Nonparametric Selection of Input Variables For Connectionist Learning

Brian Bonnlander

CU-CS-812-96

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

ANY OPINIONS, FINDINGS, AND CONCLUSIONS OR RECOMMENDATIONS EXPRESSED IN THIS PUBLICATION ARE THOSE OF THE AUTHOR(S) AND DO NOT NECESSARILY REFLECT THE VIEWS OF THE AGENCIES NAMED IN THE ACKNOWLEDGMENTS SECTION.

NONPARAMETRIC SELECTION OF INPUT VARIABLES FOR CONNECTIONIST LEARNING

by

BRIAN BONNLANDER

B.S., Stanford University, 1990

M.S., University of Colorado, 1992

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

1996

This thesis for the Doctor of Philosophy degree by
Brian Bonnlander
has been approved for the
Department of
Computer Science
by

____
Andreas S. Weigend

____
Michael C. Mozer

Date ____________________

Bonnlander, Brian (Ph. D., Computer Science)
Nonparametric Selection of Input Variables for Connectionist Learning
Thesis directed by Assistant Professor Andreas S. Weigend

When many possible input variables to a statistical model exist, removing unimportant inputs can improve the model's performance significantly. A new method for selecting input variables is proposed. Components for the proposed method include:

- Mutual information as a relevance measure
- Kernel density estimation for estimating probabilities
- Forward selection as an input variable search method

Analysis of mutual information shows that it is a natural measure of input variable relevance. It is a more general measure of input variable relevance than expected conditional variance. Under certain conditions, the two measures order the relevance of input variable subsets in precisely the same manner, but these conditions do not generally hold. An unbiased approximation to mutual information exists, but it is unbiased only if the underlying probabilities are exact.

Analysis of kernel density estimation shows that the accuracy of mutual information estimates depends directly on how densely populated the points in the data set are. However, for a range of explored problems, the relative ordering of mutual information estimates remains correct, despite inaccuracies in individual estimates.

Analysis of forward selection explores the amount of data required to select a certain number of relevant input variables. It is shown that in order to select a certain number of relevant input variables, the amount of required data increases roughly exponentially as more relevant input variables are considered. It is also shown that the chances of forward selection ending up in a local minimum are reduced by bootstrapping the data.

Finally, the method is compared to two connectionist methods for input variable selection: Sensitivity Based Pruning and Automatic Relevance Determination. It is shown that the new method outperforms these two when the number of independent, candidate input variables is large. However, the method requires the number of relevant input variables to be relatively small. These results are confirmed on a number of real world prediction problems, including the prediction of energy consumption in a building, the prediction of heart rate in a patient with sleep apnea, and the prediction of wind force in a wind turbine.

CONTENTS

FIGURES

TABLES

CHAPTER 1

INTRODUCTION

Engineering is, in a nutshell, the process of solving a problem by converting knowledge into the construction of a device. In some cases, knowledge can be formulated precisely, and a device can be designed through mathematical or logical deduction. In other cases, understanding of the solution is less precise; characteristics of the device may be determined only by collecting a set of observations, or data, and applying them to some method of statistical modeling. Whatever imprecise knowledge of the problem might be available is incorporated into the statistical model. While this may result in a device whose specifications are not ideal, tolerances for the device are often close enough to ideal to result in satisfactory performance.

This thesis is concerned with a certain class of statistical modeling problems. This class involves those problems where large amounts of data are readily available, and understanding of the problem is weak enough to suggest a nonlinear, nonparametric statistical model. Such a model is often termed a **connectionist network**.

For such problems, success in the statistical modeling process sometimes depends critically on eliminating input variables to the model that provide little or no information about the output variable. Including these irrelevant input variables can increase the number of parameters in the model to the point where none of the model parameters can be known with enough certainty to result in good performance. The practical effect of removing input variables is the reduction in the **variance** of model parameters. The more irrelevant input variables are removed, the greater the possible reduction in variance, and the more accurate the model will be.

Several methods have already been proposed in the connectionist literature for eliminating irrelevant input variables. These include Sensitivity Based Pruning (Moody and Utans, 1992; Moody, 1994; Utans et al., 1995) and Automatic Relevance Determination (MacKay, 1993; Neal, 1995). This thesis shows that while these methods work well for problems with a small to medium number of input variables, their performance worsens as the number of possible input variables grows large. The methods lose power in being able to distinguish relevant from irrelevant input variables, due to the increase in model parameters that must be estimated as part of the input variable selection process.

1.1 Description of the Proposed Method

I propose an alternative method for input variable selection that works when the set of possible input variables is large. The method's implementation differs from its competitors' in two important ways. First, the relevance of a set of inputs is assessed using nonparametric, kernel density estimation and mutual information, rather than more parametric, error based estimation schemes. Using kernels allows the relevance of input variables to be computed straightforwardly; the competitors rely upon gradient descent in some predetermined weight space, which can be slow to converge. Mutual information is also a more general definition of relevance than standard, error based measures. The proposed method is most appropriate when the available data are plentiful. For some problems, where data are less plentiful, the data set size can be boosted using "hints" (Abu-Mostafa, 1995).

The second implementation difference is the freedom to choose a particular search method for finding a minimal set of relevant input variables. While Sensitivity Based Pruning and Automatic Relevance Determination require the consideration of all possible input variables as a starting point, the method I propose can begin with the examination of individual input variables or small groups of them, depending on knowledge of the problem or computational considerations.

1.2 Summary of Method Analysis

The thesis analyzes the proposed method by breaking it down into its basic components and examining the components separately. The basic components of the method include:

- An analytic measure of input variable relevance
- A sample based approximation of the relevance measure
- A search method for choosing from among input variables
- A method for estimating the number of modes, or "bumps", in the conditional output distribution.

