could be labelled a “reverse erodible plastic method” optimal for its closed library of polynomial terms. For the latter, techniques in the GMDH family, and CART provide examples of open (plastic) modeling. During synthesis of a CART decision tree, additional bifurcations are considered for each *impure leaf* (i.e., each currently terminal node holding representatives of more than one class or estimation value.) As the tree grows (or is pruned in a later simplifying stage), the list of input space partitions changes, so the CART term library can be considered open.⁶

While the above techniques have provided investigators with impressive capabilities, there remain problems in their use for many applications. This next two Sections suggest enhancements that can directly improve the applicability of these techniques across a broad range of problems.

⁶With univariate thresholds (the CART default), there are a finite number of potential cuts (and only a fraction of the $O(N^k)$ potential terms for N cases of k variables are considered); still, the set is large enough to be treated as open. For an alternative to the greedy approach in CART see Brown and Pittard (1989).

3. DATA VISUALIZATION

The trend in the above induction hierarchy is to turn over more and more of the modeling process to our untiring, precise, obedient,⁷ blindingly fast (but dimwitted) assistant: the computer. But what does our trusty aide perceive? Typical algorithms depend on summary statistics of the data being modeled. However, Anscomb (1973) showed that, in LR terms, data sequences as diverse as those of Table 1 can appear identical. Surprisingly, those vector pairs have the same marginal characteristics ($\mu_x = 9.0$, $\mu_y = 7.5$), correlation ($\rho_{xy} = .82$), linear fit ($y \approx 0.5x + 3$, with slope t -significance = 4.24), sum of squared errors (SSE = 14), and coefficient of determination ($R^2 = 0.67$). No traditional summary statistic can distinguish between the series.

Table 1. Anscomb's (1973) Quartet
(X, Y_1), (X, Y_2), (X, Y_3), (X_4, Y_4)

X	Y_1	Y_2	Y_3	X_4	Y_4
10	8.04	9.14	7.46	8	6.58
8	6.95	8.14	6.77	8	5.76
13	7.58	8.74	12.74	8	7.71
9	8.81	8.77	7.11	8	8.84
11	8.33	9.26	7.81	8	8.47
14	9.96	8.10	8.84	8	7.04
6	7.24	6.13	6.08	8	5.25
4	4.26	3.10	5.39	19	12.50
12	10.84	9.13	8.15	8	5.56
7	4.82	7.26	6.42	8	7.91
5	5.68	4.74	5.73	8	6.89

However, *visualizing the data*, through the scatterplots of Figure 2, reveals their important differences. Clearly, only for Y_1 is the LR assumption of Gaussian errors (about a line) upheld; the data of Y_2 are instead quadratic, an *outlier* upsets the trend of Y_3 , and an *influential* (non-outlying) case drives the estimator for Y_4 . A “stretchable” inductive method could correctly discover the quadratic structure of Y_2 ($-6 + \frac{11}{4}X + \frac{1}{8}X^2$), but *case analysis* (e.g., Belsley, Kuh, and Welsh, 1980) is needed to discern problems with the latter two sequences (and that capability is not yet built into any known automated induction algorithm).

⁷The precise obedience of the computer can also be frustrating, as evidenced by the anonymous ditty: “I really hate this darn machine; I wish that they would sell it. It never does quite what I mean, but only what I tell it.”

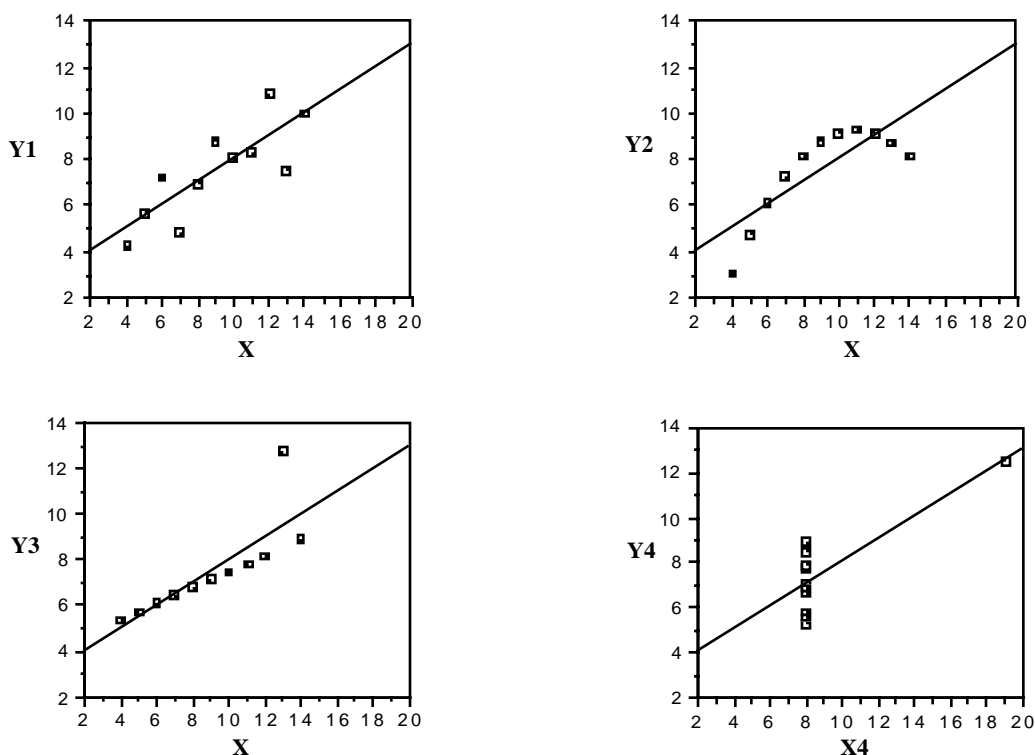


Figure 2 ^a _c ^b _d. A Quartet of Data Series (after Anscomb, 1973)
for which Common Summary Statistics are Identical

Of course, visual representation of low-dimensional problems (as in Figure 2), make the situation clear in a way statistical summaries cannot, and suggest remedies that might otherwise go unnoticed. An instructive experience for one of us occurred (years ago) on a data set eventually revealed to be similar to that of Y_3 , but with an outlier (due to a recording error) at one end of the sequence. An otherwise powerful inductive algorithm, unable to flag or down-weight the outlying case, instead chose a high-order model to fit the data. However, the program was able to winnow away a mass of variables confusing the picture and identify the one essential data feature. Plotting then revealed the (unexpected) simplicity of the relationship, obscured in summaries by the outlier - the cause of which was quickly tracked down. A similar “revelatory” insight due to visualization was reported in (Brown, Elder, and Pittard, 1992) for a sensor correlation application (explaining why a conventional “perceptron”-like linear discriminate approach was inferior to any of a number of nonlinear inductive methods). This Section outlines the excellent potential in joining automatic induction and interactive visualization tools.

3.1. Combining Induction and Visualization

In the low dimensions of everyday experience, the human ability to rapidly recognize patterns in noisy data may never be matched by automata. However, the speed and persistence of computers must be harnessed to explore spaces of high dimension. Projection Pursuit (PP) algorithms (Friedman and Tukey, 1974) seek low-dimensional views of the data which score well by some index of “interestingness”. When striving to model the data with PP regression (PPR; Friedman and Stuetzle, 1981), this perceived structure is iteratively removed and the filtered data is resubmitted, until only noise apparently remains.

In practice, the power of this idea is weakened by several factors. Though PP was inspired by early graphical data manipulation systems, most PPR-like stagewise induction programs (including GMDH, CART, ASPN, and MARS) are run in batch mode, excluding potential mid-course contributions by an analyst. PPR programs explore a single index, seeking unusual entropy, clustering, or skewness, etc.; and any structure found is addressed by a single family of model components (e.g., splines). However, many types of patterns, with different properties and shapes, may prove interesting; so, one could instead simultaneously unleash several projection index searches, as well as employ an arsenal of *corrective* basis functions (e.g., polynomial, threshold, and logistic elements).

To improve the process of “pondering point clouds”, projection pursuit and inductive modeling techniques could be added to an interactive graphic environment to effectively extend the scope of one’s judgment a few dimensions.⁸ The power of visual techniques suggests that perhaps a chief strength of such an approach would be its step “backwards” to critical reliance on the analyst in the structure search and removal process.

3.2. Projection Pursuit

In high dimensions, visual strategies for discovering structure include principal components, multidimensional scaling, *plot matrices*, a “*grand tour*” of the data (Asimov, 1985), and automated projection pursuit (PP). Both principal components and multidimensional scaling provide a view of the data in lower dimension (normally 2), although multidimensional scaling starts with ordinal proximity matrix while principal components uses the data matrix. Though a plot matrix, or pairwise arrangement of scatterplots, is only a set of two-dimensional views of the data, this is often sufficient to provide useful insight. Also, graphic programs are increasingly enforcing case links between graphs (through symbols, colors, *brushing*, etc.; e.g., Cleveland and McGill, 1988)

⁸As proposed for the Inductive Data Exploration Algorithm, IDEA (Elder, 1992b)

-- extending the effective dimension of perception. The grand tour strategy presents the analyst with a single 2- or 3-dimensional projection of the data where the view (projection matrix) smoothly transitions in such a way that one is eventually presented with “all sides” (low-dimensional views) of the point cloud. Interesting views can be saved for later analysis, or can cause the tour to be postponed for local exploration.

In high dimensions, a grand tour can take overlong, so it is useful to have the computer score candidate views (noting low entropy, high clustering, unusual skewness or Fisher information, etc.), and present, or steer the tour through, only the best views. Such a *projection pursuit tour* has recently been implemented in the workstation package, *XGobi* (Swayne, Cook, and Buja, 1991). The PP index could also include factors such as *interpretability* (Morton, 1989) or equations for complexity, or novelty -- the best choice can depend critically on the application. In Figure 3, a 2-dimensional point cloud of two classes is plotted, surrounded by some (histogram) projections of the data. The index employed (to be minimized over 180° of θ) is the number of misclassifications resulting from the best θ threshold for that view.

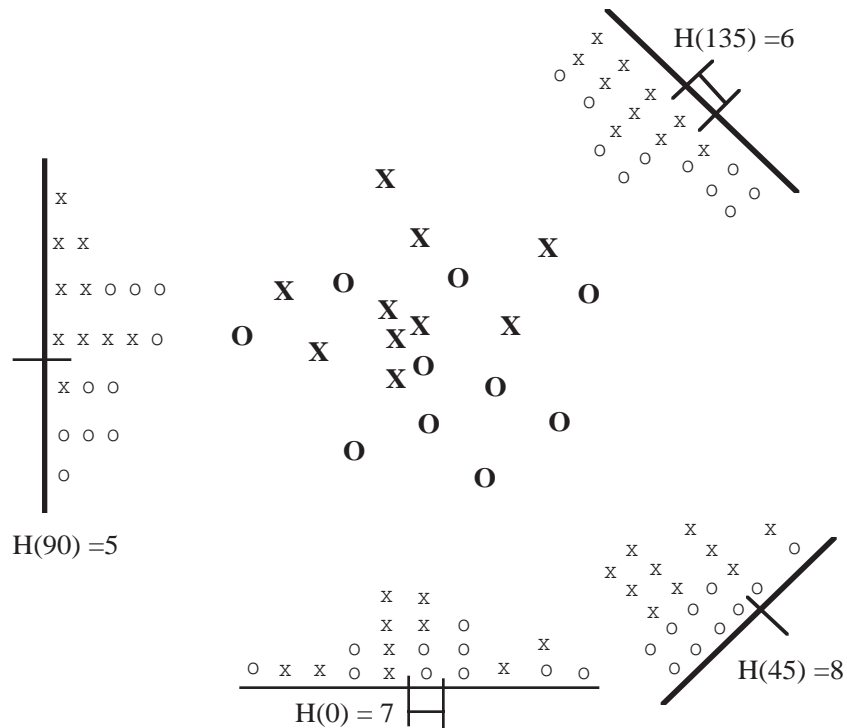
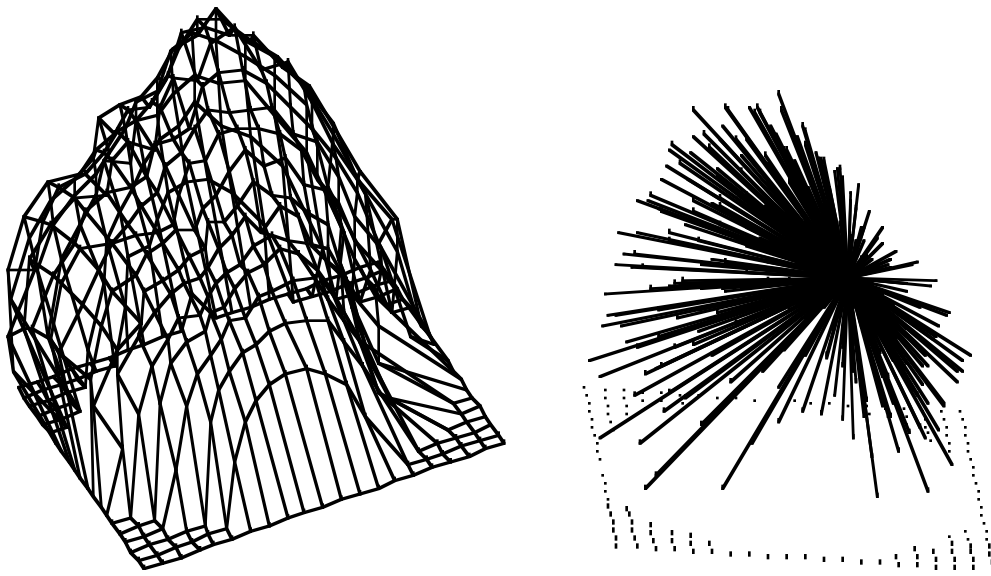


Figure 3. 1-Dimensional Projections of Sample 2-Dimensional Data.

Angles of projection are scored according to the number of misclassifications of the best threshold.

Note that the search space wraps around ($\theta \equiv 180^\circ + \theta$), causing a redundancy which could save search time if exploited (though its complexity has apparently prevented doing so, thus far). For example, when reducing 3-dimensional data to a 1- or 2-dimensional view, the projection axis can be represented as a point on a unit hemisphere, as in Figures 4a and 4b, where the *score* (PP index value) for a location is represented by its distance from the center of the sphere. However, for search continuity, the most distant points on the base of the dome should actually be connected (a 3-d wrap-around). As indicated by Figures 4a and 4b, projection spaces can be quite *rough*; they also can vary in shape widely due to changes in the projection index (e.g., Cabrera and Cook, 1992). Therefore, a sophisticated global search should be employed to discover and rank the projections to be presented to the analyst (though only local searches seem currently to be in use).



Figures 4a,b. 3-Dimensional Views of a Projection Pursuit "Score Surface"

3.3. Remarks

Automated induction and projection pursuit algorithms, despite their variety, are not (and may never be) competitive with analysts in perceiving structure in noisy data -- in the low dimensions of human experience. Of course, in high-dimensional applications, where the structure of the underlying “data generating machinery” is unclear, automated induction algorithms are quite useful. Often through nothing more than trial and error, they can construct compositions of small functions into empirical approximations of a function interpolating (and smoothing) the training data. However, automated methods must handle a host of data hazards, including influential and outlying cases, and collinear and redundant variables. Most importantly, empirical methods must avoid *overfit*; that is, the tendency to fit the noise, as well as the signal, in the data (as discussed below), since everything one does will tend to improve training performance. Also, the typical summary statistics monitored by most algorithms do not always reveal key data relations, as demonstrated by Anscomb’s (1973) quartet of data series.

Fortunately, sophisticated interactive graphic environments are becoming more widespread. Such an environment could eventually form the base of a useful interactive inductive tool, augmented by a collection of automated algorithms of two forms: *structure exploration* (like PP) and *structure removal* (i.e., approximation). That is, in the background⁹ of one’s graphical manipulations, the computer could continually search for, and queue up, a list of low-dimensional views of the data which score well by any of a number of PP criteria. Then, when a significant bump or trend (or blip or ...) is found, a variety of approximative methods could be employed (kernels, polynomials, etc.) to fit it. The model could then be removed from the data (leaving a residual, and requiring an update of all extant views) and the process repeated until only noise appears to remain. (To avoid overfit, it is anticipated that a *complexity budget* even more strict than usual must be adhered to, because of the increased approximation flexibility.)

In this manner, perhaps the muddled view the computer has of high dimensions can be enough to lead us “walking pattern recognizers” to the select low dimensional views in which all is made clear.

⁹Our computers spend most of their time waiting ...

4. MODEL SELECTION

For “plastic” models (Section 2.2), the computer must select the final structure; thus, an objective *score function* for grading candidate models is required. To generalize well from the known data to new situations, a compromise must be made between the accuracy on training data possible with complex structures and the *robustness* on new data characteristic of simple ones. Modern statistical theory helps automate this tradeoff between simplicity and accuracy but, at some level, it is a judgment call. Though models with many parameters may best fit the known data, simpler models usually estimate new cases more accurately. In the 14th century, this observation was codified as the principle of *Ockham’s Razor*, which can be stated: “an explanation of the facts should be no more complicated than necessary.” In other words, if two theories about a phenomenon seem equally accurate, choose the simpler -- it is likely to be closer to the truth. In empirical model building, this *principle of parsimony* helps avoid the danger of *overfitting* the data; that is, forcing down the model’s errors on known data at the expense of its performance on as-yet unseen data (which is its true application).

Overfit is especially dangerous when *extrapolating* outside the bounds of the data (as when forecasting), but can also happen when *interpolating* within them. To illustrate, suppose we modeled the sequence: $Y = \{ 1.1, 2.9, 3.2, 3.9, 5.2, 5.8 \}$ as a function of: $X = \{ 1.0, 3.2, 3.0, 4.0, 5.0, 6.0 \}$ Considering accuracy alone, we would get the solid line and 5th-order equation (zero error; $R^2 = 1.00$) of Figure 5, rather than the more reasonable (and nearly as accurate) dotted line of $y = x$. In general, a data set containing N samples of a function can be fit exactly by an equation with N terms (i.e., of order $N-1$).¹⁰ But, as can be seen in Figure 5 for regions near $x = 0$ or 2 , this resulting equation can be dangerous to use for unknown inputs. Thus, complexity must be regulated for an induced model to be useful.

In order of sophistication, the four ways to regulate model complexity are:

- 1) *Set by Fiat*: Fix the model terms and search for parameters to minimize a score on the training data.
- 2) *Set by Goal*: Choose the simplest model which reaches or exceeds a training accuracy goal.
- 3) *Reserve Data*: Split the available data into subsets; train the candidate models on the first set, and choose the model which is most accurate over the second set.

¹⁰Note that multiple-valued *relations*, such as \sqrt{x} , can have more than one output value for a given input vector.

- 4) *Penalize Complexity*: Measure model complexity by the number of parameters, K and, using *all* the data, choose the model which is best according to a function of K , the training error, and (perhaps) the level of data noise.

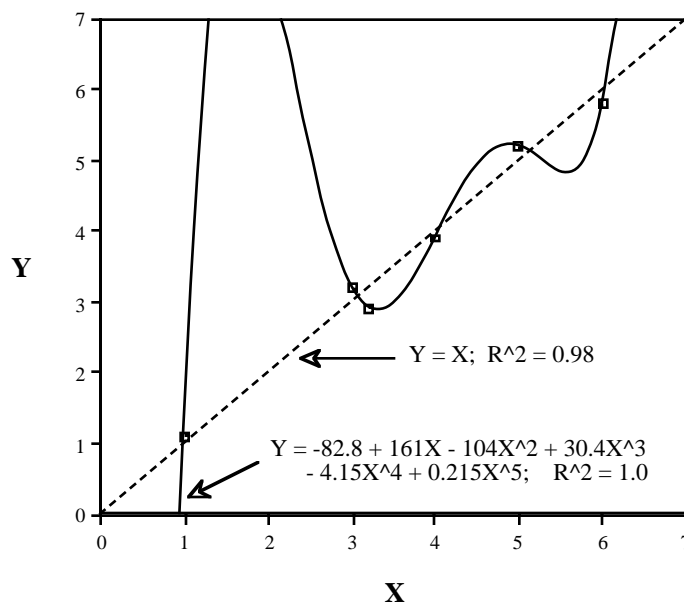


Figure 5. Example of Model Overfit.