Understanding the need for each component requires some elaboration. The first component, the analytic relevance measure, can be viewed as the central component of the method; it is the theoretical measure upon which the measurement of input variable relevance is based. The other components are simply there to facilitate the measure's practical application to data. The second component is necessary because the relevance measure is theoretically motivated, and there is no immediate way to base the measure upon a set of data, as opposed to more complete knowledge of the input variables. The third component allows a relevant subset of input variables to be found efficiently. The last component helps in deciding upon an appropriate statistical model for some selected subset of relevant input variables. Many connectionists automatically assume that the conditional output density is unimodal by using a squared error measure (Section 2.3). The generality of mutual information as a relevance measure, however, comes with a price: it does not make any assumptions about the modality of the conditional output density. The number of modes must be determined in a separate step.

For other input variable selection methods, the above components are combined in ways that make their individual analysis difficult or impossible. However, the components are quite separable for the method proposed here, and many choices are actually possible for each component. The particular implementation choices I make in this thesis are intended to be appropriate and computationally efficient for a wide variety of statistical modeling problems. As the thesis will show, however, the wide applicability of the choices comes at the expense of requiring large amounts of data.

In the same order as the above list, the implementation choices explored in this thesis are

- Mutual information as a relevance measure
- Kernel density estimation, for applying mutual information to data
- Forward selection as an input variable search method
- Kernel based mode estimation.

The following subsections summarize the results of the analysis on a component by component basis.

1.2.1 Mutual Information

In Chapter 4, it is shown that mutual information is a natural measure of input variable relevance. The chapter goes on to show that under certain conditions, mutual information and expected conditional variance (upon which sum squared error is based) order the relevance of input variable subsets in precisely the same manner. The proof establishes sufficient conditions for the two measures to order the relevance of input variable subsets identically. Following this, a counterexample shows that the two measures are not always related. It is argued that mutual information is a more general measure of input variable relevance than expected conditional variance, since it still works for multimodal probability density functions.

The chapter also discusses the difficulty of estimating mutual information analytically, even when probability densities are known. This is due to the measure involving an integration over marginal density functions. The required integration is intractable in many cases. However, the chapter shows that it is possible to fall back upon single integration to obtain marginal densities, and then to approximate mutual information from a set of sampled points. With the availability of exact probabilities for the sample, estimates of mutual information converge quickly to the true value as a function of the sample size.

1.2.2 Kernel Density Estimation

Chapter 5 moves on to the consequences of using kernel density estimation when exact probability densities are not available, as they were in Chapter 4. It shows that the accuracy of mutual information estimates depends directly on how dense the points are. As the sample size increases, or as fewer input variables are considered, the accuracy of the mutual information estimates increases. However, despite progressively less accuracy as more and more input variables are considered, the correct ordering of mutual information estimates remains preserved for a range of prediction problems. This result suggests that the selection of input variables based on mutual information and kernel density estimation is possible in practice. Lastly, the interaction between sample size and accurate relevance estimation is explored. It is shown that data set size requirements increase roughly exponentially as a function of the number of relevant input variables.

1.2.3 Forward Selection Chapter 5 also explores whether forward selection, as a search algorithm, is capable of finding subsets of input variables with high relevance. It is possible, despite the appropriateness of mutual information as a relevance measure, for forward selection to end up in a local minimum. On both a linear and nonlinear prediction problem, it is shown that the chances of forward selection ending up in a local minimum depend directly on the accuracy of relevance estimates: the more accurate the estimates, the less likely forward selection ends up in a local minimum.

1.2.4 Mode Estimation Chapter 6 discusses the importance of knowing the number of modes in the conditional output distribution. Most connectionists assume a unimodal distribution, but using mutual information implies no such assumption. For this reason, it is more general than error based measures. If the number of modes is not known, then it is possible to select an incorrect statistical model, and have a subset of highly relevant input variables perform poorly according to that model. The chapter discusses two univariate mode estimation algorithms: Silverman's (1981) kernel-based mode estimate, and the DIP test for unimodality (Hartigan and Hartigan, 1985). It proposes extensions to the algorithms for multivariate data. It goes on to test the extensions on several artificial data sets, and shows that the accuracy of mode estimates depends to some degree on how dense the points in the density estimate are. However, accuracy is affected only slightly when kernel density estimates involve kernel widths determined using likelihood cross validation.

1.3 Comparison to Other Methods

In Chapter 7, the proposed method is compared to Moody's Sensitivity Based Pruning and MacKay and Neal's Automatic Relevance Determination. The first set of results show for which prediction problems each of the methods is likely to work. All methods require a certain amount of data to be effective. The proposed method works best when the number of relevant input variables is small, and the number of irrelevant input variables is large. The other two methods work best when the number of irrelevant input variables is small. For a certain class of problems, all methods perform well.

The chapter goes on to show that problems suggested in (Finnoff et al., 1993) are not appropriate for comparing input variable selection algorithms. In general, the number of irrelevant input variables is too small to make a significant difference in performance when they are removed. In addition, two of the problems are too complex for small networks to learn adequately. A third problem is simply an inherently difficult prediction problem.

The chapter finishes by comparing the three input variable selection algorithms on six different real world prediction problems. The problems examined in this thesis turn out to be of three qualitatively different types. One type of problem involves a nonstationary time series, and it is shown that all methods may perform poorly. Another type of problem involves a small number of highly relevant input variables, where the degree of relevance varies drastically across individual inputs. A third type of prediction problem involves only a small number of irrelevant input variables. The proposed method does not perform as well as the two others on this type of problem. The results on all real world problems confirm the conclusions at the start of the chapter, which state for which types of problems each method works well.

1.4 Overview of the Dissertation

The thesis is organized as follows. Chapter 2 outlines the central mathematical ideas that will be used throughout the thesis, which span statistics, information theory, and connectionism. Chapter 3 reviews previous work in input variable selection. Chapter 4 motivates the use of mutual information as a measure for input variable relevance, and shows that it differs fundamentally from error based measures such as sum squared error. Chapter 5 presents the proposed method in detail and provides results demonstrating its strengths and weaknesses on artificial problems. Chapter 6 covers the important problem of deciding how many modes are contained in the conditional output distribution, so that a proper connectionist model can be chosen. Chapter 7 presents detailed comparisons of the proposed method with Sensitivity Based Pruning and Automatic Relevance Determination. The final chapter addresses conclusions from the research and future research directions.