Though accurate on training data, overly complex models can poorly estimate new points.

The structure of *knobby* models is set by fiat, and the cutoff to *stretchable* models is often due to an *a priori* accuracy goal. The candidate models saved on each layer of a *plastic* GMDH model are selected according to their “checking set” accuracy (method 3), and *plastic* ASPN models employ a complexity penalty, as described in the next Section. (When several testing subsets are employed to get a distribution of results, reserving data is known as *cross-validation* (Stone, 1974) -- or, with slightly different sampling strategies, the *jackknife* or *bootstrap* techniques of estimation (e.g., Diaconis and Efron, 1983).) Prudent researchers often combine methods 3 and 4 above, using the complexity penalty with most of the data to explore data transformations and to train the models, and confirming promising results with the reserved data.

In this section, in order to provide specific examples of the issues involved in model selection selection, we emphasize linear regression (LR), which remains the dominant statistical modeling technique. We note, however, that the issues raised are also common to other methods, although the specific measures used for model selection can vary. Under LR, the classical complexity penalty approach (though not usually thought of as such) is the analysis of variance (ANOVA) F-

test. We will briefly describe the F-test technique, note how it is usually misapplied in ANOVA analyses, and motivate the use of more modern selection criteria. The first component of all criteria however, is a measure of model accuracy.

4.1. Model Accuracy

A linear regression (LR) model finds the parameter values which minimize the mean squared error

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (Y_i - \hat{Y}_i)^2 \quad (2)$$

or average squared distance from the true vector, Y , to its estimate, \hat{Y} , for the N known cases. The “linear” in LR derives from the estimated response, \hat{Y}_i , being a linear function of the *parameters* of the model; it is not necessarily linear in the stimulus variables. For example, the LR equation, $\hat{Y}_i = w_0 + w_1 t + w_2 t^2$, is linear in the parameters, $\mathbf{w} = (w_0, w_1, w_2)$, but quadratic in the stimulus variable, t . (For nonlinear regression, consult, e.g. (Seber and Wild, 1989).)

The square root of MSE given in (2) is written as rMSE (root Mean Squared Error) or SEE (standard error of the estimate), and approximates the standard deviation of the model’s error, σ_e . Thus, when the responses, \mathbf{Y} , are normally distributed about the line (as assumed under LR), we expect two thirds of the cases to fall within $\pm\sigma_e$ of the predicted values, 95% within $\pm 2\sigma_e$, etc.

SEE is in the units of the data so it is an *absolute* measure. A popular *relative* measure of accuracy is the *coefficient of determination*:

$$R^2 = 1 - \frac{\text{MSE}}{S_y^2} \quad (3)$$

where S_y^2 is the estimated (sample) variance of Y :

$$S_y^2 = \frac{1}{N} \sum_{i=1}^N (Y_i - m_y)^2 \quad (4)$$

for sample mean Y value, $m_y = \frac{1}{N} \sum_{i=1}^N Y_i$. Thus, a high R^2 (near 1) suggests most of the variance

in the data can be “explained” by the model; whereas, a low value (near 0) indicates that the model is no better than the best constant, m_y . (Of course if the number of features in the regression equation, K , is high relative to the number of data points, N , then the R^2 value can be

unrealistically high. Hence many researchers use adjusted R^2 , or other complexity-adjusted metrics, as discussed below.)

The computational advantage of using squared error often tempts researchers to make the Gaussian error assumption in situations for which it is not appropriate. There are many other error metrics in use, and each implicitly or explicitly correspond to assumptions about the distributions of the model error. Least squared error is in the family:

$$L_P = \frac{1}{N} \sqrt[P]{\sum_{i=1}^N |Y_i - m_y|^P}, \quad P \geq 0 \quad (5)$$

so it is denoted L_2 . Employing the least absolute deviations (LAD) metric, L_1 , corresponds to assuming that the likelihood of an error is an exponentially decreasing function of its magnitude. As this is a slower asymptotic decrease to zero than the Gaussian curve (the distribution has “fatter tails”), use of L_1 leads to models which are more *robust* than those for L_2 (e.g., Bloomfield and Steiger, 1983). Large errors (such as produced by some outliers) have less influence under L_1 as they are considered to randomly occur less rarely; therefore, the model doesn’t as readily overfit parameters to eliminate them. Also, since the LAD problem can be mapped into a Linear Program (LP), reasonably fast solution algorithms have been available since Charnes, Cooper, and Ferguson (1955) showed the LP could be solved via the simplex algorithm (Bloomfield and Steiger, 1983).

Note that P in (4) can take on other non-negative values, including real numbers (since, to compare models having a fixed number of cases, N , the average and the P^{th} -root calculations are not actually required). L_0 is equivalent to the “hit/miss” criterion, counting the non-zero errors (misses) of any magnitude. The L_∞ metric leads to the “minimax” criterion; that is, the parameter set is selected which has the smallest maximum error (regardless of what the rest of the distribution looks like). Typically, however, a value for P between 1 and 2 is employed.

Accuracy criteria of the L_P family all raise their penalty for errors (by different relative amounts) as their magnitude increases. However, model robustness can be improved by actually *decreasing* the influence of cases at some level of their deviation from the model. Two examples of criteria which eventually ignore likely outliers (i.e., cases with sufficiently large standardized residuals, z), are Andrew’s sine:

$$A(z) = |z| \sin\left(\frac{\pi z}{c}\right) \quad \text{when } z > c; \quad (\text{otherwise } 0) \quad (6)$$

and Tukey's biweight

$$T(z) = (z(1-(z/c)^2))^2 \text{ when } z > c; \text{ (otherwise 0)} \quad (7)$$

for a smoothing constant, c .¹¹

4.2. Significance Tests

It is well known that adding terms to any model will improve its accuracy over the training data. In LR, for example, expansions raise R^2 and lower SEE. The key issue to resolve, however, is whether the improvement is significant enough to justify the added complexity. Could it instead be due to chance alone? Under the strict LR assumptions about the data, the method of hypothesis testing can answer these questions.

Assume that the "true" model is known, having K_M terms (degrees of freedom) and coefficient of determination R^2_M . (Typically, the most complex model one is willing to consider stands in for the true model.) A simpler model, called the *hypothesis*, is to be compared, having K_H terms and an accuracy of R^2_H (where $K_H < K_M$ and $R^2_H \leq R^2_M$). Since the hypothesized model is a subset of the full model, we are testing the conjecture that the $K_M - K_H$ terms not in the former are actually zero. This hypothesis can never be accepted with certainty, but we can quantify the probability, given the data, that the improved accuracy of the larger model is due to chance alone. The threshold on this probability (e.g., 1 to 10%) is the *significance level*, α , of the test.

In brief, we are given the results of a full model, and hypothesize that a simpler model is a better explanation for the data; i.e., that the higher accuracy of the full model is due only to its having more parameters to adjust and not to the inherent importance of those terms. If the accuracy difference has more than a slight chance (α) of being real, then we (reluctantly) reject the hypothesis represented by the simpler model. Conversely, if the accuracy difference is small enough, we employ the simpler model. (In the vocabulary of hypothesis testing, we "cannot reject" the hypothesis). Statistically:

$$F = \frac{(R^2_M - R^2_H) / (K_M - K_H)}{(1 - R^2_M) / (N - K_M)} \quad (8)$$

¹¹If the errors happen to be Gaussian after all, the best value for c is 2.1π for $A(z)$ and 6.0 for $T(z)$ (Press, Flannery, Teukolsky, and Vetterling, 1988).

is compared to the F-distribution table value, $F_{\alpha}(K_M - K_H, N - K_M)$. If $F > F_{\alpha}$, use the full model; otherwise, employ the hypothesis. The significance level, α , is the probability (always present at some level) of falsely rejecting the hypothesis using this procedure.

Denoting the SEE of the model as σ_M , and using (3) allows F to be written in terms of the differences of the sample error variances of the hypothesis and model:

$$F = \frac{(S^2_H - S^2_M) / (K_M - K_H)}{S^2_M / (N - K_M)} \quad (9)$$

In practice, one would first define the full model, then test its subsets (representing various hypotheses) against it. The hypotheses can be arranged hierarchically in a *lattice*, with the full model at the top. Starting at the simplest hypothesis of interest, and working up the ladder as necessary, the *first submodel* (e.g., the lowest order candidate) *that cannot be rejected* is selected as the working model.

4.3. ANOVA Model Selection

In the typical application of the analysis of variance (ANOVA) technique however, analysts start with an hypothesis and act as if the full model were just one term larger. The above F test is employed and, if the hypothesis is rejected, the analysis continues in a stepwise fashion with the larger model as the new hypothesis competing with a the new “full” model having one additional term. The process halts when a hypothesis cannot be rejected. With “stretchable” polynomials, it is the model order (highest power) that is incremented each time. Let K_M denote the number of terms in the model; then $K_M = P + 1$ for a model of polynomial order P , and $K_M - K_H = 1$.

Therefore, the F-test (9) becomes

$$F = \frac{S^2_H - S^2_M}{S^2_M / (N - K - 1)} \quad (10)$$

and is compared to $F_{\alpha}(1, N - P - 1)$. If the hypothesis is rejected (F is too large), the model becomes the hypothesis, P is incremented, and the process is continued. If, however, the hypothesis cannot be rejected (F is small enough), it is used.

Though the stepwise ANOVA method is widely employed and is available in most statistical packages, the incremental approach has long been recognized as an incorrect application of the underlying theory (e.g., Pope and Webster, 1972). Rather than comparing each hypothesis against a full model (a *fixed* standard), each is compared to its immediate successor (a *sliding* standard). To be selected by either technique, a model has to be judged better than its immediately

simpler competitor. But, the stepwise ANOVA method imposes the additional constraint that all the model's simpler versions have to be better than their subsets, in turn. Though this might be common, insistence on this added constraint can impair the quality of the models selected.

Fenster, Dolan, and Elder (1992) demonstrate this when estimating future shoreline position for a coastal setback application. Figure 6 reveals how a constant model (μ_y) would have been incorrectly chosen by the stepwise ANOVA method to describe the history of shoreline position at a transect along a Texas island. Because the linear model is no significant improvement over the constant, the stepwise procedure stops, despite the clear quadratic shape of the series (visible in this low-dimensional data space). The proper F-test method, however, correctly identifies this quadratic model, revealing the oldest point to be an outlier for the purpose of linear prediction.

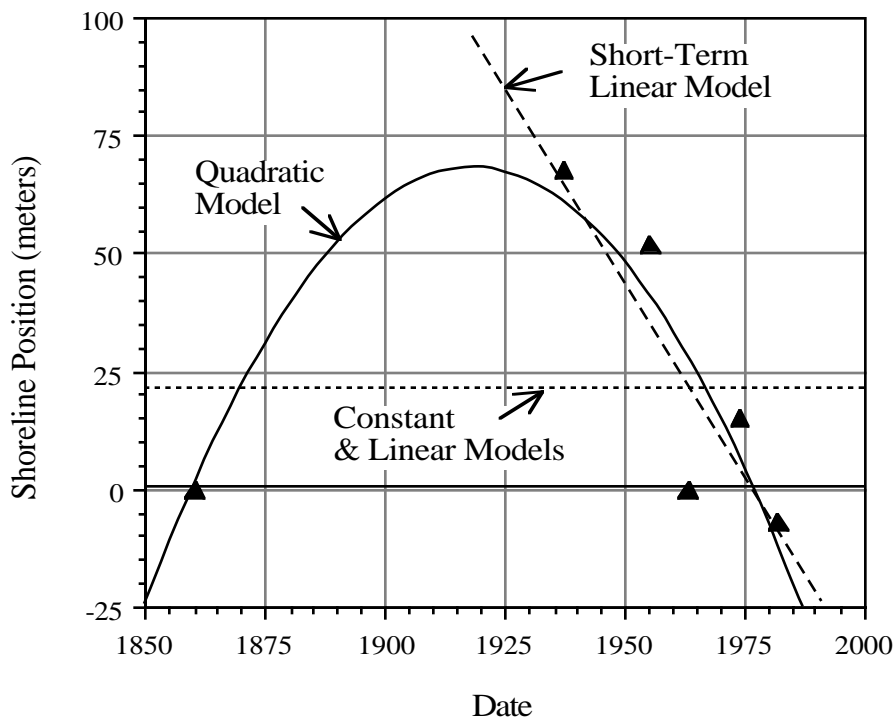


Figure 6. Model Selection Example (after Fenster, Dolan, and Elder, 1992).

Though a quadratic model is best, iterative hypothesis-testing halts at the constant model ($Y=22$), because the linear model is virtually no improvement.

4.4. Explicit Complexity Penalties

In the F-test comparisons, models are judged by their *relative* accuracy differences; for instance, in (8), the change in variance (or, accuracy difference) is scaled by the variance of the full model before comparison with an F-table entry. Relative measures are scale invariant, which is not necessarily an advantage when the natural “noise level”, or underlying data precision, is known. This issue is analogous to the problem of scaling a graph to display information appropriately. But, the best *visual* tradeoff depends on the scale with which the data are presented. Use of relative accuracy as a yardstick (whether to a fixed or sliding standard) is like sizing each graph by the data spread, rather than setting the scale objectively.¹²

When the measurement precision for the data is known and when consistency is desired between results, it is better to fix the scale; that is, to select models according to a complexity penalty method which relies on an estimate of “underlying noise level” in the data. Two of the better known and widely used such criteria are: Mallows’ C_p (Mallows, 1973) and Akaike’s Information Criterion, AIC (Akaike, 1973). A criterion similar to C_p , which is employed by a progression of polynomial network algorithms (described in the next Section), is Predicted Squared Error, PSE (A. Barron, 1984). These and many other techniques score models according to both their simplicity and accuracy, providing somewhat different tradeoffs when selecting the optimum from a set of candidates. The Minimum Description Length (MDL) criterion (Rissanen, 1978) is one of the more conservative members of this family of model criteria (that is, it favors simplicity over accuracy more than most metrics).

For linear regression, a simple version of MDL can be written:

$$\text{MDL}_K = \text{MSE}_K + \frac{\ln(N)K\sigma_p^2}{N} \quad (11)$$

for a model with K terms, based on N data points, with a given “prior estimate of noise variance”, σ_p^2 (supplied by the analyst or estimated using a contributory technique). The error component, MSE_K , decreases as terms are added, yet the second component, a complexity penalty, increases with K . The model with the minimum sum of the two components is declared the best. Dordick (1988) describes the development and somewhat radical philosophy of MDL, and Elder and Finn

¹²For an analogous effect of scale on intuition, see (Cleveland, Diaconis, and McGill, 1982) where perceived *correlation* is shown to increase with scale magnification.

(1991) explain its use for inductive modeling.¹³ Further details of its use for many types of inference applications appear in (Rissanen, 1989).¹⁴ Here we outline the concepts of MDL (based on the discussion in Elder and Finn, 1991).

4.5. Minimum Description Length

As Einstein is supposed to have said, "A theory should be as simple as possible, but not simpler". The basic assumptions of MDL are:

- 1) The more complex something is, the harder it is to describe, and
- 2) The simplest of equally plausible hypotheses is the preferred.

Thus, the *shortest* description of the data (to some precision) is taken to be the *best understanding* of its underlying generating mechanism. (This principle of inferring physical laws from their simplest explanations is central to the scientific method.) The metric is a quantification of "Ockham's razor" -- the mythical tool for paring away superfluous accretions in a hypothesis, and is designed to select rationally between competing models.

Traditional statistical methods for comparing models of different types and sizes are primarily *ad hoc*. No usual technique, for instance, can fairly decide between a moving-average model having three parameters, and a more accurate polynomial model with four parameters (especially if the errors are measured according to different metrics). But the MDL criterion purports to be a *universal yardstick* able, essentially, to "compare apples and oranges". With such a metric, the analyst must only insure that a sufficiently rich class of models is considered; that is, that the competing models are numerous and varied enough for at least one candidate to describe the data well -- a less daunting task than specifying every model term, as required in the usual approach.

These properties of MDL are due to insights from *communication theory*, where a model is viewed as a summary, or shorthand representation, of a data set. It is a *compressor* of the data, and the best models are those which (accurately) compact the data the most. Consider sending a message by telegraph wire, where the message is a stream of numbers: a *response* vector, Y . To

¹³However, the former reference does not give an equation for MDL, and the latter (inadvertently) provides only a special-purpose one, for use when the response is standardized (scaled to have a variance of 1) and the noise variance, σ_p^2 , is taken to be the very conservative value of half the response variance.

¹⁴MDL's chief developer, Jorma Rissanen, traces his inspiration to the concept of algorithmic complexity (due to the mathematician Kolmogorov), and to Bayesian predictive densities (e.g., Aitchison, 1975). Significant other contributions to MDL include (Wallace and Boulton, 1968) and (A. Barron, 1985).

send the values (at an agreed upon precision) one assigns codes to each possible number (each symbol in our "alphabet"). If the symbols are equally likely, their codes will have roughly equal length; however, if some symbols are common, we can save space by using *short codes for frequent symbols*. Intuitively, this is what Samuel Morse did with his telegraph code by, for instance, assigning a single dot "." to the letter "e" and three dots and a dash "···-" to the more rare "v". Such codes are efficient, unless the message is very unusual. What defines "usual" is a probability density of the symbols (in this example, the likelihood of hitting a given letter when randomly striking a pin into a page of text).

How does this apply to models? A measure of the "flatness" of a probability distribution, $p()$, is its entropy

$$- b \sum_i p(Y_i) \ln(p(Y_i)) \quad (12)$$

(for the discrete set of symbols, Y , indexed by i) where b is a constant denoting the arbitrary choice of base for the logarithm. This reveals how efficiently the vector can be encoded and transmitted; in fact, without outside information, this rate cannot be surpassed. But, suppose both sender and receiver have access to a set of stimulus variables, X ; then it's likely that the *message of the data* can be sent more concisely through encoding Y given X (denoted Y/X) than by encoding the Y values directly. Indeed, if there is a clear relationship between the known variables, X , and the unknown (to the receiver) Y , then just that relationship (model) needs to be transmitted. The receiver can then use the model (and X) to recreate Y unambiguously. So at one extreme, when high a degree of inherent order exists, a model can succinctly summarize the data. At the other, when Y is essentially random, a list of the (encoded) values is the most efficient way to communicate them. Thus, the minimum description length of Y given X , $MDL(Y/X)$, is an indication of *how random* Y is, given what we know (X).

Real data tend to exhibit some order, but the patterns are clouded by incomplete and "noisy" information and by complex interactions among contributing factors. Models can only approximate the response variable, Y , so encodings consist of two parts:

- 1) A description of the model, and
- 2) An encoded list of its errors.

Errors can be transmitted more efficiently than the original numbers, since they typically cover a smaller range of values; thus, fewer symbols are needed (for a given precision) and their density is more peaked (their entropy less). In general, the more complex the model, the better the

fit, so the easier the errors are to transmit; however, these savings are counteracted by the extra difficulty in describing the model itself. Using these two components of the code (model description and error description) there emerges a natural way -- total description length -- to trade off model simplicity and accuracy. The best candidate model has the shortest code; that is, leads to the most efficient encoding of the *message* in the data.