CHAPTER 2

PRELIMINARY DEFINITIONS

This chapter covers terminology used to describe the various methods for input variable selection covered in this thesis. The method I propose is based upon two concepts: nonparametric kernel density estimation, and the information theoretic notion of mutual information. Two topics are reviewed in Sections 2.1 and 2.2. To understand other methods for input variable selection, the reader will also benefit from background in statistical modeling, also termed connectionist learning in this thesis. This is covered in Section 2.3.

The review begins with entropy, a central idea of information theory. It provides a foundation for many information-related concepts, including mutual information. Since mutual information will be shown to be a function of some probability distribution, I introduce nonparametric density estimation next, which is an approach that allows estimation of probability distributions from a set of observations. Lastly, the review covers ideas from connectionist learning, which play centrally in other methods for input variable selection.

2.1 Information Theory

A central goal of information theory is to describe, mathematically, the ability of devices to transmit and receive information. A the heart of the theory is the description of communication signals as probability distributions. Within this paradigm, it is possible to effectively capture ideas such as a transmitter's reliability, its maximum information transmission capacity, and the greatest achievable compression rate of an encoded signal stream. Since the inception of information theory, its basic principles have been applied in many other fields besides communication theory. However, the foundation of the theory upon probability distributions still remains.

The definitions provided here assume discrete random variables, but there are well defined analogues for continuous random variables. Notation for the definitions follows (Cover and Thomas, 1991).

2.1.1 Entropy The entropy $H(p)$ can be viewed simply as a function of some probability distribution $p(X)$, where X is the set of values with nonzero probability according to p:

$$H(p) = -\sum_{x \in X} p(x) \log p(x) \ .$$

Within information theory, X might describe the possible values produced by a signal transmitter, and p describes the relative frequency of occurrence of each value. The function $H(p)$ then represents the maximum compression rate of a signal stream that is achievable using any bitwise encoding for the values of X (assuming that the log in the equation for $H(P)$ is base two).

For the purposes of this thesis, however, it is most helpful to think of $H(p)$ as a measure of surprise, or uncertainty: the greater the value of $H(p)$, the more uncertainty we have about the next value of X that is observed. Suppose that the number of values in X is m. If one value of X occurs with probability one, then there is no uncertainty about the next value drawn from p, and $H(p)$ equals zero. When all m values of X have equal probability, there is maximum uncertainty about the next value of X, and $H(p)$ equals is maximum possible value of $\log m$. The bounds on $H(p)$ of zero and $\log m$ hold only when p is a probability mass function; $H(p)$ is unbounded when p is a continuous, probability density function, but the intuition of "surprise" still holds. The least surprise-producing distribution is the Dirac delta function, which takes on only one value with unit probability; its entropy is negative infinity. The most surprise-producing distribution occurs when all probabilities in the distribution become infinitesimal in value.

2.1.2 Relative Entropy The relative entropy $D(p\|q)$ is a slightly more general formulation of entropy, in that it allows a kind of comparison of entropies between two probability distributions p and q defined over the same domain:

$$D(p\|q) = \sum_{x\in X} p(x)\log\frac{p(x)}{q(x)} = \sum_{x\in X} p(x)\log p(x) - \sum_{x\in X} p(x)\log q(x) \geq 0 \ .$$

In terms of information theory, $D(p\|q)$ represents a bound on the penalty in the compression rate of a signal that must be paid when the real signal distribution is p, and we thought it was q. When p and q are identical, the relative entropy is zero.

Relative entropy has an important property that is relied upon in analysis in Section 4.2: it is always nonnegative, regardless of whether p and q are probability mass functions or density functions. To gain an intuition for this, notice the difference of two summations in the equation. Here, $q(x)$ would have to be consistently larger than $p(x)$ for relative entropy to be obviously negative; this can't happen, however, because the laws of probability dictate that the sum of probabilities for both distributions must be one. In general, then, $q(x)$ is sometimes bigger than $p(x)$ and sometimes smaller, but the concavity of logarithms causes the contributions of smaller $q(x)$ values to outweigh the contributions of the larger values. Hence, the right summation involving $q(x)$ is always less than the left summation, and the relative entropy is always positive. Other properties include the fact that it is not symmetric: in general, $D(p\|q) \neq D(q\|p)$, and it is finite only when the set of support of p is a subset of the set of support of q. Relative entropy has other names, including cross entropy, Kullback-Leibler distance, asymmetric divergence, and discriminability.

For the purposes of this thesis, relative entropy measures the level of surprise in discovering that a distribution is p when we thought it was q. In general, it is small when p and q look similar, and it grows as p and q differ more and more.

2.1.3 Mutual Information Mutual information $I(\cdot;\cdot)$ is a function of the joint distribution between two sets of random variables. For the purposes of this thesis, I call one set with an arbitrary number of variables $\mathbf{X}$ and the other Y; Y consists of just one variable. What $I(\mathbf{X};Y)$ measures is the reduction in uncertainty of output variable Y due to knowledge of the input variables $\mathbf{X}$. The uncertainty of a distribution is made precise using the formula for entropy H. If we let $H(Y)$ be the uncertainty of $p(Y)$ and $H(Y|\mathbf{X})$ be the uncertainty of the distribution $p(Y|\mathbf{X})$, then

$$I(\mathbf{X};Y) = H(Y) - H(Y|\mathbf{X}) = -\sum_{y\in\mathcal{Y}} p(y)\log p(y) + \sum_{\mathbf{x}\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(\mathbf{x},y)\log p(y|\mathbf{x}) \ ,$$

where $\mathcal{X}$ and $\mathcal{Y}$ represent the domains for $\mathbf{X}$ and Y. An analogous definition applies for continuous distributions; integrals merely replace sums.