4.6. Other Complexity Criteria

MDL is one of a class of complexity penalty criteria for model selection. Some others (e.g., ICOMP (Bozdogan, 1988) and Stochastic Complexity (Rissanen, 1989)) employ determinants of the design matrix, but for scoring LR models, most can be expressed in such a way as to rely only on the sum of squared errors, SSE, the number of parameters, K , the number of cases, N , and (perhaps) a prior estimate of the error variance, σ_p^2 . Other complexity criteria include, in rough order of conservatism (i.e., protection against overfit):

$$\text{adjusted Mean Squared Error, } aMSE = \frac{SSE}{N-K} \quad (13)$$

$$\text{Akaike's (1970) Future Prediction Error, FPE} = aMSE \cdot \frac{N+K}{N} \quad (14)$$

$$\text{Akaike's (1973) Information Criterion, AIC} = \ln(\text{MSE}) + \frac{2K}{N}, \text{ where } \text{MSE} = \frac{SSE}{N} \quad (15)$$

$$\text{Schwarz's (1978) Bayesian Criterion, SBC} = \ln(\text{MSE}) + \frac{K \ln(N)}{N} \quad (16)$$

$$\text{Mallow's (1973) } C_p = \frac{SSE}{\sigma_p^2} - N + 2K \quad (17)$$

$$\text{A. Barron's (1984) Predicted Squared Error, PSE} = \text{MSE} + \frac{2\sigma_p^2 K}{N} \quad (18)$$

$$\text{Sawa's BIC} = \ln(\text{MSE}) + 2q(K+2-q)/N, \quad \text{where } q = \frac{\sigma_p^2}{\text{MSE}} \quad (19)$$

-- all of which are to be minimized. (Other criteria are members of this family, such as adjusted R^2 , Parzen's (1977) CAT; Craven and Wahba's (1979) Generalized Cross-Validation, GCV; Amemnya's (1980) Prediction Criterion, PC; Hockey's S_p ; Hurvich and Tsai's (1989) Corrected AIC, AIC_C ; and metrics proposed by Shibata (1980), and Breiman and Freedman (1983). Some

are redundant; for example, adjusted R^2 is a reverse version of (13), equal to $1 - \frac{aMSE}{S_y^2}$, and C_p (17) is equivalent to PSE (18) for LR.)¹⁵

To illustrate use of these criteria, they were applied to the most accurate (L_2) models of each size, 1 to 18 terms, of the rocket temperature estimation application discussed in (Elder [this volume], Section 4.5 on branch and bound modeling).¹⁶ Table 2 provides the model/criterion matrix. The *a priori* error variance estimate, σ_p^2 (needed for the PSE, MDL, and BIC criteria) was set to 3.54, which is half the variance obtained by nearest neighbor (NN) estimation.^{17,18} The use of the NN error, as opposed to a "full model" error variance,¹⁹ seems more robust. The issue of defining a "full model" is avoided; also, the error estimate, while being inferred (like the model) from the data, comes from a local, contributory technique, and is somewhat orthogonal to the global, consensus estimates of LR.

In Table 2, the global optima according to each criterion are boxed, and any local optima (scores lower than those both above and below) are underlined. (The existence of the latter demonstrate that any minimizing search should continue a bit past the point of complexity where a criterion ceases to improve; often there are later model "breakthroughs".) Note that the criteria employing the NN σ_p^2 estimate choose smaller optimal models; that is, they are more conservative with respect to model complexity (probably a good trait on this small and dichotomous data base). (The estimates are conservative primarily because the NN σ_p^2 is roughly equivalent in SSE to the two-term model.) Based on these results, the five-term model looks to be the choice for new data:

¹⁵For the problem of selecting the optimal model order, some of the criteria have been shown to be *asymptotically efficient*, and some others to be *consistent*. One cannot be both (Schwarz, 1978).

¹⁶The data, from (Lloyd and Lipow, 1962), are reproduced in (Elder [this volume], Table 9), and the forms of the simplest ten models appear in (Elder [this volume], Tables 10 and 11).

¹⁷This heuristic, suggested by A. R. Barron (1987, personal communication), is due to the NN estimator having an error probability of less than twice the Bayes (optimal) probability (Cover and Hart, 1967).

¹⁸The particular NN algorithm used the original variables as dimensions and measured distance according to the L_1 criterion with standardized $\left(\frac{x - m_x}{S_x}\right)$ axes. Each case, in turn, is compared to the others; the output value of the closest case, Y_{nn} , estimates the recorded value, Y , and the resulting errors are accumulated. (When distances tie, the observation which appears first in the data base is employed, though averaging output values could be done.)

¹⁹As called for by the C_p criterion, in its original form, and by hypothesis testing methods; see the ANOVA discussion above.

most of the criteria were at least locally minimal there, and the least permissive (MDL and BIC) identified it as best overall.

Table 2. Criteria Scores for the Most Accurate Model of Each Size, K

(*with $\sigma_e^2 = 3.54$, half the Nearest Neighbor error variance)

K	SSE	aMSE	FPE	AIC	SBC	PSE*	MDL*	BIC*
1	325.03	14.13	14.72	64.54	2.74	13.84	14.01	2.67
2	163.16	7.42	8.03	50.00	2.18	7.39	7.74	2.07
3	88.76	4.23	4.76	37.39	1.71	4.58	5.10	1.63
4	69.70	3.49	4.07	33.59	1.60	4.08	4.78	1.55
5	56.28	2.96	3.58	30.45	1.51	3.82	4.69	1.54
6	52.12	2.90	3.62	30.61	1.57	3.94	4.98	1.64
7	38.85	2.29	2.95	25.56	1.41	3.68	4.90	1.72
8	29.13	1.82	2.43	20.65	1.25	3.57	4.96	1.92
9	20.45	1.36	1.87	14.16	1.03	3.51	5.07	2.21
10	16.86	1.20	1.71	11.53	0.97	3.65	5.39	2.57
11	14.34	1.10	1.61	9.64	0.94	3.84	5.75	2.98
12	12.37	1.03	1.55	8.09	0.93	4.06	6.14	3.42
13	12.13	1.10	1.70	9.62	1.04	4.34	6.60	3.98
14	11.40	1.14	1.81	10.13	1.11	4.61	7.04	4.56
15	11.32	1.26	2.04	11.96	1.23	4.90	7.50	5.19
16	11.12	1.39	2.32	13.54	1.35	5.18	7.96	5.83
17	11.08	1.58	2.70	15.45	1.48	5.48	8.43	6.47
18	11.06	1.84	3.23	17.41	1.61	5.77	8.90	7.11

4.7. Implications of the Computer Age

The enormous flexibility and scope of model searching made possible by the computer has led to something of a breakdown of traditional model selection criteria. Clearly, the best representative of thousands (or millions ...) of candidate models has a different nominal performance distribution than that of any single model, so one's standards of "significance" must be raised to avoid overfit. That is, the chance of finding a model with attractive accuracy, but which has no predictive power on new data (i.e., is randomly good) increases with the number of candidate models one explores.²⁰ Traditional significance tests, which assume a model is being examined in isolation, are therefore not appropriate, despite their continued wide use. This phenomenon (huge candidate model sets and limited data -- a result of the computer age) violates the assumptions of asymptotic analysis, and may be partly responsible for the growing conservatism of model selection metrics.

The model selection stage of data analysis can invalidate the standard significance checks performed by induction procedures. For example, with LR, the F-statistic (Sections 4.2 and 4.3) is known to be improper for stepwise methods (Pope and Webster, 1972), "best subsets" tend to have an inflated value of R^2 (Rencher and Pun, 1980), and prediction MSE estimates can be strongly biased (Breiman, 1988). Hurvich and Tsai (1990) note that

... conditionally on the event of having selected a particular model, the distribution of the data may be substantially different from their unconditional distribution. (p. 214)

and blame much of the problem on employing the same data for both structural identification and parameter inference. Similarly, Hjorth (1989) argues that

... the evaluation of a selected model can not be based on that model alone, but requires information about the class of models and the selection procedure. (p. 101)

(leading to the puzzle of designing model selection metrics to include the effect of model selection!) Hjorth (1989) also warns that

...selection minimizing a criterion will cause underestimation of this criterion even if the estimators are locally unbiased. (p. 111)

²⁰This observation leads to philosophical puzzles. For instance, if the first model examined remains the best found (so far) is our confidence in its quality strengthened or weakened as the list of rejected alternatives grows? The former perspective is bolstered by the fact that the model is so hard to dethrone, and the latter by the realization that its performance should be judged by a best-out-of- M standard, for growing M . (What if the winner were the last model, rather than the first?)

Other data analysis procedures also affect model confidence. Case diagnosis (leading to weighted or deleted cases) changes metrics; for example, weighted LR invalidates R^2 measures (Willet and Singer, 1988). Also, removing low-order model terms (e.g., carving in polynomial networks) can make the model “not well-structured” (Peixoto, 1987), in the sense that linear transformations of the inputs (such as a change of scale) can lead to the selection of a different model structure. (In those situations in which the model form is to be interpreted therefore, one should consider retaining lower-order terms.) Lastly, transformations of the variables, such as Box-Cox normalizations, can inflate the variance (uncertainty) of some model parameters (Bickel and Doksum, 1981).

Finally, it is the experience of several induction researchers that nonlinear parameters are more powerful than linear ones, and should be considered larger contributors to model complexity.

[The results of Hastie and Tibshirani (1985)], together with those of Hinkley (1969, 1970) and Feder (1975), indicate that the number of degrees of freedom associated with nonlinear least squares regression can be considerably more than the number of parameters involved in the fit. (Friedman and Silverman, 1989; p. 9)

Clearly, adjusting for the added complications introduced by modern model selection strategies (including massive candidate model sets, nonlinear parameters, case weighting, and variable transformation) requires rather conservative (larger) complexity penalties, or careful application of sample re-use techniques (e.g., cross-validation, jackknife, or bootstrap procedures). We described the former approach in Sections 4.4-4.6; a good example of the latter can be found in the recent Regression Analysis Tool (RAT) prepared by Faraway (1991) in the Lisp-Stat language (Tierney, 1990). There, regression analytic steps are characterized as functions acting on regression models and returning regression models. A multi-stage procedure -- including skewness and heteroskedasticity transformations, outlier and influential point removal (and potential re-introduction), feature creation, and backward term elimination -- is performed multiple times on different subsets of the data, and the resulting *distribution of models* is analyzed. Also, the distribution of individual model parameters (if nearly ubiquitous) or of model predictions for a particular data case, can be examined for a graphic indication of their precision and certainty. An assessment of distributions could even be done concurrent with data analysis to signal the effectiveness of analyst actions, and perhaps indicate when to stop the process.

Yet, RAT simulations are quite time-consuming. Though a single regression stage (for a few hundred data points) took only 2 seconds on a workstation in (Faraway, 1991), each entire simulation required 2 days! More importantly, though the multi-stage procedure was designed to

approximate the behavior of a human analyst, only deterministic steps -- i.e., those which can be completely characterized and programmed *a priori* -- can be included (excluding, for example, the integration of visual information). Still, the technique points a way toward assessing the impact of computer-intensive methods on model selection (and on their predictions), and is worthy of continued attention.

4.8. Remarks

Modern induction techniques (such as the polynomial network methods discussed in the next two Sections) often induce the structure, as well as the parameter values, from sample data. This brings to the fore questions of how best to measure accuracy, complexity, and their tradeoff. The goal must always be *generalization*; that is, performing well on new cases arising from the same system as the known "training" cases. To attain this, candidate models must be powerful enough to fit the old data, but simple enough to avoid overfit on the new. Traditional significance tests for evaluating the utility of a given model term are weakened or invalidated by modern analysis procedures, such as extensive model search, variable transformations, case weighting, and consideration of nonlinear model terms. In addition, it is becoming clear that inference must even be adjusted to account for any prior data analysis performed. Use, therefore, of conservative complexity penalties and re-sampling techniques to identify good models should only increase as the scope of induction tasks left to the computer continues to grow.

5. POLYNOMIAL NETWORK ALGORITHMS

When computing power was expensive, analysts were forced to employ rigid parametric models even when the relationships between stimulus and response variables were not clear enough to warrant the necessary prior specifications. Armed with a tireless “computing assistant” however, researchers have increasingly turned to nonparametric, iterative, and exploratory techniques to model difficult data. This Section explores a set of these powerful induction techniques that exploit the available cycles on modern machines to create effective models.

Polynomial networks combine many of the strengths of ANNs and iterative linear regression methods, and have been successfully employed in a number of applications, including automatic pipe inspection (Mucciardi, 1982), fish stock classification (Prager, 1988), reconfigurable flight control (Elder and Barron, 1988), tactical weapon guidance (Barron and Abbott, 1988), and prediction of temperature distributions (Fulcher and Brown, 1991). One of the earliest polynomial network approaches, the Group Method of Data Handling (GMDH) was partly inspired by research with *Perceptrons* (Rosenblatt, 1958) and *Learning Filters* (Gabor, Wilby, and Woodcock, 1959). Introduced by Ukrainian cyberneticist and engineer A. G. Ivakhnenko in 1968 (and discussed in virtually every issue of *Soviet Automatic Control* since), GMDH has influenced the development of several techniques for synthesizing (or “self-organizing”) networks of polynomial nodes. This Section describes and differentiates those algorithms, and the next Section demonstrates the use of one of them (ASPN) on three increasingly challenging estimation tasks.

5.1. The Group Method of Data Handling (GMDH)

There is little advantage in precisely estimating the parameters of a model if its basic structure (the input variables, and their transformations and interactions) is rather tentative. Rather, the chief challenge is often the discovery of a working approximation to the underlying “machinery” responsible for generating the data one has been able to record. The GMDH (e.g., Ivakhnenko, 1968; Farlow, 1984) attempts a hierarchic solution, by trying many simple models, retaining the best, and building on them iteratively, to obtain a composition (or feed-forward network) of functions as the model.

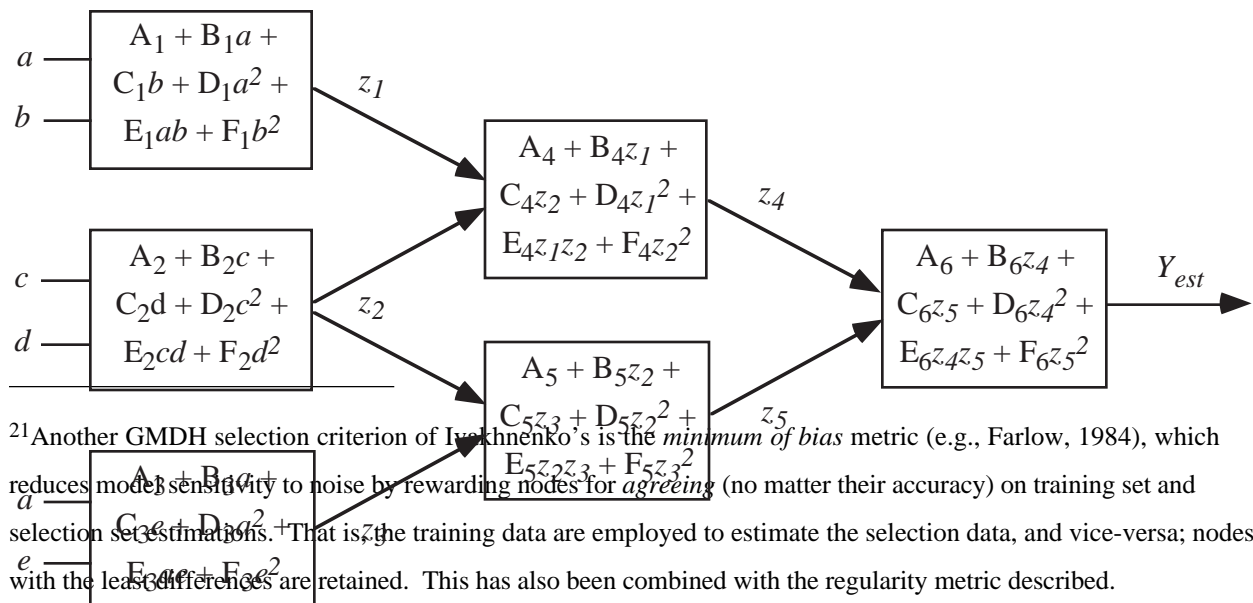
The building blocks, or polynomial nodes, usually have the quadratic form

$$z = w_0 + w_1a + w_2b + w_3a^2 + w_4b^2 + w_5ab \quad (20)$$

for inputs a and b , coefficient (or weight) vector \mathbf{w} , and node output, z . The coefficients are found by solving the Linear Regression (LR) equations with $\mathbf{z} = \mathbf{y}$, the response vector. (Thus, each

node, with only two inputs and six terms, attempts to solve the whole problem; that is, to match the response variable, on its own.) All $\binom{M_0}{2}$ pairs of M_0 inputs are examined and the best M_1 are retained as the first layer (or generation) of nodes. The process continues at the second layer, using the M_1 outputs of the first layer, z_{11} to z_{1M} , as input variables (and so on) until the model's complexity impairs its performance. The best final node, z^* , and its ancestors (those earlier nodes recursively feeding into z^*) compose the final model. An example appears in Figure 7.

The GMDH gets its name from the cross-validation-like process employed both to select nodes and to halt model construction. The data is split into two sets: one for training, one for selection. The training cases are employed with LR to find the parameters, \mathbf{w} , and the selection cases provide new data by which those models are compared. According to the *regularity* modeling criterion, nodes doing the best on the second set are retained and the rest are discarded.²¹ Likewise, when the best node of a layer (according to the selection data) is worse than the best representative of the previous layer, model synthesis halts with the latter node as the crown of the model, and all connections not feeding into it are pared away. As illustrated in Figure 8, training accuracy always improves with complexity but testing accuracy can sharply deteriorate. To avoid overfit, the GMDH halts when testing worsens.²²



²¹Another GMDH selection criterion of Ivakhnenko's is the *minimum of bias* metric (e.g., Farlow, 1984), which reduces model sensitivity to noise by rewarding nodes for *agreeing* (no matter their accuracy) on training set and selection set estimations. That is, the training data are employed to estimate the selection data, and vice-versa; nodes with the least differences are retained. This has also been combined with the regularity metric described.

²²The best stopping place is at the global minimum (occurring in Figure 8 at 4 layers), though most GMDH implementations probably halt at the first (local) minimum (at 2 layers in the example).

Figure 7. Three-Layer GMDH Network

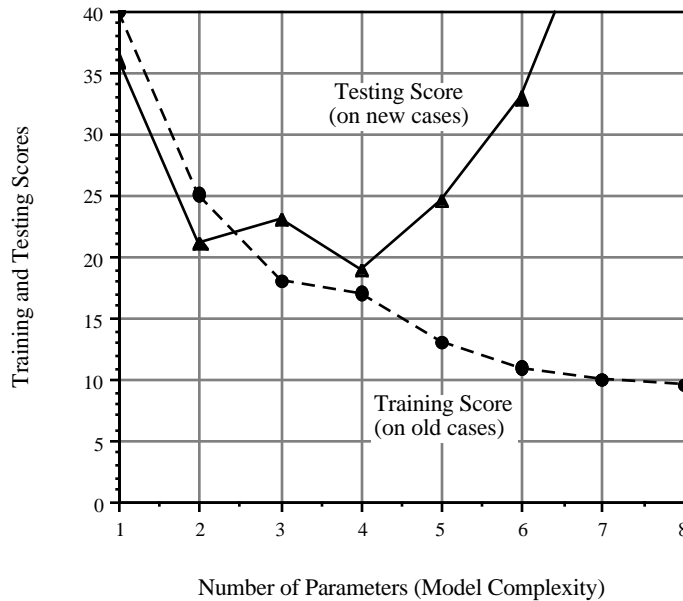


Figure 8. Model Accuracy vs. Complexity (after Elder and Finn, 1991).

Though training accuracy improves with additional terms, test performance can sharply worsen.

Through compositions of low-dimensional functions of a few variables, it is possible to build up high-dimensional functions of several variables. In fact, Lorentz (1966) has shown (extending (Kolmogorov, 1957), which disproved Hilbert’s famous 13th conjecture) that *any* multivariate function can, in theory, be approximated to an arbitrary accuracy by a (particular) compositional network of univariate functions (see Elder [this volume], Section 7). However, *inducing such a model from sample data remains a great challenge*. A sobering example is that of Artificial Neural Networks (ANNs), for which even the “data memorization” problem has been shown to be *NP-complete*: “There is no reliable method to configure a given arbitrary network to remember a given arbitrary body of data in a reasonable amount of time.” (Judd, 1990; p.7).