Another view of mutual information can be seen through relative entropy. With the identity $p(\mathbf{X},Y) = p(Y|\mathbf{X})p(\mathbf{X})$, the expression for mutual information can also be written as

$$I(\mathbf{X};Y) = \sum_{\mathbf{x}\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(\mathbf{x},y)\log\frac{p(\mathbf{x},y)}{p(\mathbf{x})p(y)} = D\left(p(\mathbf{X},Y)\,\|p(\mathbf{X})p(Y)\right) \ .$$

When $\mathbf{X}$ and Y are statistically independent, then $p(\mathbf{X},Y) = p(\mathbf{X})p(Y)$, causing the fraction in the formula to equal one, and thus the value of $I(\mathbf{X};Y)$ is zero. The value of $I(\mathbf{X};Y)$ grows as $\mathbf{X}$ and Y become more dependent. The more dependent Y is on $\mathbf{X}$, the more information one gains about Y once $\mathbf{X}$ is known, and therefore the less uncertain Y is when $\mathbf{X}$ is known.

We can imagine that mutual information is an appropriate measure of the relevance of some set of input variables $\mathbf{X}$ in determining an output variable Y. The large the mutual information, the more dependent $\mathbf{X}$ and Y are, and the more one can determine about Y if $\mathbf{X}$ is known. More justification for mutual information as a relevance measure is given in Chapter 4.

2.2 Nonparametric Density Estimation

From the previous section, it is clear that probability distributions play a central role in information theory and the definition of mutual information. Unfortunately, such distributions are seldom known for practical problems; the most that is usually available is a set of observations drawn from these distributions. Therefore, it is necessary to estimate the distributions from available observations. This is the goal of density estimation.

There is an extensive literature on density estimation. It includes methods for forming discrete distribution estimates using histograms (Scott, 1992), continuous estimates using continuous kernel functions (Silverman, 1986; Härdle, 1989), and even estimates using connectionist networks (Bishop, 1994; Weigend and Srivastava, 1995). Because none of these methods assume that the underlying distribution is describable by a model with a small number of parameters, they are termed nonparametric density estimation methods. The advantage of a nonparametric approach is the ability to let the data "speak for itself"; it has been proven that many nonparametric density estimation methods are capable of approximating large classes of distributions **arbitrarily** well, given enough data (Silverman, 1986).

2.2.1 Kernel Density Estimation One nonparametric method I have explored is based on continuous kernels. The estimates are formed by placing a continuous "bump", or kernel function, over each data point sampled from the distribution (Figure 2.1). A kernel based method works naturally with data that are continuous valued, although it also works well when the data consist of values from both discrete and continuous random variables.

The method explored in this thesis uses a kernel function known as a multi-dimensional Epanechnikov product kernel (Epanechnikov, 1969). If each data point $\mathbf{x}$ has d elements, then the canonical kernel function has the form

$$K(\mathbf{x}) = \begin{cases} \prod_{i=1}^{d} \frac{3}{4}(1 - {x_i}^2) & \text{for } ||\mathbf{x}||_\infty < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

The factor of $\frac{3}{4}$ is there simply as a normalization factor; it causes the kernel function to integrate to one.

One reason for using this kernel instead of other kernels, such as a Gaussian, is computational speed: only nearby data points can contribute to a density estimate, as opposed to having all data points involved for every estimated probability density. By using k-d trees to find neighboring points, it is possible to compute a density estimate in time $O(n \log n)$, where n is the number of data points used in the estimate.

The probability density for some point $\mathbf{x}$, $P(\mathbf{x})$, is estimated as

$$P(\mathbf{x}) = \frac{1}{N}\frac{1}{h^d}\sum_{j=1}^{N} K\left(\frac{1}{h}(\mathbf{x} - \mathbf{x}(j))\right),$$

where N is the size of the data set, $\mathbf{x}(j)$ is element number j in the set, and h is the "window width", or spread, chosen for the kernel functions. This formula guarantees that regardless of the choice for the kernel width h, the integral over the entire density equals one. Note that the expression $1/h$ appears within the kernel function argument, which may seem confusing at first. It could be included as part of the definition of the kernel function itself; it is simply more accepted within the literature to refer to canonical kernel function, which does not include a kernel width term.

2.2.2 Selecting a Kernel Width The best kernel width to use, where "best" implies the width that results in the greatest approximation accuracy of the underlying density function, cannot be determined **a priori**. There are heuristic methods for selecting a good width, however. The following description is adapted from (Silverman, 1986). One way to evaluate the goodness of fit of a density estimate is by examining the estimate's likelihood of generating data. For density estimation, this is possible using a technique known as **likelihood cross validation**: if h is too narrow, the likelihood will drop on future examples because much of the data space will have low associated density. On the other hand, if h is too wide, then the density is spread too uniformly over the data space, and likelihood will also drop for future examples (Figure 2.2).

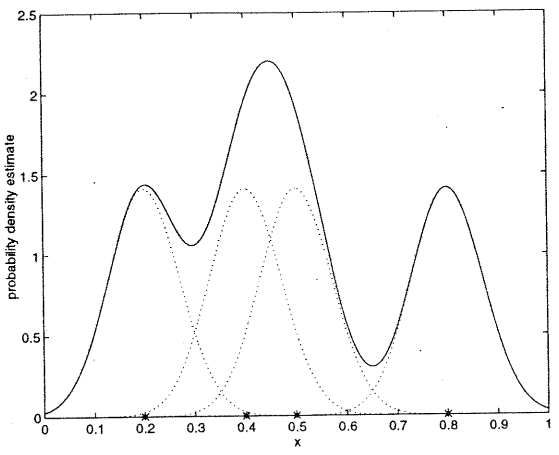


Figure 2.1. Nonparametric density estimation using continuous kernels for a simple one dimensional example. A kernel function is centered over every data point (represented by stars in the graph), and the contributions of all kernels are added together to estimate the probability density at any point (solid line).

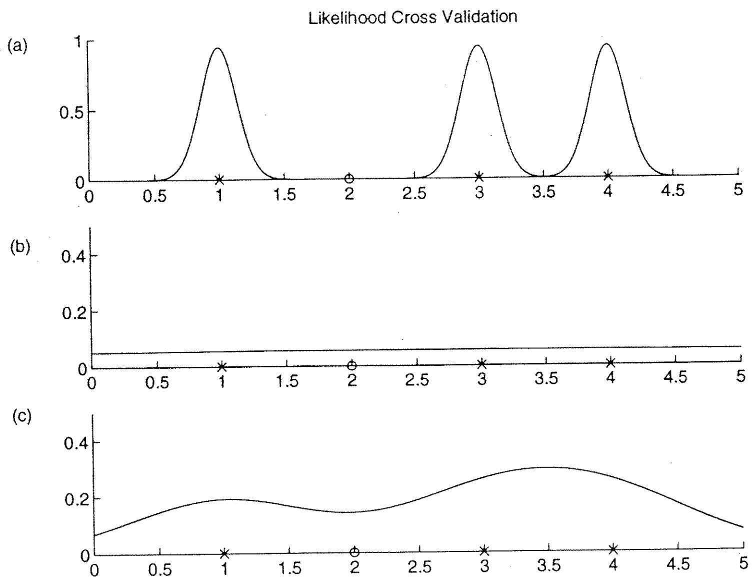


Figure 2.2. Kernel width determination using likelihood cross validation. In the three plots, density estimates are produced from three points located at the values 1, 3, and 4. A withheld point is located at the value 2. Each plot is produced by using a different kernel width. For (a), the kernel width is too narrow; likelihood at the value 2 is very small. For (b), the kernel width is too wide, and likelihood at the value 2 is also small. For an intermediate kernel width value, as in (c), the likelihood at the value 2 is largest. This is the kernel width derived from the likelihood cross validation method.

I describe the method using the leave-one-out cross validation procedure, although the method can be formulated using other cross validation based approaches. Define $\hat{f}_{-i}(\mathbf{X}, Y)$ to be the density estimate created by leaving out example $(\mathbf{x}_i, y_i)$. Then, an estimate for the log likelihood of a density function is given by

$$CV(h) = \frac{1}{n} \sum_{i=1}^{n} \log \hat{f}_{-i}(\mathbf{x}_i, y_i) .$$

One simply searches for the value of h that makes $CV(h)$ the largest. Although this method is reportedly sensitive to outliers, it is one of the few principled approaches for automatically selecting a width h from a finite amount of data (Silverman, 1986). One can compensate for outlier sensitivity by removing outliers from the data in advance.

Having a principled approach for kernel width selection is important, since the computation of mutual information depends directly on the kernel width. If the bins or kernels for a density estimate are too narrow, then the density estimate will appear to contain a great deal of structure, and mutual information with the output variable can be overestimated. If the bins or kernels are too broad, then the opposite happens: crucial structure in the density can be lost, and the input variables will appear to have little mutual information with the output variable.

2.3 Connectionist Terminology

Because of its interdisciplinary nature, connectionism has borrowed terminology from a number of fields, including statistics, computer science, mathematics, and psychology. The main components of a connectionist network are illustrated in Figure 2.3. There is a set of **input variables x**, often termed independent variables or regressors in statistics. These are connected to a collection of nonlinear **hidden units h** through a matrix of adjustable **weights** $\mathbf{W}_1$, which are known in statistics as regression coefficients. A common function for hidden, or intermediate processing units is

$$h_i = \tanh\left(\sum_j \mathbf{W}_{1(ij)} x_j\right) .$$

The network **output** $\hat{y}$ is a function of the hidden unit function values. A common output function is

$$\hat{y} = \sum_j \mathbf{W}_{2(j)} h_j ,$$

which involves a vector $\mathbf{W}_2$ of adjustable weights or regression coefficients.

Learning is the process of finding weight values that maximize some measure of the network's ability to produce good output values. A set of input-output examples is provided for learning; this is termed a **training set**. Often, an important goal in learning is good **generalization** performance of a network; this represents the performance of a trained network on novel examples. Statisticians refer to generalization as out-of-sample performance. To evaluate generalization, there is often a **test set** of examples; the training set and test sets are disjoint. Sometimes, examples from the training set are set aside and used as a surrogate test set; this is similar to the statistical notion of **cross validation**. Such a collection of examples is called a **validation set**.

Learning in a connectionist network is also termed nonlinear regression by statisticians. There are many important concepts from the statistics literature that have been borrowed by connectionists to better understand the learning process. Perhaps the central concept from statistics is that learning in a network involves finding a **maximum likelihood** solution for the given data, which is done by selecting weights so that the likelihood that the network generated the training set is maximized. This is the way in which "producing good output values" is made mathematically precise. Within this framework, a likelihood function L becomes

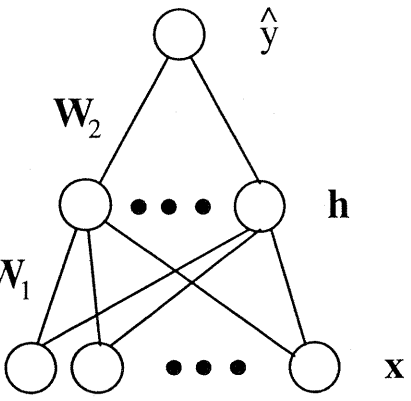


Figure 2.3. Components of a connectionist network. Processing in the network flows from bottom to top through the edges of the graph.