The expressive power of GMDH compositions appears extreme when one expands the network to obtain a single equation in the original inputs. The equation for the network of Figure 7 for example, has 981 terms (such as cd^5e and $a^2bc^2e^3$)²³. Prager (1984) expanded GMDH models and then performed extensive feature selection to remove unneeded components; still, most networks are left in compositional form, as even simple models can have thousands of terms (Elder, 1980). The actual degrees of freedom are, of course, bounded by the number of

²³Terms employ at most 4 of the 5 input variables, and have a total power ≤ 8 (2^{layers}).

parameters in the network, but two factors constrict model freedom even further. First, as demonstrated in (Elder [this volume], Section 3.3) for a simpler node form, not all apparent model flexibility is actually available. (There, similar terms in a network of 12 parameters results in a polynomial with fewer than 8 degrees of freedom.)²⁴ Secondly, early nodes become fixed during training and only parameters in the latest nodes are adjustable.

Still, the simple GMDH algorithm is quite powerful, due primarily to its creating data features, \mathbf{z} , from the original variables, and to its adjusting the complexity of the model according to performance. Also, the method grows models by a *chunk* of terms at a time, thereby avoiding some of the pitfalls of one-step greedy techniques.

Table 3. Greedy Counter-Example for Regression

Case	Y	X_1	X_2	X_3
1	1	1	1	0
2	1	1	1	0
3	1	1	0	1
4	1	0	0	1

To illustrate this last point, consider a stepwise regression solution to fitting \mathbf{Y} in Table 3 using \mathbf{X} . The best single variable is x_1 , but once it is included, no improvement is possible. Yet, $x_2 + x_3$ is an exact solution. This phenomenon is not confined to regression. Cover (1974) showed that E_1 can be the best single experiment, yet if two are allowed, two copies of the “worst” experiment E_2 can be preferred.²⁵ Likewise, the CART algorithm (described in Section 2) grows its decision tree in a greedy manner, and is thus stymied by the data of Table 4. The CART tree of Figure 9 misclassifies 2 cases, whereas the simpler tree of Figure 10 would make no errors.²⁶

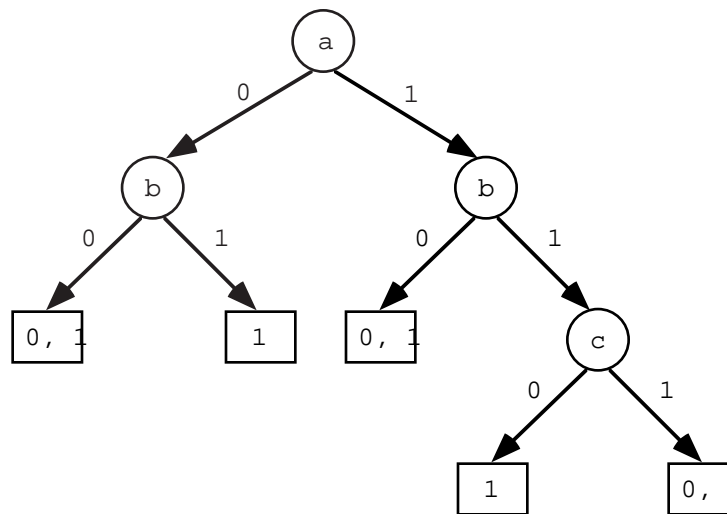
²⁴Several authors (e.g., Green, Reichelt, and Bradbury, 1988) incorrectly assume the expanded GMDH model with L layers in V variables will have the terms of the full polynomial of degree, $D = 2^L$, numbering $\binom{D+V}{V}$. However, not all interactions are made possible by pairwise connectivity. For example, the two-layer, three-node, four-input GMDH network has 54 rather than $\binom{4+4}{4} = 70$ terms; lacking, for example, a^2bd and bc^3 .

²⁵The article, “The Best Two Independent Measurements are Not the Two Best” referenced similar cases; e.g., when the two worst individual experiments (of three) were the best pair.

²⁶Under the default CART settings, no tree is constructed. This result employs resubstitution accuracy and disallows splits of small nodes (≤ 2 cases). CART requires 6 splits for exact classification of these (7 distinct) cases.

Table 4. Greedy Counter-Example for CART

Y	a	b	c
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
1	0	1	0
1	1	1	0
0	1	1	1
0	1	1	1

Figure 9. (Sub-optimal) CART Tree for Data of Table 4 ($XOR(b,c)$)

The data of Table 4 conform to the parity function $y = \mathbf{XOR}(b,c)$, but CART (and stepwise LR and other greedy techniques) begin with variable a , as it is the single best. Also, (non-greedy) *perceptrons* or other linear discriminators would fail on this task, because the parity function is not *linearly separable* (Efron, 1964; Minsky and Papert, 1969). The GMDH however, can find the correct function -- an algebraic equivalent of $\mathbf{XOR}(b,c)$:

$$y = b + c - 2bc \quad (21)$$

-- as it employs nonlinear terms and is less greedy in its search (simultaneously setting six terms, rather than one).

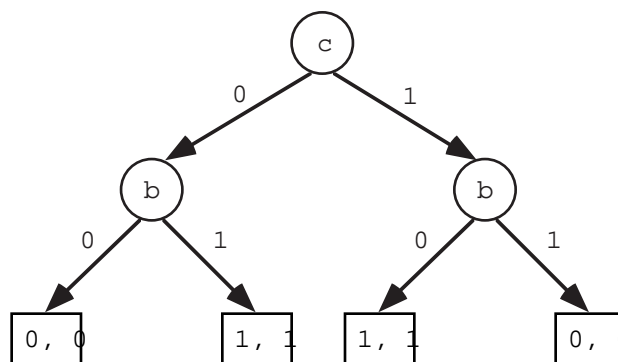


Figure 10. Binary Tree Implementing $XOR(b,c)$

5.2. The Polynomial Network Training Routine (PNETTR)

In 1968, R. Barron of Adaptronics, Inc. (ADI) met Ivakhnenko while attending conferences in the USSR, and invited him to submit a paper to the newly formed *Systems, Man, and Cybernetics* Journal. According to (Barron, Mucciardi, Cook, Craig, and Barron, 1984) that article, “Polynomial Theory of Complex Systems” (Ivakhnenko, 1971) “greatly stimulated interest in GMDH outside the USSR”. A. Mucciardi (also at ADI, and the IEEE Editor who annotated Ivakhnenko's paper for clarity) then programmed a Polynomial Network Training Routine (PNETTR), enhanced with *clustering* procedures stemming from his doctoral research (Mucciardi and Gose, 1972). Other additions included the use of a third data subset to evaluate model quality, reintroduction of some (preassigned) original inputs as candidate variables in subsequent layers, global adjustment of the final network weights, and cascading of networks (Barron et al., 1984).

The cluster information was found to be particularly useful, and separate networks were sometimes trained for individual data clusters. Also, considering original input variables for use in later layers was an important innovation (discussed further below). The cascading of networks however, (where the errors of one network become the training goal, \mathbf{y} , for the next), undoubtedly led to overfit, and perhaps was a motivation for the additional fragmenting of the data into a third set. In any case, the heavy use in GMDH of the selection data set does impair its “independence” and thus its ability to determine model accuracy, so a third set is useful -- *if* the reduction in training data can be afforded.

Global parameter adjustment after the network structure is established allows weights in early layers (which are fixed under GMDH) to be changed -- potentially improving the network. For nearly a decade, ADI researchers had been relying entirely on such a search procedure to establish weights for their multi-layer (polynomial) neural network models (called Adaptive Learning Networks, or ALNs), with some success (see Elder [this volume], Section 3.2; Barron et. al,

1984; Hecht-Nielsen, 1990). However, switching from networks of fixed structure to the self-connecting networks formed by GMDH seemed to be a breakthrough, and the PNETTR algorithm was successfully employed on commercial and military applications for a number of years (e.g., Barron, 1975).

Still, the heavy data demands of the algorithm (each of the three data sets were to contain representatives of every cluster of data points), and its tendency to overfit (especially with cascading) highlighted the need to better employ the available data. At this juncture (1981), A. Barron (son of R. Barron) derived a complexity penalty metric²⁷ which allows use of all the data, by penalizing models (in units of accuracy) for each parameter they employ.

A. Barron completely overhauled PNETTR, eventually introducing three-input cubic nodes and nonlinear (cube-root and exponential) nodes, allowing candidate inputs to come from *all* preceding layers, producing density estimates and classification tables, and automatically creating Fortran routines implementing the models. Also, as networks increasingly consisted of fewer layers of larger (up to 13 term) nodes, pre-determined subsets of the largest were considered (i.e., some terms were removed *a priori*) to allow simpler networks, when possible. To enhance understanding of network weights, stimulus variables were mean-sigma normalized (and response variables "unitized"); also, to ease the growing combinatorial explosion, input pairs were screened before combination. The early PNETTR additions of clustering, cascading, and global adjustment were dropped, but the speed and usability of the code was enhanced (Barron et al., 1984). PNETTR (Version IV) was extensively employed in-house and occasionally leased to customers. Some of its capabilities are even in hardware devices; in particular, robots for nondestructive evaluation of materials (from the Adaptronics Products division of Flow General, Inc., of McLean, Virginia -- the company which purchased ADI in the early 1980s).

²⁷The Predicted Squared Error (PSE) metric is similar to those of Akaike (1972) and Mallows (1973); see Sections 4.4 and 4.6). A. Barron later (1985) extended Rissanen's MDL model selection criterion (Section 4.5). Prager (1984) also employed a complexity penalty (as well as subset selection, to reduce terms of an expanded model).

5.3. The Algorithm for Synthesis of Polynomial Networks (ASPEN)

Inspired by descriptions of, and some experience with, PNETTR, J. Elder (1985, 1989) wrote the Algorithm for Synthesis of Polynomial Networks (ASPEN).²⁸ The primary building blocks (nodes) of ASPEN are incomplete third-order polynomials with one to three inputs

$$z_{single} = w_0 + w_1a + w_2a^2 + w_3a^3 \quad (22)$$

$$z_{double} = w_0 + w_1a + w_2b + w_3a^2 + w_4b^2 + w_5ab + w_6a^3 + w_7b^3 \quad (23)$$

$$\begin{aligned} z_{triple} = w_0 + w_1a + w_2b + w_3c + w_4a^2 + w_5b^2 + w_6c^2 + w_7ab + w_8ac + w_9bc \\ + w_{10}abc + w_{11}a^3 + w_{12}b^3 + w_{13}c^3 \end{aligned} \quad (24)$$

In addition, nonlinearly-weighted nodes of cube-roots ($z = w_0 + w_1\sqrt[3]{a - w_2}$) and exponentials ($z = w_0 + w_1e^{w_2a}$) are considered²⁹, along with a multi-linear node ($z = \sum_{k=1}^M w_k x_k$), for each layer.

This last element indefinitely extends the number of possible inputs that can be considered at one time, albeit with a very simple structure (and ensures that the networks outperform ordinary LR!). Interestingly, the one term in the ASPEN triple not in the largest PNETTR nodes ($w_{10}abc$) turns out to be important; enabling, for instance, solution of the 3-input parity problem³⁰. (In general, the M -input parity problem can be solved by a single, multi-product term, $w \prod_{k=1}^M x_k$, if the binary

inputs are encoded to the values ± 1 , rather than 0 and 1 (Elder [this volume], Table 7). Otherwise, the multilinear polynomial in M inputs, with 2^M terms, is required.)

Other ASPEN innovations (demonstrated in the next Section) include potential *sharing* of nodes between different networks, *carving* of building blocks (greedy reverse term reduction)³¹,

²⁸Some of the contributing ideas stemmed from a 1985 workshop, hosted by Barron Associates, Inc. (BAI) which included researchers R. Barron, J. Elder, and P. Hess from BAI, A. Desrochers from Rensselaer Polytechnic, A. Barron and S. Boyd from Stanford, M. Prager from U. Rhode Island, and P. Chandler and the (late) E. Rachovitsky, from Wright Aeronautical Labs, the Air Force agency which sponsored the early development of the algorithm.

²⁹PNETTR apparently calculated these nonlinear transformations “between layers” for a subset of the outputs of the previous layers, and it is not clear whether they impacted the perceived complexity of the network. In ASPEN, they were treated similarly to all other nodes (but seemed to be relied on more rarely).

³⁰XOR(a,b,c) is 1 when an odd number of inputs are 1, and is equivalent to $y = a + b + c - 2ab - 2ac - 2bc + 4abc$.

³¹An improvement would employ optimal branch and bound subset selection (e.g., Elder [this volume], Section 4).

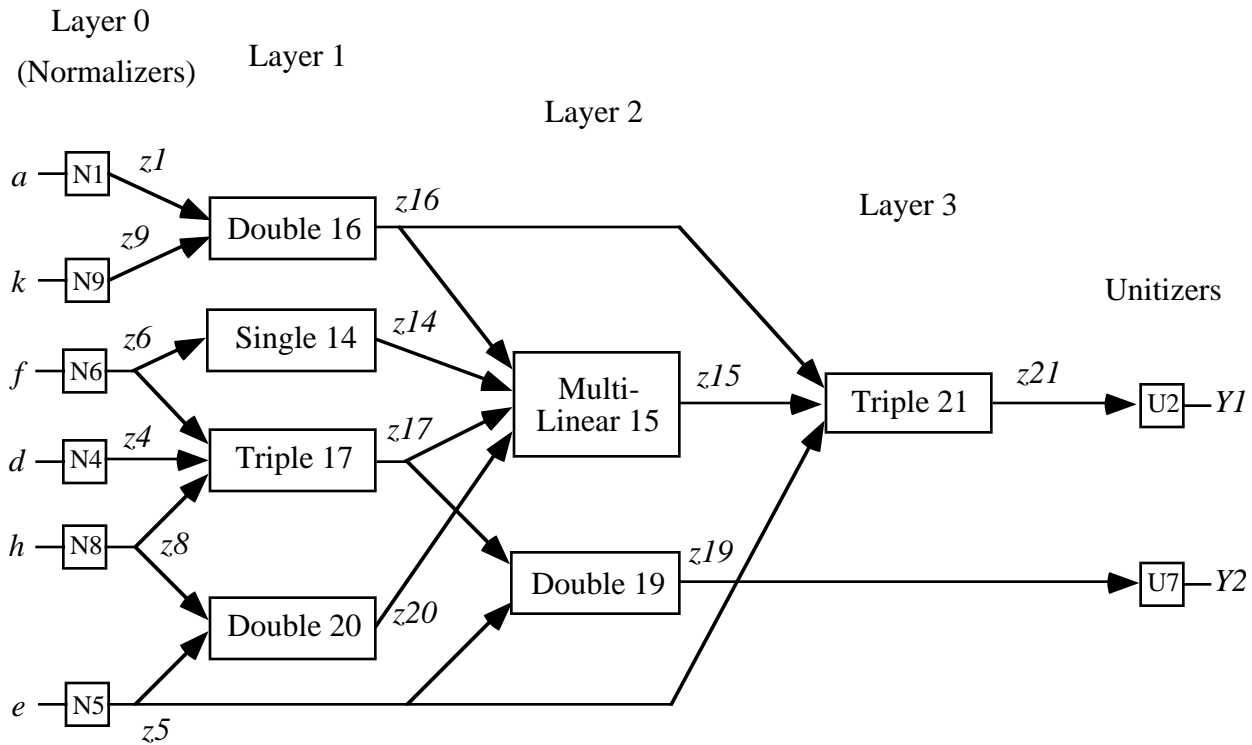


Figure 11. Sample ASPN Network.

The network simultaneously estimates the two response variables, Y_1 and Y_2 . The Y_1 model is three layers deep and employs six polynomial nodes and six original stimulus variables; that for Y_2 has two layers, three nodes, and four input variables. Note that there is cross-coupling, or sharing of intermediate results (Triple₁₇), between the sub-networks. The stimulus variables (automatically selected from the set of candidates) are each normalized before use, and the estimates for the response variables are similarly *unitized* upon output.

Nearest-Neighbor initialization of the complexity parameter, use of the PSE or MDL modeling criterion, several measures of variable influence, real-time network graphics, efficient re-use of intermediate data statistics, and a great deal of flexibility in defining limits and secondary options. An example network for two response variables is shown in Figure 11.

ASPEN is accompanied by five utility programs:³²

1. *Imp*: Evaluate the model on new data and/or output the model as C or Fortran code.
2. *Style*: Design network nodes.

³²Other direct contributors are S. Goldschmidt (Style, Imp, and DBPrep), D. Barron (Xpress), K. Cascone (some DBPrep refinements and documentation), and P. Hess and R. Cellucci (utility routines for GlobalO).

3. *DBPrep*: Transform, graph, and extract features from data.³³
4. *GlobalO*: Globally optimize the network according to a user-defined accuracy metric.³⁴
5. *Xpress*: Algebraically expand the network into a single equation (after Elder, 1980) and/or derive partial derivatives for the output with respect to the inputs.

ASPN has been extensively employed in aerospace applications (e.g., R. Barron, Cellucci, Jordan, Beam, Hess, and A. Barron, 1990), and is available from Barron Associates, Inc. of Stanardsville, Virginia. A scaled-down version of ASPN and Imp for personal computers (known as *AIM*; e.g., Cellucci and Hess, 1990; Montgomery and Drake, 1990) is available at a reasonable cost from AbTech, Inc., of Charlottesville, Virginia. Though lacking many useful ASPN options, the key GMDH-like ideas are implemented within a user-friendly object- and menu-driven interface, as shown (for the MacIntosh) in Figure 12. Polynomial networks can take orders of magnitude less time to train than conventional back-propagation ANNs, and typically achieve better results (e.g., Tenorio and Lee, 1989). A recent *Spectrum* reviewer, performing such a comparison, dubbed AIM “far and away the most practical machine learning tool on the market today” (Shewhart, 1992).

5.4. The Analogy with Evolution

Despite its name, the essence of the Group Method of Data Handling lies not in grouping the data into training and testing sets. Nor are polynomial nodes essential. The key innovation is adapting a compositional network to sample data; that is, inducing a general model structure, as well as its parameters, from example cases in a stagewise manner. In its creation and development, GMDH researchers have often been consciously guided by an evolutionary paradigm:

...starting with a few basic primeval forms or equations, we construct a new generation of more complex offsprings and then allow a survival-of-the-fittest principle to determine which new equations live and which equations die. Continuing this process for many generations, we will “grow” a progeny of mathematical models that will behave more and more like the real system. (Farlow, 1984; pg. 1)

³³DBPrep can display scatterplots, strip charts, histograms, and Fourier spectrums. Available transformations include power, polynomial, rational, trigonometric, piecewise linear, and time-series functions, as well as Linear Polynomial Filters (Gilstrap, Goldschmidt, and Elder, 1986).