the measure of the quality of a particular set of weight values. This function maps a set of examples and a network with specific weights to a real number, which represents the probability that the network generated the training set. An example of a likelihood function, representing the "Gaussian output noise" model, is

$$L\left(\{\mathbf{x}_i, y_i\}, \mathbf{w}\right) = \left(2\pi\sigma^2\right)^{-\frac{n}{2}} \prod_{i=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{y_i - \hat{y}\left(\mathbf{x}_i; \mathbf{w}\right)}{\sigma}\right)^2\right) ;$$

refer to Figure 2.4. Once we have a likelihood function, it is possible to compute the **gradient** of the function with respect to the network weights. More commonly, the gradient for the **negative log likelihood** function $-\log L$ is computed. The negative log likelihood for the above function is

$$\begin{aligned} -\log L\left(\{\mathbf{x}_i, y_i\}, \mathbf{w}\right) &= \left(\frac{n}{2}\log 2\pi\sigma^2\right) + \sum_{i=1}^{n} -\frac{1}{2}\left(\frac{y_i - \hat{y}\left(\mathbf{x}_i; \mathbf{w}\right)}{\sigma}\right)^2 \\ &\propto K + \sum_{i=1}^{n}\left(y_i - \hat{y}\left(\mathbf{x}_i; \mathbf{w}\right)\right)^2 , \end{aligned}$$

where K is an unspecified constant, if one considers σ and n to be fixed during the search for $\mathbf{w}$. By following the negative gradient of the above function, it is possible to find the weight values that maximize the likelihood. A good source for more information on connectionist ideas is (Hertz et al., 1991).

Another common definition is the **Hessian**, which is the matrix of second derivatives of the negative log likelihood function. For weight values that minimize the log likelihood function, the Hessian can be used to describe the curvature of the error surface. This is a common way of finding out how well determined the weights in a network are; that is, one can find out how much the minimized weight values would vary over possible training sets. For weights where the elements of the Hessian are large, idiosyncrasies in the training set are not likely to have had a large effect on the weight's value, and the weights can be said to be well determined. However, if the values of the Hessian are small, then some weights in the network may not be well determined, which makes the network's performance on novel examples less certain.

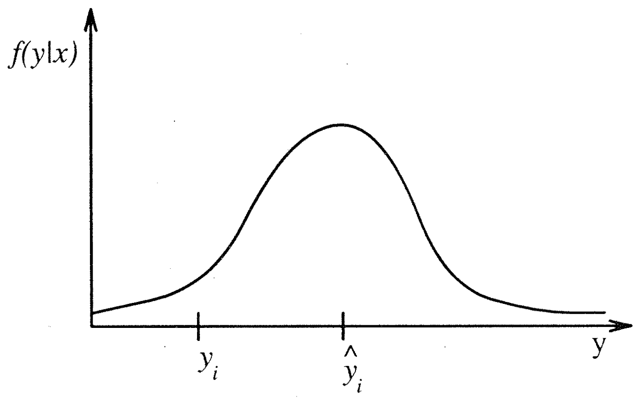


Figure 2.4. An interpretation of likelihood maximization. The likelihood of generating y_i is given by $f(y_i|x)$. If $\hat{y}_i$, representing the mean of the distribution, is moved closer to y_i, then the likelihood of generating y_i increases. The likelihood is maximized when $\hat{y}_i = y_i$. Hence, when y_i is an observed value for an output variable and $\hat{y}_i$ is the value produced by a network, maximizing the likelihood function causes the network to try bringing $\hat{y}_i$ as close as it can to y_i.

CHAPTER 3

PRIOR RESEARCH IN CONNECTIONIST INPUT VARIABLE SELECTION

The goal of this chapter is to present the ideas and results of previous work in input variable selection. It is not meant to be an exhaustive account of all methods that have been proposed; it covers only those methods that have become accepted into the connectionist literature and have addressed the issue of improving generalization performance as a result of the method's application.

The methods reviewed here can be grouped into two categories: those that perform input variable selection and network training as a single, combined step, and those that perform input variable selection as a step prior to, and separate from, training. The first category, by definition, involves a regression model in the selection of input variables; for this reason, the methods are labeled **model based**. Although the second category can also involve regression models in both steps, the ones reviewed here do not use regression models in selecting input variables. Therefore, methods from the second category are labeled **model free**. The review begins with model based methods. Because some methods require a form of combinatorial search algorithm for selecting input variables, the chapter ends with a discussion of several commonly used combinatorial search algorithms and their known properties.

3.1 Model Based Approaches

The model based approaches reviewed here can be viewed as generalized versions of traditional methods in linear regression. The ARD method (Section 3.1.1) can be seen as a generalization of ridge regression to statistical models with intermediate, nonlinear processing components. The SBP algorithm (Section 3.1.3) can be viewed as a modified version of backward elimination (Beale, 1970). The NS$_p$ model is an extension of linear regression using a weight regularizer such as Thompson's S$_p$ statistic (Thompson, 1978).

3.1.1 Automatic Relevance Determination Derivation of Automatic Relevance Determination (ARD) lies squarely within the realm of Bayesian statistics. A mean zero, Gaussian prior is placed over each set of network weights leading from an input variable to intermediate, nonlinear processing components. For an input variable c, the standard deviation α_c of each Gaussian is left as a hyperparameter to be determined in part from the data, rather than being determined before training begins. This set of assumptions leads to a negative log likelihood function that is slightly different from the one mentioned in Section 2.3:

$$M(\mathbf{w}) = \beta E_D + \sum_{c=1}^{m} \alpha_c E_{W(c)} \, . \tag{3.1}$$

The first term measures the quality of fit to the data

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \, ,$$

and the second term is a sum over the number of input variables m, which embodies a penalty for large weights from the input unit layer to the hidden unit layer. The α_c embody the separate Gaussian priors over each input variable's weights, and the term $E_{W(c)}$ is simply equal to a weight decay regularizer:

$$E_{W(c)} = \frac{1}{2} \sum_{j=1}^{h} w_{cj}^2 \, ,$$

where h is the number of hidden units in the network. For notational convenience, the vector composed of all α_c is denoted $\boldsymbol{\alpha}$.