³⁴A re-introduction of the early ADI network training method, but allowing arbitrary “score functions”, not just least squares (L_2). Still, network structures found by minimizing L_2 seem to at most be fine-tuned by a later optimization process, and not radically affected, even if the criterion is as different as L_0 (hit/miss metric). -- JFE

File Edit Operations Parameters

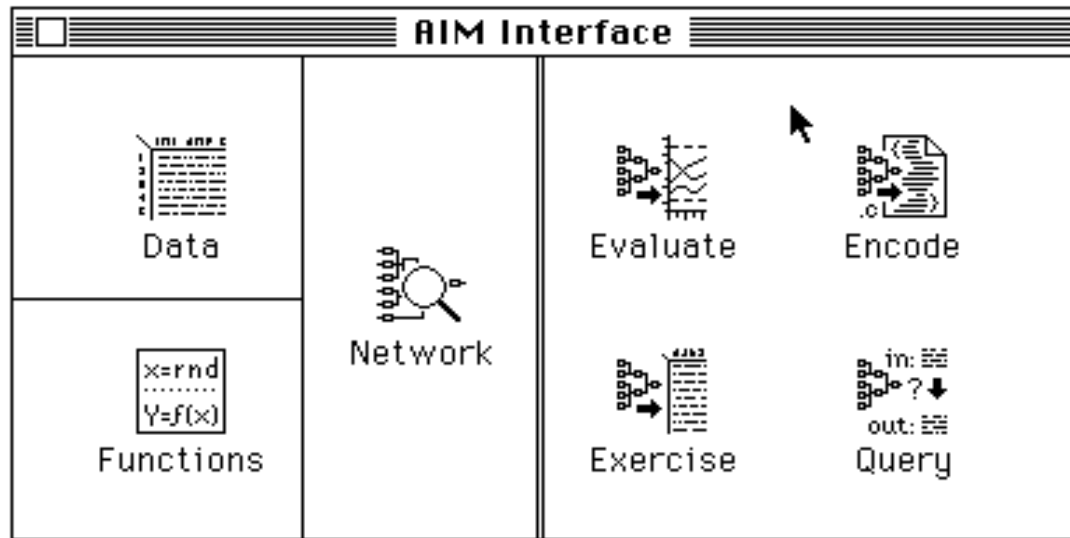


Figure 12. AIM Interface for MacIntosh™.

The leftmost icons allow one to view and edit the data base, generate test functions, and synthesize a network. Once created, a network can be inspected node by node (not shown), encoded into C, evaluated statistically, exercised over a data base, or queried one case at a time by the icons on the right. The pull-down menus, at the top, contain further options.

Modern faith in the power of evolution seems to have been a factor in the development of many algorithms -- e.g., ANN, GMDH, Genetic Search (e.g., Holland, 1975; Denning, 1992), and Evolutionary Programming (Fogel, Owens, and Walsh, 1966) -- which consciously draw on that analogy. However, the supposed wonder-working ability of the paradigm is often appropriated for the derivative programs, regardless of their actual performance. Ivakhnenko for example, expects much from the GMDH methodology:

It is remarkable that there is no end in sight. New horizons open continuously, turning the GMDH algorithm into an apparatus unifying many engineering sciences. The GMDH algorithm has become the center of cybernetics and of a universal metascience for studying the modeling of communication and control of complex systems.[but] There are no sessions at world or national congresses or conferences devoted to GMDH. Publications in major journals on control or communication theory in the United States and the USSR are not devoted to GMDH papers. This situation must be changed for the benefit of humankind. (Ivakhnenko, 1984; pp. 108, 116).

However, the algorithm is less analogous with *macroscopic* evolution (to which so much is ascribed) than to *microscopic* evolution -- particularly, guided adaptation (with its more humble improvements). That is, GMDH modeling is more like breeding animals for a specific purpose (e.g., dogs to pull sleds in the Arctic) than it is “creating new species of creatures”. Firstly, the offspring are not random mutations, but are *directed* toward a specific goal (the response variable, Y). Secondly, the end results are (perhaps surprisingly) strongly dependent on the quality of the starting variables, as demonstrated in Section 6.3. “Within-species refinement”³⁵ is thus the evolutionary analogy appropriate to the GMDH family of algorithms, and which can help illuminate its strengths and weaknesses.

5.5. Potential Improvements

Inductive polynomial networks have proven useful for many challenging statistical problems.

[GMDH] appears to be advantageous with systems characterized by complexity with many variables and parameters, ill-defined mathematical structures, and limited data. In other words, this algorithm is useful for empirically generating hypotheses about systems of which relatively little is known. (Duffy and Franklin, 1975; p. 226)

However, Green, Reichelt, and Bradbury (1988) tested a straightforward implementation of GMDH on variations of several (known) test functions, and found it to fail to perform as well as had been claimed, *especially* in the aforementioned cases:

... GMDH is inferior to regression for processes that are noisy, nonlinear, or multivariate, and for small sets of source data. A major problem is that extrapolations can diverge rapidly from true values of a process unless care is taken. (p. 49).

Yet, they also noted that introducing lagged values of the response variable as candidate inputs to GMDH “dramatically improved the model fit [particularly] outside of the training range for each of the cases examined” (p. 55).

³⁵It was observation of such a physical process that led Darwin (1859) to extrapolate to the revolutionary idea of cross-species evolution, and to initially declare “I can see no limit to the amount of change to organic beings which may have been affected in the long course of time through nature’s power of selection.” However, despite the theory’s nearly universal modern following (as the sole naturalistic contender for our origins), it remains weakly supported: “The influence of evolutionary theory on fields far removed from biology is one of the most spectacular examples in history of how a highly speculative idea for which there is no really hard scientific evidence can come to fashion the thinking of a whole society and dominate the outlook of an age.” (Denton, 1985; p. 358) - JFE

Obviously, the quality and information content of the input variables matter a great deal (and Examples in Section 6 confirm that even when the data contain the necessary “raw materials”, it can pay to refine the features). Analyst experience too, can significantly impact the quality of models constructed with this exploratory technique. Still, algorithmic limitations exist. Green, Reichelt, and Bradbury (1988) suggest the main GMDH weaknesses stem from “combining variables in pairs, rather than larger groupings, when forming submodels” (p. 68). Earlier such observations led to extending node “width and depth” with three-input cubic building blocks in PNETTR (III & IV), and to further introducing (variable-sized) multilinear nodes in ASPN. Still, the “reach” of the method is limited when contributions from variables must truly be simultaneous, as with the parity problem. (Only ASPN can handle three variables, and none will automatically identify a parity function of four or more.)

A more pressing limitation may stem from what is most often a strength of the method: its *directedness*. Pointing every node toward the final goal often allows simple models to be found, and their compliance provides a natural measure (along with complexity) by which nodes may be compared. Use of a goal vector (and the L_2 metric) allows the weights to be determined very efficiently by LR; however, the various intermediate solutions become more and more alike as the network grows. For this reason, and because they come with no complexity “price tag”, original input variables are very often employed in every layer by PNETTR and ASPN models. The networks are then not pyramidal (as for GMDH in Figure 7), but more serial -- new nodes successively employing original inputs to refine a leading candidate model from the previous layer.

This seeming propensity for input variety also reveals why ASPN often shares nodes between outputs trained simultaneously. Even though nodes trained for another response variable are costly (in terms of complexity penalty) their approximation to that related output can contain contextual information useful to a “peer” model. (An example appears in Section 6.3.) Other ways to encourage population diversity (while avoiding the morass of sheer randomness) include

- 1) *Multiple aim points*: A suite of nonlinear, monotonic transformations of the response variable (e.g., $\ln(Y)$, Y^{-1} , $\Phi(Y)$) could provide alternate goals from which the original output can be recovered. (This is similar to node sharing, but is also possible when only one variable is being modeled.)
- 2) *Multiple criteria*: Retain a few nodes for qualities other than accuracy, as with the “minimum of bias” GMDH criterion which rewards robustness, or through a correlation penalty (e.g., the classification metrics of Mucciardi and Gose, 1971) which discourages use of redundant variables. In the limit, some nodes could be

trained to combine well with their peers, by focusing on residuals, $Y - Z_k$, rather than the response vector, Y (as performed by several iterative methods, e.g. Generalized Additive Models (Hastie and Tibshirani, 1986)).

- 3) *Multiple regions*: Rather than approximating the response for all cases (using the data projections available to each node) cluster the cases, prepare a model for each region, and combine the sub-models (similar to Regression Trees (Chaudhuri, Huang, Loh, and Yao, 1990), and Composite Models (Skeppstedt, Ljung, and Millnert, 1992)).
- 4) *Bi-directional growth*: Increasingly complex features of the stimulus variables “reach forward” to match the response; but, the response could also “reach backward” to meet the original variables or their transformations -- generalizing multiple aim points. If those reverse functions are invertible (as above), a complete mapping from stimuli to response is possible; if not, perhaps some useful data transformation will nevertheless be suggested (as sought by the Alternating Conditional Expectation, ACE, technique for function smoothing (Breiman and Friedman, 1985)).

Incorporation of any of these suggestions should enhance the quality of polynomial networks synthesized. Still, the current inductive algorithms have the flexibility and power to address a wide range of challenging applications. The next Section demonstrates use of ASPN for two estimation tasks and a control problem, and highlights several algorithm capabilities (and modeling dangers) mentioned.

6. EXAMPLE ASPN SESSIONS

This Section demonstrates detailed use of ASPN and its utilities to solve a sequence of increasingly complex modeling tasks in forecasting, design, and control. In this discussion, ways to identify and employ intermediate results to guide research is emphasized. The ASPN User Manual (4th edition; Elder, 1989) contains further details on the program options and particular steps taken in these examples.³⁶

6.1. Task 1 -- Moon Ballistics

Suppose a fanciful problem: A cannon on the moon has been test-fired 50 times at various random angles of inclination, γ (degrees), and muzzle velocities, V (meters/second), and the corresponding distance that the cannonball travelled, X (meters), has been recorded (Table 5). From these data, we want to induce the general relationship that will predict X for new cases, when given V and γ . In other words, we want to run ASPN on a training subset of Table 5, where X is the response (output) variable (identified with the letter "Y"), V and γ are candidate stimulus (input) variables ("X"), and the observation number is a validation variable ("V") (used to flag missing cases or data items). Such commands to ASPN are transmitted through a *control file*, which list an option, its value, and an optional comment. (Lines beginning with a "!" character are also comments.) The file could be as brief as:

```
! (minimum) ASPN control file for Task 1:
InFile   : moon0.dat           Training data file
Vstring  : V, Y, XX           Variable identification
Inputs   :      2             (& 1 output assumed)
Validate:                     Obs. number in V column
```

which instructs ASPN to read the entire file moon0.dat for training observations, where each case (row) contains four variables (columns), the last two of which are candidate inputs. Further, the number contained in the first column will be checked as each observation is read to help insure that the data are correctly interpreted. (The rest of the ASPN arguments take on default values.)

Case	X	V	γ
1	28462	411.6	58.4
2	31737	475.9	24.0
3	40767	467.2	41.3
4	28113	482.9	19.9
5	30943	408.5	50.1
6	31408	461.7	25.7

³⁶Permission by Barron Associates, Inc. of Stanardsville, Virginia, to extract most of this Section from Appendix B of the ASPN User Manual (4th Edition; Elder, 1989), is gratefully acknowledged.

7	18628	424.7	16.6
8	33181	483.6	24.4
9	20527	408.0	20.4
10	30533	419.5	56.5
11	38478	484.2	59.7
12	35721	435.6	44.1
13	43482	480.9	43.3
14	38491	453.0	42.3
15	23102	474.2	16.5
16	24575	402.0	26.9
17	37081	488.0	27.9
18	29039	426.1	29.1
19	42385	489.6	55.1
20	27092	403.8	59.1
21	29572	402.5	37.9
22	28648	405.7	33.8
23	21803	405.1	22.4
24	42866	499.2	33.0
25	45538	498.6	38.3
26	40293	473.2	53.6
27	36201	473.0	29.6
28	45302	496.7	51.5
29	38053	459.0	36.8
30	46442	497.9	42.0
31	29641	401.7	38.6
32	20151	414.4	19.3
33	20712	438.4	17.4
34	29680	404.8	52.9
35	34542	433.5	51.3
36	35660	438.5	39.9
37	38986	466.8	35.9
38	40452	471.7	52.6
39	28603	403.3	34.5
40	33510	487.8	24.2
41	33442	421.5	43.9
42	41984	475.1	40.5
43	38479	452.3	46.6
44	37194	453.0	37.1
45	33875	481.8	25.4
46	44264	489.1	39.6
47	39591	460.5	48.8
48	26028	449.5	21.6
49	33755	498.6	66.9
50	45438	491.1	44.8

A more thorough control file appears in Table 6,³⁷ which differs from the simple one above in initializing σ_p (the complexity penalty factor) through the nearest-neighbor method, employing at most two layers of nodes, allowing more carving, re-solving elements after carving, allowing cube-root and exponential nodes, not combining input variables which are very highly correlated (positively or negatively), and naming the variables.³⁸

³⁷Commented out parameters (starting with “!”) take on default values, but are listed to suggest reasonable settings.

³⁸If no title line heads the data file, the k^{th} variable name can be passed with “Name k <vname>”.

Table 6. Complete ASPN Control File (Task 1)

InFile : moon0.dat		Training data file
!UnForm		(input file is formatted instead)
NumObs : 40		(max) #observations (cases) to input
Title		(heading file: "obs, X, V, gamma")
Vstring : V, Y, XX		Variable identification
Outputs :	1	(confirming Vstring)
Inputs :	2	(confirming Vstring)
Validate		(confirming Vstring)
Neighbor		Initialize SigmaP with NN method
!UseMDL		(would use MDL instead of PSE)
!SigmaP : 100.		(init σ_p with Nearest Neighbor instead)
!UseUP		(would enable Uncertainty Penalties)
!UvalueM : 0.02		(would be 2% V uncertainty)
!UvalueP : 0.50		(would be 1/2 deg. gamma unc.)
Pix		Draw figures of elements
!Verbose		(would show all details)
LayerLim :	2	Stop at or before two layers
Layersize:	10	Save best 10 on first layer
Layersize:	2	Save best 2 on last layer
CubeRoot		Try cube-root elements
Expon		Try exponential elements
!Wfirst		(would restrict 1st layer to multi-linear element)
Carvelim :	0.85	Carve up to 85% of terms
ReSolve		Re-solve coeffs after carving
CCthresh :	0.98	Correlation limit for pairs
!Borrow :	4	(no need to share; only 1 output)
!Offactor : 1.25		(use default of 1.0 instead)
!Exhaust		(would try all triples)
!Tdepth : 2		(would take triple inputs from X and 2 most recent layers; instead, allow from all previous layers)

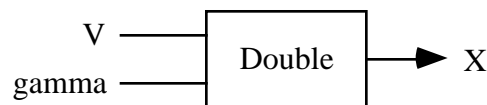
In addition to input data, and a control file, ASPN employs an element file defining the polynomial "building blocks" for model synthesis. The default nodes, with from 1 to 3 inputs (see (22) to (24) in the previous Section) may be altered by the Style utility.

To judge the model after synthesis, it's a good idea to reserve some original data (not showing it to ASPN) to use as an independent test once the model has been created. To set aside 20% of the cases here, we could instruct ASPN not to read past the 40th case (NumObs=40), or we could employ DBPrep to split the data file by randomly removing 10 cases. This latter option is best if the data are listed systematically; i.e., if it's roughly sorted by one of the variables, since random exclusion makes it more likely that all clusters of the original data space will be represented in the training data. Such representation is extremely important if the model is meant to be valid for each region, as it is not likely to do well in a novel region of data space (i.e., extrapolating). In this example, the data appear to be more random than systematic, so we can take the shortcut of lowering NumObs to reduce the training set. To check for "data space coverage" though, it is often useful to scatter-plot several variable pairs or construct their histograms to discover regions with either too much or too little data.

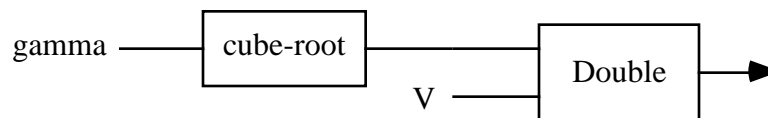
ASPN runs rather fast on this small problem³⁹ and, after a few seconds (on even a personal computer), recursively outputs the model description file, moon0.mod, of Table 7, and an APN file encoding the model, and draws the final screen of Figure 13.

Synthesis in progress. Best model so far:

Score = 0.931e-02



Current element in layer 2:



--> Network for X completed <--

--> Halt due to lack of network growth on layer 2 <--

³⁹An output summary is recorded in the log file, but to provide results during a run, stop text scrolling with the keys CTRL-S and restart scrolling with CTRL-Q. The current and leading candidate networks are graphed continuously during model synthesis.

Figure 13. ASPN Screen at End of Task 1.

Table 7. Model Description File “moon0.mod” (Task 1)

Unitizer for X has coefficients: 33010.0 7539.11 and used element # 7 as input.

Element 7 (style: Double) Input(s): 3 4

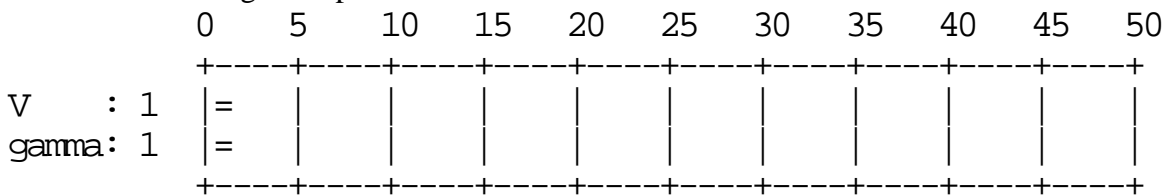
Model performance:

FSEp:	0.00354732	FSE:	201623.	38% FSE
KPp :	0.00576008	KP :	327393.	62% KP
UPp :	0.00000000	UP :	0.	0% UP
PSEp:	0.00930940	PSE:	529130.	
		rPSE:	727.413	

Coefficient Values:

0.496111 0.570040 0.590458 0.0 -0.490606 0.0682152 0.0718913 0.0

Path counts for original inputs:



Each "=" represents one point.

Normalizing coefficients:

Element 3 (style: normalizer) V

Coefficient Value(s): -12.9665 0.0288902

Element 4 (style: normalizer) gamma

Coefficient Value(s): -2.81550 0.0757467

The Slope of the Model at the Data Mean:

Output	Input	Slope	Normalized Units
X	V	124.154	0.570022
X	gamma	337.191	0.590463

The Average Sensitivity of the Model Over the Modeling Database:

Output	Input	Sensitivity	Normalized Units
X	V	171.131	0.785703
X	gamma	561.290	0.982887

How good is the model? The first clue is the *score*, printed to the screen. The simplest (least accurate) model, the constant m_y , has a score of 1.00; whereas, a "wire" element (a normalizer connected directly to a unitizer, and considered to have no complexity) exactly fitting the data would have the best score: 0.00. The score is the normalized Predicted Squared Error (denoted PSEp), so its square root is an estimate of the *proportion of the standard deviation* of the problem remaining unexplained.

To see what that means in the units of the data, examine the root-Predicted Squared Error, rPSE, or predicted error standard deviation.⁴⁰ Here, $rPSE = 727.4$, which, for a Gaussian error distribution, suggests that two-thirds of the model errors, when used on new data, will be off by less than 727 m. This is an order of magnitude smaller than the unmodeled standard deviation of 7539 m. (σ_y appearing as the second unitizer coefficient), but is not acceptable if our "created moon craters" need to be predicted more accurately. To compare further, the prior estimate of this standard error, σ_p , was set to just over 1 km. due to the Nearest Neighbor (NN) option ; that is, $\sigma_p = \text{half the nearest neighbor error } \sigma$ (as explained in Section 4.6):

(Unitized) Nearest neighbor error statistics:

Y AveErr	AveAbsErr	SigmaE
1 -98.40	1557.15	2089.31

(printed to the screen and log file). The predicted error of the model, rPSE, being less than the prior estimate, σ_p , is a good sign; if it were greater, we would have been proven overly optimistic. Note that the proportion of the total PSE penalty ascribed to each component is listed with each node in the model file. In this example, a majority (62%) of the PSE is due to complexity; again, a good sign: if all our predicted error were due to inaccuracy, we would have less room in which to work. To look at accuracy alone, take the square root of the normalized Fitting Squared Error, FSEp. Here, $\sqrt{.003547} = 0.06$, so only 6% of the training data deviation remains to be fit.