The values for $\boldsymbol{\alpha}$ and β control the balance between the two terms in (3.1), and setting them to proper values can improve generalization performance on future examples.[1] Certain additional assumptions about the shape of $M(\mathbf{w})$ allow the derivation of algebraic formulae for $\boldsymbol{\alpha}$ and β; this is the approach taken in (MacKay, 1993). However, the additional assumptions necessary for deriving the equations are rather strong for the neural network paradigm; they include assuming that the error surface is quadratic and that estimates of the Hessian during training accurately reflect the Hessian near the set of weights at the global error minimum.

Training an ARD network within MacKay's framework involves initializing network weights and the values of $\boldsymbol{\alpha}$ and β, updating the network weights every epoch as in normal network training, and updating the values $\boldsymbol{\alpha}$ and β as desired, perhaps less often than once every epoch. However, there is a more general, Bayesian approach to network training within the ARD framework (Neal, 1995). Neal's version of ARD is the implementation used in my thesis.

3.1.2 Neal's Implementation of ARD Neal (1995) takes a more general approach to network training and setting the $\boldsymbol{\alpha}$ and β, which appears more in line with the traditional Bayesian learning paradigm. Whereas many connectionists are satisfied with finding a single set of weights that minimizes an equation such as (3.1),[2] Bayesians are interested in finding the entire posterior distribution, and they may base predictions on an expectation taken over the posterior distribution. Formally, if we write the posterior as

$$P(\mathbf{w}\,|D,\boldsymbol{\alpha},\beta) = \frac{P(D\,|\mathbf{w},\boldsymbol{\alpha},\beta)\,P(\mathbf{w}\,|\boldsymbol{\alpha},\beta)}{P(D\,|\boldsymbol{\alpha},\beta)} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}},$$

where $\mathbf{w}$ represents the network weights and D represents the available data for training, then a Bayesian may base a future prediction $\hat{y}(\mathbf{x};\cdot)$ on the expectation

$$\hat{y}(\mathbf{x};\cdot) = \int \hat{y}(\mathbf{x};\mathbf{w})\,P(\mathbf{w}\,|D,\boldsymbol{\alpha},\beta)\,d\mathbf{w}\,.$$

Rather than compute the integral analytically, which is usually intractable in practice, it is approximated using Monte Carlo techniques:

$$\hat{y}(\mathbf{x};\cdot) = \frac{1}{n}\sum_{i=1}^{n}\hat{y}(\mathbf{x};\mathbf{w}_i)\,.$$

Although a generate-and-test approach could be used to sample points in weight space for the approximation, most points would have close to zero data likelihood, making the posterior probability small also, and the approximation would be poor. Neal instead uses a stochastic search algorithm that combines Gibbs sampling and the Metropolis algorithm for finding regions of weight space with high posterior probability. Unlike MacKay, this approach requires no assumptions about the shape of the error surface or the approximation accuracy of the Hessian. It can also combine the results of qualitatively different network solutions in a principled manner, and theoretically it has the ability to search weight space without becoming caught in local minima.[3]

The $\boldsymbol{\alpha}$ and β hyperparameters are updated periodically in a way similar to the network weights. Instead of assuming that these hyperparameters are fixed, as the in the previous equation for the posterior, we can write the posterior as

$$\hat{y}(\mathbf{x};\cdot) = \int \hat{y}(\mathbf{x};\mathbf{w})\,P(\mathbf{w},\boldsymbol{\alpha},\beta\,|D)\,d\mathbf{w}\,d\boldsymbol{\alpha}\,d\beta\,.$$

[1] From a practical viewpoint, one does not need to modify β because it is simply a scaling factor on the overall error. However, modifying it preserves a probablistic interpretation of the error.

[2] This is equivalent to finding the maximum a posteriori (MAP) estimate.

[3] In practice, however, the stochastic search algorithm may take a long time to find good areas of weight space if these areas are disconnected (Neal, personal communication).

and find a Monte Carlo estimate as before. Rather than treating the hyperparameters exactly as network weights, Neal fixes the network weights while applying Gibbs sampling to the hyperparameters. Briefly, Gibbs sampling cycles through the components of $\boldsymbol{\alpha}$ and β; for each component, all others are fixed while a new value for the free component is chosen. Because there is usually no gradient information available for the hyperparameters, the choice for a new value is based on a scheme known as **rejection sampling**. Candidate values are generated from a prior distribution over the hyperparameters (one distribution for β and one for $\boldsymbol{\alpha}$). In Neal's implementation, this is a Gamma distribution with shape and width parameters chosen at the start of learning.[4] A candidate value is accepted with probability proportional to the data likelihood for that choice: the combination of generation using priors and acceptance using likelihood results in values with high posterior probability.

In addition, Neal includes sophisticated, mean zero priors over the network's hidden-to-output unit weights. The goal is to force the weights to zero, unless an opposing pressure from the training algorithm exists, to help control network complexity during training without requiring a validation set. This allows more data to be used directly for training, which often results in improved generalization performance. Neal uses asymptotic analysis of hidden units to choose the level of pressure in a principled fashion.

Empirical results for ARD appear in (MacKay, 1993; Allison, 1994; Thodberg, 1994; Neal, 1995). For several standard machine learning benchmark problems, it is shown that ARD performs better than networks where all potential input variables are included. In (Neal, 1995), it is shown that ARD does as well as an approach by Quinlan (1993) on the standard Boston housing data set. Neal does not compare performance against other input variable selection methods. One goal of this thesis is to make wider, more systematic comparisons of ARD against competing input selection algorithms.