Before leaving Table 7, let's note other items for future reference. Two of the eight terms (of the node type, "double") are zeroed-out, or *carved*; their position reveals them to be V^2 and γ^3 . The histogram depicts the number of distinct element paths by which each original input affects the output of the network -- a robust measure of influence. (These graphs are more interesting in the log file, where they count the paths for *all* the several elements saved on the layer, not just for the best model.) The last table, Average Sensitivity, can be useful, as it describes the average degree to which small changes in each of the inputs propagate through the network to affect the output.

⁴⁰The rPSE for each element is at the bottom right of the "model performance" section of the moon0.mod file, Table 7. The value for the network is that for the last element -- the one feeding into the unitizer at the top.

The sensitivity value of 171 for $\delta X/\delta V$ means a change of 1 m./sec. in initial velocity will affect the range estimate by 171 m., on average (the actual amount depends on the context; e.g., the value of γ). Lastly, the normalized expression of sensitivity quantifies the relative importance of the inputs; values of 0.786 and 0.982 suggest both inputs are important but that γ is slightly more influential.

Another output file, the Adaptive Polynomial Network, or APN, file, encodes the model for the ASPN utilities to access. The elements employed are described to allow the APN to "take its vocabulary with it". Also, statistics of each employed variable (measured extremes, μ , σ) are included to verify that the model is appropriate for new data. Lastly, the network modules, interconnections, weights, and training performance are recorded.

To view the equation in a more standard format than a network, run Xpress. A drawing of the model appears first, then a menu of options. Displaying the model (to a chosen number of digits), and asking for the derivative of the estimate with respect to each of the inputs, results in

$$X \approx -1.224E7 + 7980V + 1410\gamma - 17.60V^2 + 1.125V\gamma - 21.22\gamma^2 + 1.307E-2V^3 \quad (25)$$

$$\delta X/\delta V \approx 7980 + 1410\gamma - 35.18V + 1.125\gamma + 3.922E-2V^2 \quad (26)$$

$$\delta X/\delta \gamma \approx 1410 + 1.125V - 42.43\gamma \quad (27)$$

These equations are simple enough to be used instead of the APN (encoded through the IMP program), but generally, the APN is more accurate (in coefficient precision), can be more illustrative (normalized coefficients and the network form often reveal relationships otherwise hidden in units), and much more reliable and easy to use (IMP can automatically encode the model in FORTRAN or C, removing the need to type in all the coefficients).⁴¹ (If a single equation is necessary, employ various Xpress options to filter out insignificant terms. For example, a model with 3093 expanded terms was reduced to only 39 when coefficients smaller than 10^{-14} in magnitude were culled (Elder, 1989; p. 55).)

IMP writes network model routines with comments, optional debug statements, and optional input and output limits; also, the variable names originally provided in the ASPN control or data file are employed (which is helpful a week later!). Under the "create" option, IMP wrote the C routine of Table 8⁴² to be embedded in a simulation program for real-time use.

⁴¹Ada and other languages may be available now, as well.

⁴²"White space" in the file has been edited slightly to improve its appearance.

Table 8. C Code Generated by IMP for a Network Model (Task 3)
(edited slightly for appearance)

```

apn()
{
    extern double exp();
#define pow1(x) (x)
#define pow2(x) ((x)*(x))
#define pow3(x) ((x)*(x)*(x))

    extern double V, gamma; /* link to input variables */
    extern double X; /* link to output variables */

    /* Performance for output Id 2 (X)
    FSE'= 0.0035473 FSE = 201623.
    KP' = 0.0057601 KP = 327392.
    UP' = 0.0000000 UP = 0.
    PSE'= 0.0093094 PSE = 529130.
    rPSE = 727.413 */

    double Z[101]; /* Array of 101 network elements */

#ifdef DEBUG
    printf("APN: V is %g\n", V);
    printf("APN: gamma is %g\n", gamma);
#endif

    /* Layer #0: Normalize original inputs */
    Z[3] = -12.9665 + 0.0288902*V;
    Z[4] = -2.81550 + 0.0757467*gamma;

    /* Layer # 1: */
    /* Element # 7: Double */
    Z[7] = 0.496111 + 0.570040*pow1(Z[3]) + 0.590458*pow1(Z[4])
        - 0.490606*pow2(Z[4]) + 0.0682152*pow1(Z[3])*pow1(Z[4])
        + 0.0718913*pow3(Z[3]);

    /* Final layer: Outputs */
    Z[2] = 33010.0 + 7539.11 * Z[7]; /* Unitize X */

#ifdef DEBUG
    printf("APN: X (was %g) reset to %g\n", X , Z[2]);
#endif
    X = Z[2];
}

```

Returning to evaluating the quality of the model, the training information from ASPN was that roughly 90% of the problem had been solved ($\sigma_p/\sigma_y \approx 0.10$), and further clues suggested that this estimate may be a conservative ($rPSE < \sigma_p$, and much of that was due to complexity), so there is reason to expect that training performance can be improved. Still, caution is advisable. A common rule of thumb (for moderately-sized databases) is to have 10 independent cases for every model coefficient; here, we have a smaller ratio, 40:6, or one third too few. Testing the model on the reserved data may reveal evidence of over- (or under-) fit.

The program IMP ("output" option) evaluates models on sets of data. With no input or output limiters, we obtain the overall statistics file (moon0.sts) of Table 9 (and, optionally, a line-by-line results file). The most striking thing about the first is the presence of an *outlier*, or unusual data point. This is signalled in two ways (also written to the screen): the warning at the top that an evaluation point was outside the (rectangular) training region, and the "lonely" error point on the top of the (sideways) histogram. The last section, listing each extreme, shows what the problem is: case 49 has a γ value (variable 4) of 66.9° ; whereas, the maximum γ encountered during training was 59.7° ⁴³. This difference resulted in an error nearly four times larger in magnitude than the next worst case.

The single outlier has a (perhaps surprisingly) large effect on the error statistics such as root of average squared error, the evaluation equivalent to rMSE. This is due to using squared (L_2) error rather than the more robust absolute (L_1) error metric. (For comparison, both are calculated by IMP.) The rMSE for the evaluation data, 700.4 (m), is below the predicted value (rPSE) of 727.4, but should be closer to the fitting value of 449.0 if our model were as good as we supposed. This is true especially since four fifths of the evaluation data *was* the training data! (All the cases were employed for evaluation.) Thus, in a weighted sense, the evaluation cases are about three times worse than the training cases.

The case-by-case error file (not shown) reveals how most of this deviation is due to the outlier, case 49; none of the other new cases, 41 through 50, are off by even a fourth as much. Thus, if we can do without a model good for very large γ values, we could run the evaluation again on just the nine reasonable new points to get a more accurate estimation of model quality. Doing so, gives an evaluation rMSE of 539, well under the value predicted. (The errors range from -751 m. to 854 m. over the 9 observations, and the average absolute value is 427.5 m.)

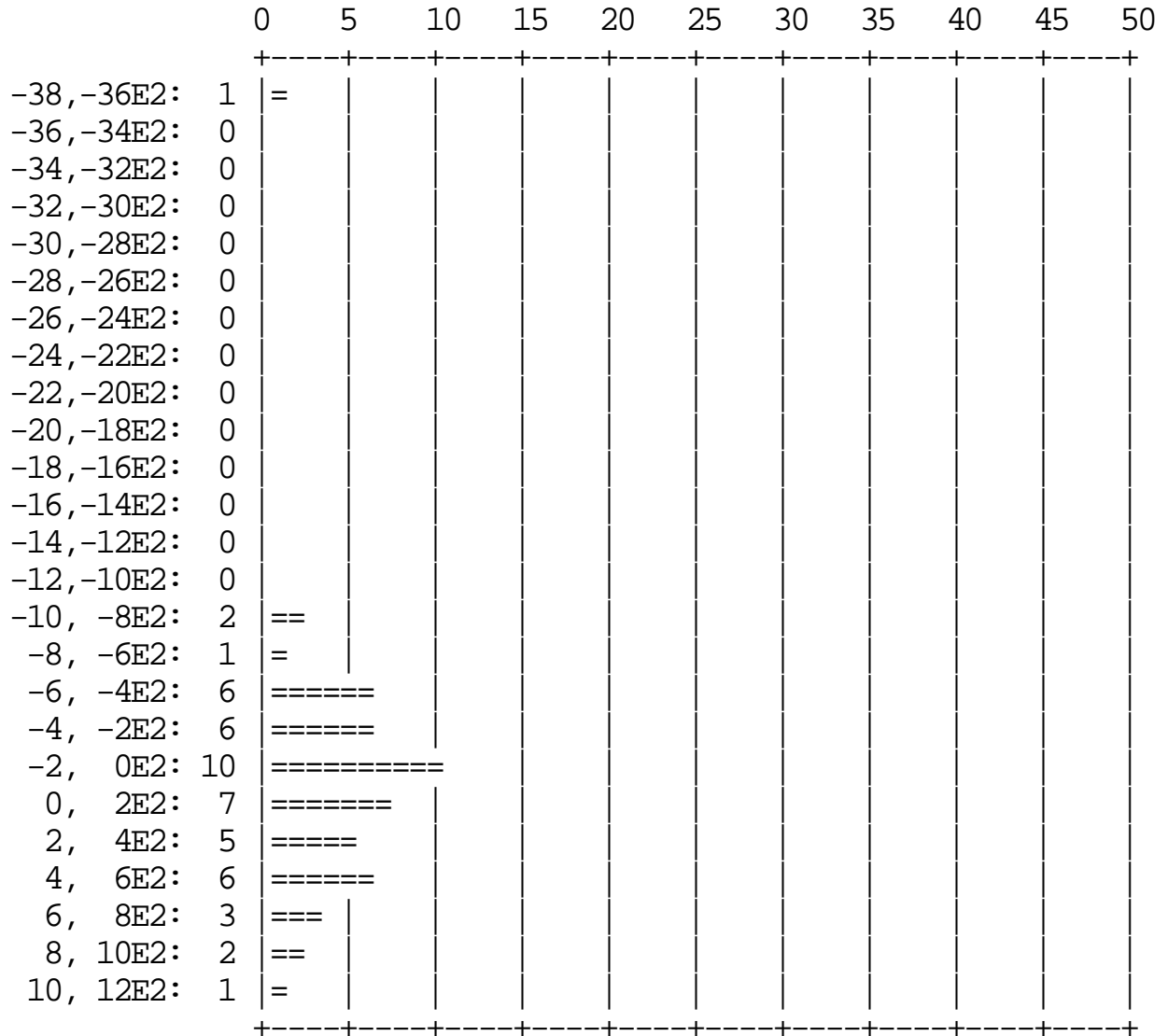
⁴³See the input statistics in the APN or log files.

Table 9. Statistics File Generated by IMP for a Network Model (Task 3)

Entire moon1.dat file read successfully.

*Warning: 1 evaluation point(s) (2% of database) were outside the training region.

Histogram of error distribution for variable # 2:



Each "=" represents one point.

Statistics for variable # 2 output:

Database:	evaluation	modeling
Number of observations:	50	40
Average Ytrue:	33889.0	33010.0
Estimated Ytrue variance:	5.47498e+07	5.68382e+07

Table 9 (Continued)

Estimated Y_{true} standard deviation:	7399.31	7539.11
Maximum Y_{true} :	46442.0	46442.0
Minimum Y_{true} :	18628.0	18628.0
Average error:	-50.6668	
Average squared error:	490508.	
Estimated error variance:	487941.	
Root of predicted squared error:		727.413
Root of average squared error:	700.363	449.025
Estimated error standard deviation:	698.528	
Average absolute error was:	446.929	
Median error was:	-17.0869	

Most positive error was 1046.56 on observation # 20:
variables were: (2) 27092.0 (3) 403.800 (4) 59.1000

Most negative error was -3720.85 on observation # 49:
variables were: (2) 33755.0 (3) 498.600 (4) 66.9000

Would the outlier error have been smaller had IMP been run with data limiters? It is usually helpful to enforce output limits in operational models and to alert an analyst if any input or output repeatedly exceeds its training limits. In this case however, such limits would not have helped. Limiting outputs alone has no effect (the outlier does not err on the side of excess in either direction), and limiting inputs, or both inputs and outputs, is more than twice as bad (in resulting outlier error magnitude) as doing nothing! Clearly, rectangular data limits are no replacement for careful attention to the data space regions represented by the training data, for it is the data alone which communicate to ASPN where the model should be most closely tailored.

Despite the model performing better than advertised (evaluation $rMSE < rPSE$), it is not very accurate. Some small errors due to data precision are unavoidable, but surely we can do better for such a simple problem! Let's look at the data again. The inputs, V and γ , describe a vector in polar coordinates; what about the Cartesian equivalents? All the information that exists about the problem *is* embedded in the original variables, but what if another form were more convenient?

Despite the flexibility of ASPN to build up features from raw variables, their particular representation can matter a great deal. For instance, totally unrestricted angles (which are congruent modulo 360°) are discontinuous (e.g., 10° is as close to 0° as 350°), so we can imagine that, for large angle ranges, degree representation alone may cause difficulty in creating a simple model. We don't need to think about it too long; give the data to ASPN and let it select the inputs that work -- after all, that's its job.

Calculating the data features X' and Y' , for lateral and upward velocity, respectively, can be done by the DBPrep utility.⁴⁴ The resulting model is much better; the rPSE is now under 500 m., but may actually be ignored because there is virtually no fitting error. This (essentially) exact fit, with only 4 coefficients, means we have stumbled across something *explanatory*. Expanding the model obtains the equation $Y = -0.957 + 0.002X' - 0.01Y' + 0.377X'Y'$, which is dominated by the last term. (In this case, the first three model coefficients essentially only undo the effects of input and output normalization.)

Checking our results by analyzing the problem theoretically (a luxury!), we note that the equation for a ballistic object in a vacuum is $X = X'T_f$, where the time of flight

$$T_f = \frac{1}{g} [Y'_0 + \sqrt{Y'_0{}^2 + 2g(Y_f - Y_0)}] \quad (28)$$

$= \frac{2Y'_0}{g}$ for $Y_f = Y_0$. Therefore,

$$X = 2V^2 \sin(\gamma) \cos(\gamma) / g = 2X'Y' / g = 0.377X'Y' \quad (29)$$

for a moon gravity, g , of 5.3087 m/sec².

6.2. Task 2 -- The Inverse Modeling (Control) Problem

Predicting where the cannonball will impact for given cannon muzzle velocity and angle of inclination turned out to be quite straightforward; what of the inverse (and potentially more useful) problem of estimating the angle necessary for a desired impact location? This *control* problem uses the same data as the original, just from a different perspective (as suggested by Figure 1).

⁴⁴Remember the variable identities by adding a header to the input file, or using the *Name* keyword in ASPN. Now, with two more inputs, the *Vstring* becomes "V,Y,4X" instead of "VYXX", *Inputs* must be 4, and the data file may have a new name (such as moonl.dat); otherwise, the control file is as in Table 6.

Using the Cartesian data of the prior task is not really "legal" here, as the features contain direct information about the new output, γ ; instead, we must employ the original data of Table 5. But, a direct run obtains a *terrible* model (score = .415, rPSE = 8.8° , complexity proportion of error = 6%). What went wrong? The control parameters were reasonable, there were no errors or warnings, ... could it have been something with the data?

Running DBPrep to check, we scatterplot the output, γ , vs. the main input, X , to get (a version of) Figure 14. The arc shape of the data suggests that we may not be estimating a single-valued function; instead, it looks as if several cases exist for which more than one γ value will result in a given distance, X . If the other variable, V , can't explain the difference, we may be asking ASPN to model a multiple-valued relation, rather than a function. (In such a case, low-quality networks result as ASPN *compromises* between valid solutions to minimize the squared error of asserting a single value.) Further examination reveals that V is unable to distinguish between γ cases, and reflection suggests why: for a ballistic projectile in a vacuum, there are two initial angles under which many distances may be reached; one above 45° , and one below.

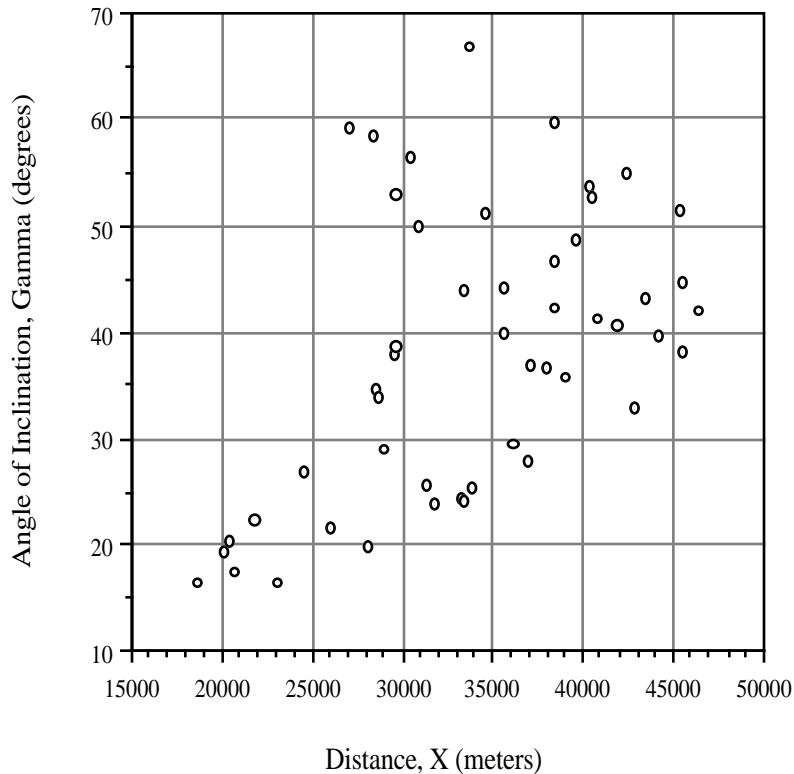


Figure 14. γ vs. X (Task 2)

More low- γ cases (36) exist than high (14), so let's focus on them. There are several ways the training data could be pared; the simplest would be to sort the workspace by γ (using DBPrep) and output the first 36 observations to a new data file. Alternately, to keep the original data intact, one could disable selected observations with the validation *ground* value. (When the *validate* option is in use, ASPN will ignore any cases for which the case index is replaced by the value “-1”, so the input data file could be edited to disable cases with large γ).

Using the data subset, a much better (two-layer) model ensues: a multi-linear node feeding into a double with X . This network has a rPSEp (or $\sqrt{\text{score}}$) of 0.18 -- not bad on scale of 0 (best) to 1 (worst), and an rPSE of 1.66° , less than half of which ($\sqrt{21\%} = 45\%$) is due to fitting error. On the other hand, the network has nine coefficients (when given only 36 cases; a 4:1, rather than 10:1 ratio), and the output is sensitive to changes in the inputs (> 1.0 for each). Both of these indicate possible overfit. Overall, the network synthesized from such slight data seems acceptable; but it would be wise to test it thoroughly over the range of allowable inputs before use.

A graph of the true γ value vs. the model estimate for each of the observations is reassuringly like a straight line (the ideal case). Zooming in to focus on the residuals, or errors, provides us with Figure 15 when the errors are plotted not against one of the inputs, but against the *model output* (to illustrate a point about polynomial approximations). The residual pattern looks much like a sine wave; it appears that the answers could be improved by taking the existing model output and adding a corrective term that depends only on that output. That is, we could append some further function onto the end of the network to make it better on the training data.

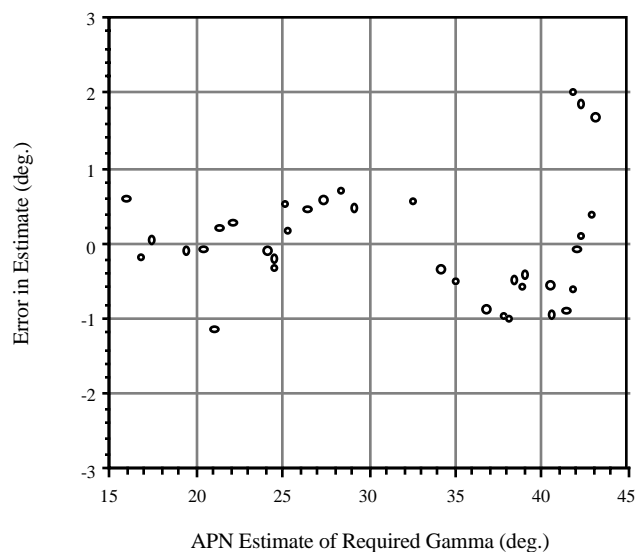


Figure 15. Model Error vs. Estimate (Task 2)

However, one should *rarely, if ever, cascade models!* ASPN itself would have continued making the network model more accurate and more complex if the data could have supported such growth. Adding degrees of freedom to the model, without fairly considering the complexity (data dependency) of such an action, will *almost surely lead to overfit.*

But why the output pattern? From (29) $X = 2V^2 \sin(\gamma) \cos(\gamma) / g$. As $2 \sin(\gamma) \cos(\gamma) = \sin(2\gamma)$

$$\gamma = 0.5 \arcsin\left(\frac{Xg}{\sqrt{2}}\right) \quad (30)$$

Thus, the true relationship involves division, and inverse trigonometric functions. But ASPN employs polynomial elements (almost exclusively), and has no division or arcsine operators; why did it do well? Because polynomials are flexible and powerful basis functions for describing data. Recall power series expansions which use polynomial terms; though they must be infinitely long to be exactly accurate for transcendentals (such as sines and exponentials), the series expansions can achieve arbitrary accuracy for some finite length. Also, the residual of a transcendental is a (smaller) transcendental. That is what we are viewing in this example: ASPN is essentially discovering the underlying nature of the function as reflected in the data and deciding where the best cutoff to its polynomial approximation should be.

6.3. Task 3 -- Impact Estimation in the Atmosphere

Figure 16 summarizes some actual (experimental and simulation) data for projectiles released in an atmosphere from a set of initial velocity (V , ft./sec.), height (H , ft.), and angle of inclination (γ , °) conditions. The cases represented range from fast dive-bombing ($V = 550'$, $\gamma = -60^\circ$) to slower “lofting” ($V = 350'$, $\gamma = 60^\circ$) scenarios. For each of the thirty (V , γ) pairs shown in Figure 16, ten different heights, H , are recorded.⁴⁵ The impact (final) condition is described by four quantities: position (X_f , ft.), time of flight (T_f , sec.), velocity (V_f , ft./sec.), and angle (γ_f , °). Our task is to simultaneously and accurately estimate all four outputs as functions of the three input conditions.

Approaching the problem directly, we set up a control file with the components:

```
InFile   : missile.dat           Training data file (from raw.dat)
Vstring  : 3x, 4y               Variable identification
Outputs  : 4
Inputs   : 3
```

⁴⁵Heights of 3, 5, 8, 10, 15, and 20 thousand feet occur for each (V , γ) pair; the other four heights are scenario dependent, and are taken from the set {200, 500, 750, 1000, 1500, 2000, 4000, 6000, 7000, 9000, 12500}.

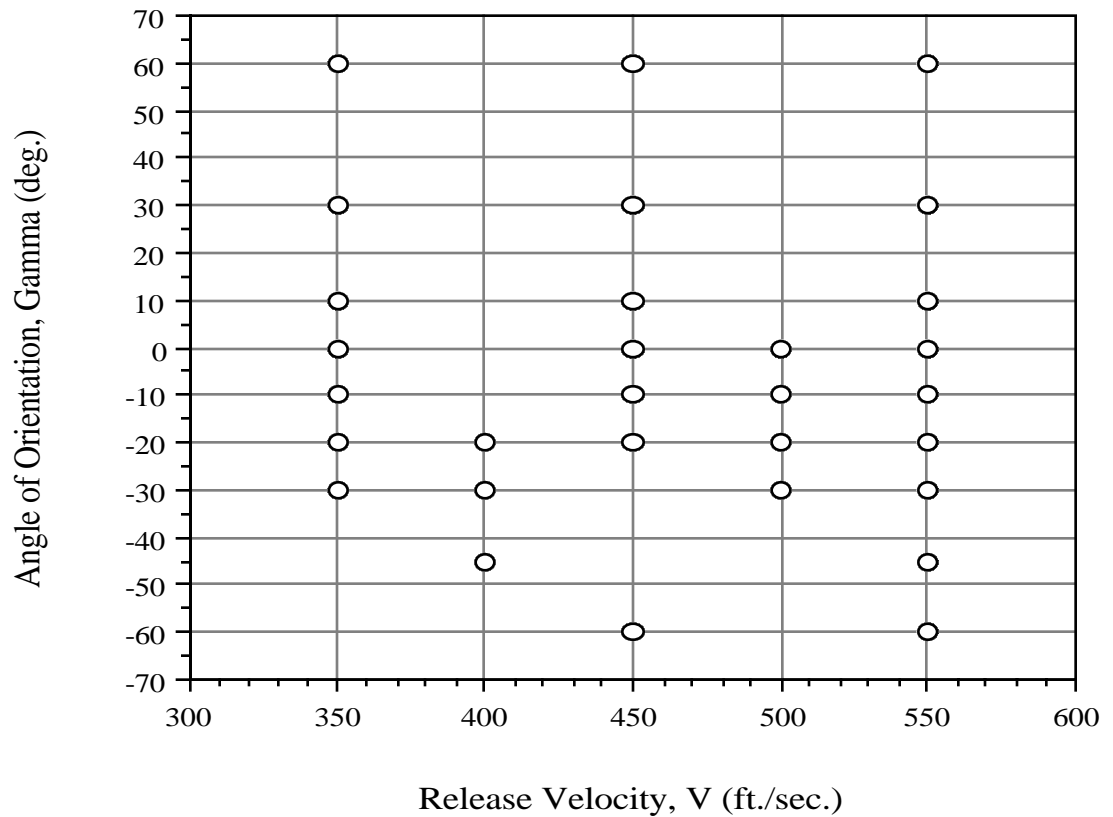


Figure 16. Task 3 Data Regions.

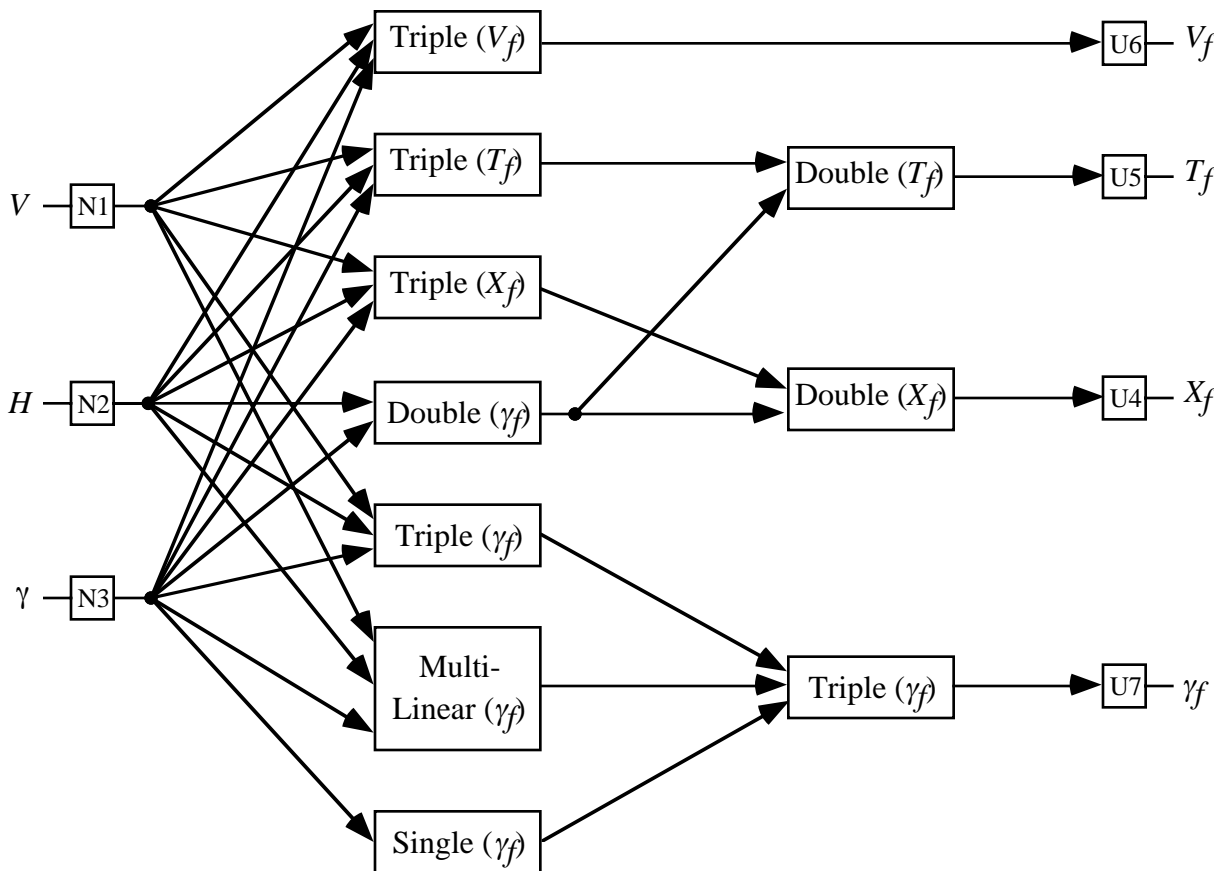
The impact conditions for ten cases, at various release heights, H , are recorded for each (V, γ) pair shown.

As the data are somewhat systematic (the inputs are not truly random, but occupy intersections of an irregular three-dimensional grid), it would be best to reserve a *randomly-selected* portion of the set (using DBPrep) and save the resulting file as “missile.dat”.

Running ASPN, the best model for each of the four networks is displayed on the same screen. Why four at a time? Wouldn't sequential solutions be easier? Yes, but there is a great advantage to using multiple outputs: *node sharing* (enabled by the Borrow argument). The four variables all describe different aspects of a related flight condition; it is reasonable to assume that the value of one (more precisely, a prototype model of one) could be useful to another of the inputs. If such a sub-network is helpful enough as an input to another to overcome the relatively large complexity penalty associated with its coefficients, then ASPN will “cross-pollinate” the networks, and element sharing will occur. This is often a very useful feature.

The network now takes several minutes to be synthesized, due to the increase in cases and outputs, but the wait is worth it, because we get the more interesting networks of Figure 17.⁴⁶ Shown decomposed (as on the screen), it is not possible to tell if any sharing has occurred in the network, but careful examination of the log file reveals that the models for X_f and T_f share element 58 (a double with inputs H and γ), which was trained on γ_f ! Performance measures and normalized sensitivities for the networks are summarized in Table 10.

Depending on the accuracy required, the models for T_f and V_f may be good enough to use, but γ_f required a fair amount of complexity and didn't get as good a fit, while X_f is off by a thousand feet (the output with the greatest error proportion, as seen from $rFSEp$ or from $\sqrt{\text{score}} = rPSEp = rPSE/\sigma_y$). We need to enhance our inputs and try again, focusing on the two worst.



⁴⁶However, one may halt synthesis of the current sub-network, by hitting Ctrl-C, and ASP gives the best model found thus far for that response variable and moves on to any remaining ones! Nodes are labelled by type and goal; note that the X_f and T_f models share a γ_f double.

Table 10. Network Performance Summary

Output	σ_y	σ_{NN}	rPSE	rFSEp	%KP	$\Delta/\Delta V$	$\Delta/\Delta H$	$\Delta/\Delta \gamma$
X_f	7152	2066	1077	.135	20	0.36	1.05	1.08
T_f	13.65	3.08	1.11	.062	42	0.13	0.84	0.71
V_f	163.7	28.9	8.89	.046	27	0.46	0.86	0.17
γ_f	16.33	3.00	1.69	.092	27	0.16	1.05	0.89

The equation in the previous section for the time of flight in a vacuum (say, T_{go}) could come in handy here, as well as the lesson about Cartesian coordinate conversion. Once we get rolling, a whole lot of data features suggest themselves. If they are deterministic calculations *not arrived at from looking at the data* (a distinction that would require their requiring a complexity penalty), we could put them in and let ASPN try them out, sorting through all the possibilities. Perhaps a lot of vacuum-ballistic quantities will provide good jumping-off points for the models. Let's define H , X , and T_{go} as before, as well as:

$$\begin{aligned}
 T_{go}^{-1} &= \frac{1}{T_{go}} & X_{go} &= XT_{go} \\
 S_{go} &= \sqrt{X_{go}^2 + H^2} & \sin(\gamma_T) &= \frac{-H}{S_{go}}
 \end{aligned} \tag{31}$$

and Energy Height, $se = H + \frac{V^2}{2g}$

and augment the (entire) data base (i.e., compute features for both training and evaluation). Now, with the control commands

```

InFile   :250.dat           Training data file
Vstring  : -, 3x, 8x, y--y,- Variable identification
Outputs  :      2
Inputs   :     11
!Nearest
SigmaP   : 1000.           Prior error est for Y1 (Xf)
SigmaP   :      3.         Prior error est for Y2 (gf)
    
```

the next stage of modeling can be attempted.⁴⁷

⁴⁷The NN calculation has been disabled not only to save time, but for accuracy. The multi-dimensional distance calculations implicitly assume variable independence, which is not the case with all the new, derived features.

This time for X_f , the multi-linear element leads the pack until the triples are calculated -- a sign that the data relationships have been simplified, and an indication that at least three inputs are needed to score well (which is reasonable, since the underlying dimensionality of our problem is at least three). After a few minutes, synthesis halts; though now voluntarily "due to lack of network growth on layer 2", and not due to a layer limit (as before). Now, γ_f is approximated by a *triple* of γ , T_{go} , and $\sin(\gamma_T)$, with $\sqrt{\text{score}} = 0.047$ and an rPSE of 0.76° . The X_f model is a simpler combination (a *triad*) of H , H' , and X_{go} with $\sqrt{\text{score}} = 0.032$ and rPSE of 226 ft. -- the majority of which is due to complexity. The new features have helped!

Expanding the network and collecting terms with the Xpress program gives the equation, $X = 252 + .007 H + .065 H' + .938 X_{go}$ (minus terms with negligible coefficients). The first few terms are lightly adjust the main driver: the vacuum ballistic estimate of downrange distance to travel, X_{go} . In fact, our model could be improved by first subtracting that estimate from the actual range, X , in the training data, and modeling the *difference*. This would correspond to *focusing on the unknown part* of the problem; that is, since the bulk of the distance travelled can be explained by simple physics, the question of interest is how the atmosphere and the context of the initial conditions *affect that nominal trajectory*.

Performing this output transformation and running once more, with ΔX_f as the output, rather than X_f , and cutting σ_p in half (since we're doing so well), provides a network using H' , X_{go} , and $\sin(\gamma_T)$, with an rPSE of 127 ft. (again conservative, as complexity = 74%). Note that this technique of "removing the known part" also proves effective on the other three outputs.

6.4. Remarks

The three example modeling tasks described focused on *estimation* (and touched on *control*), using simple trajectory mechanics. A simple problem, "moon ballistics", was addressed from two directions, and the lessons learned were carried into the real problem domain of in-atmosphere firing-table generalization. In the course of the solution iterations, typical modeling problems arose, and it was demonstrated how ASPN and its utility programs can be used to recognize and solve those difficulties. In all modeling tasks, the necessary relationships were *inferred* from the data to create a *generalized* network model that stored key information in a compact and readily retrievable form (i.e., performed *data reduction*) and which was useful for new, but similar cases.

7. SUMMARY AND CONCLUSIONS

This chapter has provided both a survey of induction methods and a focused look at a class of notably effective network approaches. We began with a general discussion of the leading induction techniques and organized them according to the amount of decision making they leave to the computer. We then described promising enhancements to induction methods, such as integrating visual information into the modeling process and employing complexity-penalty metrics. The former takes advantage of human capabilities to perceive structure in lower dimensions, while using the computer to scout ahead for promising projections among the myriad of potential views. The latter enhancement shows how measured trade-offs between complexity and accuracy on training data can provide insight into model selection. We view the employment of a model selection criterion to choose among a broad collection of families of competing models as the most general form of induction. Unfortunately such an ambitious approach is only contemplated but not yet realized. Clearly however, as methods to discover structure in data become even more computer intensive, the criteria used to judge between competing explanations of the data must become more sophisticated as well. In particular, as thousands of candidate models (many with nonlinear parameters or weighted cases) are considered, performance standards must be raised before considering accuracy differences between models to be significant.

The last two Sections of this chapter focused on a particularly promising class of induction techniques – polynomial networks. These methods developed out of work in neural networks and adaptive systems and are known by names such as GMDH, PNETTR, and ASPN. Multi-layer polynomial networks are constructed in a data-driven manner from nodes whose outputs are polynomial functions of their inputs. We described the range of approaches to constructing these networks and, in the process, provided some of the history of their creation and use. Current polynomial techniques are not without their weaknesses and we noted these and suggested methods to enhance their performance. Finally, we provided detailed examples of the usefulness of polynomial networks, and specifically ASPN, in forecasting, design, and control problems. These examples served to illustrate the power of the ASPN technique as well as potential pitfalls and their solution.

We derive several conclusions from the material in this chapter. First, the variety of approaches to induction provide contemporary researchers with an impressive array of methods to employ across a startlingly wide range of applications. Problems in pattern recognition (e.g. character recognition and acoustic signal processing) are beginning to yield to a number of these techniques. However, there remain a host of applications that are well beyond the scope of today's techniques. Perhaps, most frustrating are tasks which are easily solved by humans (once viewed

appropriately) but remain seemingly intractable to automated methods. Though such problems could well remain outside of the effective range of inductive modeling for some time, there is evidence that approaches such as polynomial networks, ANNs, decision trees (e.g. CART) and some contributory methods, have the capability to produce excellent results on induction problems that heretofore have resisted solution.

Second, impressive improvements in induction can sometimes be achieved with rather straightforward enhancements. Certainly the use of visual information represents one example. The effective employment of model selection within ASPN is another. This latter enhancement provided significant performance gains over the earlier GMDH method. The wide variety of contemporary induction techniques can encourage a researcher to try one after another in pursuit of the "best" model. Often however, this leads to disappointment with the whole group of methods, because none of them yet represent an all-purpose tool. However, experience in applications suggests that potentially many approaches can perform well, if the data are carefully analyzed and transformed, and if the method is modified (perhaps by creating a hybrid approach with another method) to account for peculiarities in the problem. Hence, a good research strategy would be to first seek a good match between problem characteristics and induction algorithm(s) (Section 2.2) and then look for enhancements to that technique or set of techniques that can further improve performance (e.g., Section 5.5).

Inducing general rules or models from specific cases is a process central to the scientific method, and it is our experience that research into developing and improving automated methods is a challenging and rewarding path.

REFERENCES

- Aitchison, J. (1975). Goodness of Prediction Fit, *Biometrika* **62** no. 3: 547-554.
- Akaike, H. (1970). Statistical Predictor Identification, *Annals of the Institute of Statistical Mathematics* **22**: 203-217.
- Akaike, H. (1973). Information Theory and an Extension of the Maximum Likelihood Principle. In *Proceedings of the 2nd International Symposium on Information Theory*, P.N. Petrov and F. Csaki (Eds.), Kiado Academy, Budapest: 267-281.
- Amemiya, T. (1980). Selections of Regressors, *International Economic Review* **21**: 331-354.
- Anscomb, F.J. (1973). Graphs in Statistical Analysis. *American Statistician* **27**: 17-21.
- Asimov, D. (1985). The Grand Tour: A Tool for Viewing Multidimensional Data. *SIAM J. on Scientific and Statistical Computing* **6**: 128-143.
- Barron, A.R. (1984). Predicted Squared Error: A Criterion for Automatic Model Selection. *Self-Organizing Methods in Modeling: GMDH Type Algorithms*, S.J. Farlow, Ed., Marcel Dekker, New York: 87-103.
- Barron, A.R. (1985). Logically Smooth Density Estimation. PhD thesis, Dept. Electrical Engineering, Stanford.
- Barron, A.R. (1991). Complexity Regularization with Application to Artificial Neural Networks, in *Nonparametric Functional Estimation and Related Topics*, G. Roussas (ed.), Kluwer, Netherlands: 561-576.
- Barron, R.L. (1975). Learning Networks Improve Computer-Aided Prediction and Control, *Computer Design*, August: 65-70.
- Barron, R.L., A.N. Mucciardi, F.J. Cook, J.N. Craig, A.R. Barron (1984). Adaptive Learning Networks: Development and Application in the United States of Algorithms Related to GMDH, Ch. 2 in *Self-Organizing Methods in Modeling: GMDH Type Algorithms*. S.J. Farlow (ed.), Marcel Dekker, New York: 25-65.
- Barron, R.L., D. Abbott (1988). Use of Polynomial Networks in Optimum, Real-time, Two-point Boundary Value Guidance of Tactical Weapons, *Proc. Military Computing Conf.*, May 3-5, Anaheim, CA.
- Barron, R.L., R.L. Cellucci, P.R. Jordan, N.E. Beam, P. Hess, and A.R. Barron (1990). Applications of Polynomial Neural Networks to FDIE and Reconfigurable Flight Control, *National Aerospace Electronics Conf.*, Dayton, OH, May 23-25 [Best Paper Award].
- Belsley, D.A. (1991). *Conditioning Diagnostics: Collinearity and Weak Data in Regression*. Wiley, New York.
- Belsley, D.A., E. Kuh, R.E. Welsch (1980). *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. Wiley, New York.

- Bickel, P., K. Doksum (1981). An Analysis of Transformations Revisited, *Journal of the American Statistical Association* **76**: 296-311.
- Bloomfield, P., W.L. Steiger (1983). *Least Absolute Deviations: Theory, Applications, and Algorithms*. Birkhauser, Boston.
- Bock, M.E. (1992). Nonparametric Regression Using Wavelets, *Proc. 24th Symposium on the Interface: Computing Science and Statistics*, College Station, Texas, March 18-21.
- Bozdogan, H. (1988). ICOMP: A New Model Selection Criterion, *Classification and Related Methods of Data Analysis*, H.R. Bock (ed.), North Holland, Amsterdam: 599-608.
- Breiman, L. (1988). Submodel Selection and Evaluation in Regression -- the Conditional Case and Little Bootstrap, *Tech. Rpt.* 169, Dept. Statistics, UC Berkeley, CA.
- Breiman, L. (1991a). The π Method for Estimating Multivariate Functions from Noisy Data (with discussion), *Technometrics* **33** no. 2: 125-160..
- Breiman, L. (1991b). Hinging Hyperplanes for Regression, Classification, and Function Approximation, *Tech. Rpt.* 324, Dept. Statistics, UC Berkeley, CA.
- Breiman, L., D. Freedman (1983). How Many Variables Should be Entered in a Regression Equation? *Journal of the American Statistical Association* **78**: 131-136.
- Breiman, L., J.H. Friedman (1985). Estimating Optimal Transformations for Multiple Regression and Correlation (with discussion), *Journal of the American Statistical Assoc.* **80**: 580-619.
- Breiman, L., J.H. Friedman, R.A. Olshen, C.J. Stone (1984). *Classification and Regression Trees*. Wadsworth & Brooks, Monterey, California.
- Broomhead, D.S., D. Lowe (1988). Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems* **2**: 321-355.
- Brown, D.E., J.F. Elder IV, C.L. Pittard (1992). Visualizing Correlation Decision Making in Data Fusion Problems, *Proc. 24th Symposium on the Interface: Computing Science and Statistics*, College Station, Texas, March 18-21.
- Brown, D.E., C.L. Pittard (1989). A Parallel Approach for Improved Tree Structured Classifier Construction, *Proceedings of the Annual Conference of the International Association of Knowledge Engineers*, College Park, Maryland, June.
- Cabrera, J., and D. Cook (1992). Projection Pursuit Indices based on Fractal Dimension, *Proc. 24th Symposium on the Interface: Computing Science and Statistics*, College Station, Texas, March 18-21.
- Cellucci, R.L., P. Hess (1990). Techniques for Developing and Applying Polynomial Network Synthesis Software, *Proc. 22nd Symposium on the Interface: Computing Science and Statistics*, East Lansing, MI, April.
- Chaudhuri, P. M-C. Huang, W-Y. Loh, R. Yao (1990). Regression Trees and Function Estimation, *Tech. Report* 871, Dept. Statistics, Univ. Wisconsin-Madison, Dec.

- Charnes, A., W.W. Cooper, R.O. Ferguson (1955). Optimal Estimation of Executive Compensation by Linear Programming. *Management Science* **1**: 138-151.
- Chen, S., C.F.N. Cowan, P.M. Grant (1991). Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks, *IEEE Trans. on Neural Networks* **2**, no. 2: 302-309.
- Cleveland, W.S., S.J. Devlin (1988). Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting, *J. Amer. Stat. Assoc.* **83** #403: 596-610.
- Cleveland, W.S., P. Diaconis, R. McGill (1982). Variables on Scatterplots Look More Highly Correlated When the Scales Are Increased, *Science* **216**: 1138-1141.
- Cleveland, W.S., M.E. McGill (1988). *Dynamic Graphics for Statistics*. Wadsworth, Inc., California.
- Corruble, V. (1992). A Comparison of Methods used in Classification, Master's Thesis, Dept. Systems Engineering, Univ. Virginia, Charlottesville, May.
- Cover, T.M., P.E. Hart (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory* **13**: 21-27.
- Cover, T.M. (1974). The Best Two Independent Measurements are Not the Two Best, *IEEE Trans. Systems, Man, and Cybernetics*, Jan.
- Craven, P., G. Wahba (1979). Smoothing Noisy Data with Spline Functions: Estimating the Correct Degree of Smoothing by the Method of Generalized Cross-Validation, *Numerische Mathematik* **31**: 317-403.
- Darwin, C. (1859). *The Origin of the Species*. (6th Ed. reprint, 1962, Collier Books, New York).
- Denning, P.J. (1992). Genetic Algorithms, *American Scientist* **80**: 12-14.
- Denton, M. (1985). *Evolution: A Theory in Crisis*. Adler and Adler, Maryland.
- Diaconis, P., B. Efron (1983). Computer-Intensive Methods in Statistics, *Scientific American* **248**: 116-126.
- Dordick, R.L. (1988). Understanding the 'Go' of it. *IBM Research Magazine*, Winter: 2-7.
- Draper, N.R., H. Smith (1966). *Applied Regression Analysis*. Wiley, New York.
- Duan, N., K.C. Li (1991). Slicing Regression: A Link-Free Regression Method, *Ann. Stat.* **19** no. 2: 505-530.
- Duffy, J.J., M.A. Franklin (1975). A Learning Identification Algorithm and Its Application to an Environmental System, *IEEE Trans. Systems, Man, and Cybernetics* **5** no. 2: 226-240.
- Efron, B. (1964). The Perceptron Correction Procedure in Nonseparable Situations, *Rome Air Dev. Center Tech. Doc. Rept.*, RADC-TDR-63-533, Feb., Rome, NY.
- Elder, J.F. IV (1980). *Network: A Program for Unravelling PNETTR Models*, Adaptronics Inc., Internal Report, McLean, Virginia.

- Elder, J.F. IV (1985). *User's Manual: ASPN: Algorithm for Synthesis of Polynomial Networks*, (4th Edition, 1989) Barron Associates, Inc., Stanardsville, Virginia.
- Elder, J.F. IV (1990). Feature Elimination Using 'High-Order Correlation', *Proc. Aerospace Applications of Artificial Intelligence*, Dayton, OH, Oct. 29-31 p. 65-72.
- Elder, J.F. IV (1992a). A Novel Efficient Global Search Algorithm for Multiple Dimensions, *Proc. 24th Symposium on the Interface: Computing Science and Statistics*, College Station, Texas, March 18-21.
- Elder, J.F. IV (1992b). Assisted Visual Search for Data Structure, *Proc. 24th Symposium on the Interface: Computing Science and Statistics*, College Station, Texas, March 18-21.
- Elder, J.F. IV (this volume). Perceptrons, Regression, and Global Network Optimization, *Advances in Control Networks and Large Scale Parallel Distributed Processing Models (Vol.2)*, M.D. Fraser (Ed.), Ablex, Norwood, NJ.
- Elder, J.F. IV, R.L. Barron (1988) Automated Design of Continuously-Adaptive Control: The "Super-Controller" Strategy for Reconfigurable Systems, *Proc. American Control Conf.*, Atlanta, GA, June 15-17.
- Elder, J.F. IV, M.T. Finn (1991). Creating 'Optimally Complex' Models for Forecasting. *Financial Analysts Journal*, Jan/Feb, pp. 73-79.
- Everitt, B.S. (1974) *Cluster Analysis*. Wiley, New York.
- Fan, J. (1991). Design-Adaptive Nonparametric Regression, *Tech. Report*, Dept. Statistics, UNC Chapel Hill.
- Faraway, J.J. (1991). On the cost of Data Analysis. *Tech. Report 199*, Dept. Statistics, Univ. Michigan, Ann Arbor.
- Farlow, S.J. (1984), Ed. *Self-Organizing Methods in Modeling: GMDH Type Algorithms*. Marcel Dekker, New York.
- Feder, P.I. (1975). The Log Likelihood Ratio in Segmented Regression, *The Annals of Statistics* **3**: 84-97.
- Fenster, M.S., R. Dolan, J.F. Elder IV (1992). A New Method for Predicting Shoreline Positions from Historical Data. *Journal of Coastal Research*, [in press].
- Fogel, L.J., A.J. Owens, M.J. Walsh (1966). *Artificial Intelligence through Simulated Evolution*, Wiley, New York.
- Friedman, J.H. (1988). Fitting Functions to Noisy Scattered Data in High Dimensions, *Proc. Computing Science and Statistics: 20th Symposium on the Interface*, Reston, VA.
- Friedman, J.H. (1991). Multiple Adaptive Regression Splines (with discussion), *Annals of Statistics* **19**: 1-141.
- Friedman, J.H., B.W. Silverman (1989). Flexible Parsimonious Smoothing and Additive Modeling, *Technometrics* **31**, no. 1: 3-21.

- Friedman, J.H., W. Stuetzle (1981). Projection Pursuit Regression, *J. of the American Statistical Assoc.* **76**, no. 376: 817-823.
- Friedman, J.H., J.W. Tukey (1974). A Projection Pursuit Algorithm for Exploratory Data Analysis, *IEEE Trans. Computers* **23**: 881-889.
- Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition (2nd Ed.)*. Academic Press, New York.
- Fulcher, G.E., D.E. Brown (1991). A Polynomial Network for Predicting Temperature Distributions, *Tech. Report IPC-TR-91-008*, Institute for Parallel Computation, University of Virginia, June.
- Furnival, G.M., R.W. Wilson, Jr. (1974). Regressions by Leaps and Bounds. *Technometrics* **16**:4, pp. 499-511.
- Gabor, D., P.L. Wilby, R. Woodcock (1959). A Universal Non-linear Filter, Predictor, and Simulator which Optimizes Itself by a Learning Process, *Journal of the IEE*, Oct. 17.
- Gilstrap, L.O., S.R. Goldschmidt, J.F. Elder IV (1986). Estimating Signals with Polynomial Filters, Barron Assoc., Inc. *Tech. Report* for Century Computing, Inc. under P.O. 5035 of F33615-84-C-3609, March 6.
- Golub, G.H., C.F. Van Loan (1989). *Matrix Computations (2nd Ed.)*. Johns Hopkins University Press, Baltimore, MD.
- Green, D.G., R.E. Reichelt, R.H. Bradbury (1988). Statistical Behaviour of the GMDH Algorithm, *Biometrics* **44**: 49-69.
- Hastie, T., R. Tibshirani (1985). Discussion of "Projection Pursuit" by P. Huber, *The Annals of Statistics* **13**: 502-508.
- Hastie, T., R. Tibshirani (1986). Generalized Additive Models (with discussion), *Statistical Science* **1**: 297-318.
- Hecht-Nielsen, R. (1990). *Neurocomputing*. Addison-Wesley, Reading, Mass.
- Hinkley, D.V. (1969). Inference About the Intersection in Two-Phase Regression, *Biometrika* **56**: 495-504.
- Hinkley, D.V. (1970). Inference in Two-Phase Regression, *Journal of the American Statistical Association* **66**: 736-743.
- Hjorth, U. (1989). On Model Selection in the Computer Age, *Journal of Statistical Planning and Inference* **23**: 101-115.
- Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. U. Michigan Press, Ann Arbor.
- Hurvich, C.M., C.L. Tsai (1989). Regression and Time Series Model Selection in Small Samples, *Biometrika* **76**, no. 2: 297-307.

- Hurvich, C.M., C.L. Tsai (1990). The Impact of Model Selection on Inference in Linear Regression, *The American Statistician* **44**, no. 3: 214-217.
- Ivakhnenko, A.G. (1968). The Group Method of Data Handling -- A Rival of the Method of Stochastic Approximation, *Soviet Automatic Control* **3**: 43-71.
- Ivakhnenko, A.G. (1971). Polynomial Theory of Complex Systems, *IEEE Trans. Systems, Man, and Cybernetics* **1** no. 4: 364-378.
- Ivakhnenko, A.G. (1984). Past, Present, and Future of GMDH, Ch. 5 in *Self-Organizing Methods in Modeling: GMDH Type Algorithms*. S.J. Farlow (ed.), Marcel Dekker, New York: 105-119.
- Judd, S.J. (1990). *Neural Network Design and the Complexity of Learning*. MIT Press, MA.
- Kolmogorov, A.N. (1957). On the Representation of Continuous Functions of Several Variables by Superpositions of Continuous Functions of One Variable and Addition, *Dokladi* **114**: 679-681.
- Kozek, A.S., E.F. Schuster (1991). On 'Fit the Short Curve' Principle for Smoothing Nonparametric Estimators, *Proc. 23rd Symposium on the Interface: Computing Science and Statistics*.
- Lloyd, D. K., M. Lipow (1962), *Reliability: Management, Methods, and Mathematics*, Prentice-Hall, Englewood Cliffs: 360.
- Lorentz, G.G. (1966). *Approximation of Functions*. Holt, Rinehart, and Winston, New York.
- Mallows, C.L. (1973). Some Comments on Cp. *Technometrics* **15**: 661-675.
- Minsky, M., S. Papert (1969). *Perceptrons*. MIT Press, MA (3rd Ed., Science Press, 1988).
- Montgomery, G.J., K.C. Drake (1990). Abductive Networks, *Proc. SPIE*, April 1990.
- Morton, S.C. (1989). Interpretable Projection Pursuit, *Tech. Report* 106, Dept. Statistics, Stanford Univ., CA.
- Mucciardi, A.N. (1982). ALN 4000 Ultrasonic Pipe Inspection System. *Nondestructive Evaluation Program: Progress in 1981*, Electric Power Research Institute Report NP-2088-SR, Jan., Adaptronics, Inc., McLean, VA.
- Mucciardi, A.N., E.E. Gose (1971). A Comparison of Seven Techniques for Choosing Subsets of Pattern Recognition Properties, *IEEE Trans. on Comp.* **20**: 1023-1031.
- Mucciardi, A.N., E.E. Gose (1972). An Automatic Clustering Algorithm and its Properties in High-Dimensional Spaces, *IEEE Trans. Systems, Man, and Cybernetics* **2** no. 2: 247-254.
- Parzen, E. (1962). On Estimation of a Probability Density Function and Mode, *Annals Math. Stat.* **33**: 1065-1076.
- Parzen, E. (1977). Multiple Time Series Modeling: Determining the Order of Approximating Autoregressive Schemes. In *Multivariate Analysis IV*, P. Krishnaiah (Ed.), North-Holland, Amsterdam: 283-295.

- Peixoto, J.L. (1987). Hierarchical Variable Selection in Polynomial Regression Models, *The American Statistician* **41**, no. 4: 311-313.
- Pope, P.T., J.T. Webster (1972). The Use of an F-Statistic in Stepwise Regression Procedures, *Technometrics* **14**, no. 2: 327-340.
- Prager, M. H. (1984). An SAS Program for Simplified GMDH Models, Ch. 16 in *Self-Organizing Methods in Modeling: GMDH Type Algorithms*. S.J. Farlow (ed.), Marcel Dekker, New York: 291-315.
- Prager, M. H. (1988). Group Method of Data Handling: A New Method for Stock Identification. *Trans. American Fisheries Society* **117**: 290-296.
- Press, W.H., B.P. Flannery, S.A. Teukolsky, W.T. Vetterling (1988). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge Univ. Press.
- Rencher, A.C., F.C. Pun (1980). Inflation of R^2 in Best Subset Regression, *Technometrics* **22**: 49-53.
- Rissanen, J. (1978). Modeling by Shortest Data Description. *Automatica* **14**: 465-471.
- Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry*. World Scientific, Teaneck, New Jersey.
- Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, *Psychological Review* **68**: 386-408.
- Rumelhart, D.E., G.E. Hinton, R.J. Williams (1986). Learning Internal Representations by Error Propagation. Ch. 8 of *Parallel Distributed Processing*, Rumelhart, McClelland, and the PDP Research Group, MIT Press.
- Sacks, J., S.B. Schiller, W.J. Welch (1989). Designs for Computer Experiments, *Technometrics* **31** no. 1: 41-47.
- Schwarz, G. (1978). Estimating the Dimension of a Model, *Annals of Statistics* **6**: 461-464.
- Scott, D.W. (1985). Averaged Shifted Histograms: Effective Nonparametric Density Estimators in Several Dimensions, *Annals of Statistics* **13**, no. 3: 1024-1040.
- Seber, G.A.F., C.J. Wild (1989). *Nonlinear Regression*. Wiley, New York.
- Sethi, I.K. (1990). Entropy Nets: From Decision Trees to Neural Networks, *Proceedings of the IEEE* **78** no. 10: 1605-1613.
- Shewhart, M. (1992). A Neural-Network-Based Tool, *IEEE Spectrum*, Feb.: 6.
- Shibata, R. (1980). Asymptotically Efficient Selection of the Order of the Model for Estimating Parameters of a Linear Process, *The American Statistician* **34**: 147-164.
- Skeppstedt, A., L.Ljung, M. Millnert (1992). Construction of Composite Models from Observed Data, *International Journal of Control* **55** no. 1: 141-152.
- Specht, D.F. (1967). Generation of Polynomial Discriminant Functions for Pattern Recognition. *IEEE Transactions on Electronic Computers* **16** no. 3: 308-319.

- Stone, M. (1974). Cross-Validatory Choice and Assessment of Statistical Predictors (with discussion), *Journal of the Royal Statistical Society, Series B* **36**: 111-147.
- Swayne, D.E., D. Cook, A. Buja (1991). *A User's Manual for XGobi, a Dynamic Graphics Program for Data Analysis Implemented in the X Window System* (Release 2), Bellcore.
- Tenorio, M.F., and W.T. Lee (1989). Self-Organizing Neural Networks for the Identification Problem, in *Advances in Neural Information Processing Systems*, D.S. Touretzky (ed.), Morgan Kauffman, 57-64.
- Tierney, L. (1990). *Lisp-Stat: An Object Oriented Environment for Statistical Computing and Dynamic Graphics*, Wiley, New York.
- Wallace, C.S., D.M. Boulton (1968). An Information Measure for Classification, *Computer Journal* **11** no. 2.
- Werbos, P. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, PhD thesis, Harvard, August.
- Willett, J.B., J.D. Singer (1988). Another Cautionary Note About R^2 : Its Use in Weighted Least-Squares Regression Analysis, *The American Statistician* **42**, no. 3: 236-238.