3.1.3 Sensitivity Based Pruning The Sensitivity Based Pruning (SBP) algorithm is presented in (Moody and Utans, 1992; Moody, 1994; Utans et al., 1995). The algorithm begins with a network that has been trained using all potential input variables. A sensitivity measure S_c is computed for each input variable x_c to evaluate the change in training error that would occur if it were removed from the network. The definition for S_c is

$$S_c = \frac{1}{n}\sum_{i=1}^{n}\left[SE\left(\bar{x}_c\right) - SE\left(x_{ci}\right)\right] \qquad \text{with} \qquad \bar{x}_c = \frac{1}{n}\sum_{i=1}^{n} x_{ci}\ .$$

The value S_c measures the average effect on the training squared error (SE) of replacing input unit x_c with its average value for all examples. The input variable with the smallest sensitivity is removed, and the network is retrained with one less input variable; weights from the previously trained network are used as a starting point to avoid complications that could arise from the presence of multiple local minima. This process of training, computing sensitivities, and deleting is repeated; cross validation based techniques are used to decide when this cycle should be stopped.[5] SBP can be viewed as an extension of the backward elimination algorithm from linear regression (Beale, 1970). The step of computing sensitivities in SBP provides a shortcut: whereas, in backward elimination, a separate network would be trained for each input variable's removal, SBP trains only one network and uses the measure to select an input for deletion.

Results for SBP are presented for a real-world problem where the task is to predict corporate bond ratings from a small data set of less than two hundred examples. The results show that pruning two of ten input variables improves network performance on a test set, although there are no controls to check whether the greatest possible improvement in performance is obtained over all possible input variable subsets. There are also no comparisons to other methods. In (Utans et al., 1995), results are presented for predicting the Index of Industrial production. The data consist of 480 monthly observations obtained from ten financial time series.

[4] An explicit goal in Neal's implementation is to make learning performance insensitive to choices for the Gamma distribution parameters. He claims that this is true for the problems studied in his thesis, and all runs used the same choice. I follow his lead by using the same choice for all of my simulations also.

[5] In this thesis, cross validation for SBP is based on ten random splits of the data into training and validation sets.

Both raw and "filtered" versions of the time series are used to create 48 possible input variables. The SBP algorithm is able to delete 35 variables for an improvement in generalization ability over a network with all input variables included. Once again, no comparison to other input variable selection algorithms is performed.

3.1.4 A Linearized Network (NS$_p$) With this method, input variables are not selected as a part of training, but instead through the comparison of different trained network architectures, where certain input variables are selected for each architecture. Yet another approach for controlling network complexity, besides Bayesian assumptions or validation sets, involves minimizing the **expected error** on future examples: the **Mean Squared Error of Prediction** (MSEP).[6] In order to apply this method, however, certain linearizing assumptions about neural networks must be made first.

Discussion of linearized nonlinear networks can be found in (Seber and Wild, 1989); I use ideas from this reference to extend the analysis in (Thompson, 1978) for minimizing an error function known in the linear regression literature as the S_p statistic.[7]

As always, the notation uses capital letters (Y) for random variables and lower case letters (y) for individual instances drawn from the random variables. For this discussion, it is assumed that the random variables Y and $\hat{Y}$ are determined by an input variable X and a set of network weights $\mathbf{w}$. Sometimes, $\hat{y}(x;\mathbf{w})$ is written explicitly to remind the reader of this relationship.

MSEP is defined as the expected error on a future output value $y(x;\mathbf{w})$. Assume we have an unbiased estimate of Y, denoted $\hat{Y}$. Based on the identity $E\left[Y^2\right] = E\left[Y\right]^2 + \mathrm{Var}(Y)$, we can write:

$$\begin{aligned} \mathrm{MSEP}\left(\hat{Y}\right) &= E\left[\left(Y-\hat{Y}\right)^2\right] \\ &= E\left[\left(Y-\hat{Y}\right)\right]^2 + \mathrm{Var}\left(Y-\hat{Y}\right) \\ &= 0 + \left[\sigma_Y^2 + \mathrm{Var}\left(\hat{Y}\right)\right] . \end{aligned} \tag{3.2}$$

The rightmost bracketed term of equation (3.2) shows two distinct sources contributing to the variance of $Y - \hat{Y}$: the variance of the intrinsic output noise σ_Y^2, and the sampling error of the estimated output value, due to the finite amount of available training information. Hence, a network that minimizes expected error must trade off ability to fit the data against a criterion that measures how well network output values are determined by the data. Networks with too few input variables will rate badly because of poor fit to the data (high σ_Y^2), and networks with too many input variables will also rate badly because the network weights are not determined well enough, and an estimated output value $\hat{y}(x;\mathbf{w})$ will have a high variance (high $\mathrm{Var}\left(\hat{Y}\right)$).

Now for analysis showing how to obtain estimates for these two sources of error. Both are variances, and therefore can be combined. The value for σ_Y^2, the intrinsic output noise, is estimated straightforwardly using sum squared error (SSE). Common estimates for σ_Y^2 are either $\frac{\mathrm{SSE}}{n}$ or $\frac{\mathrm{SSE}}{n-m}$, where n is the number of examples, and m is the number of network weights.

The variance for $\hat{y}(x;\mathbf{w})$ is more difficult to estimate; it represents how much a different, independently drawn training set would change the form of $\hat{Y}$. One could imagine an estimating the variance by producing, say, a hundred independent training sets and approximating the sample variance of the $\hat{y}(x;\mathbf{w})$ across all training sets. This would work, except that one rarely has access to one hundred independent training sets. In order to estimate the variance from a single training set, additional assumptions are necessary. The first assumption relates the variance of the learned network weights $\mathbf{w}$ to the variance of $\hat{Y}$. The second assumption results in a simple formula for the variance of network weights.

[6] Other, similar objective functions can be minimized, such as the Akaike Information Criterion (AIC) or Bayes Information Criterion (BIC).

[7] The S_p statistic, unlike the analogous C_p statistic of Mallows, assumes that the input variables are random variables, rather than fixed. In both cases, the objective function to be minimized is the MSEP.

The first assumption states:[8]

$$\hat{y}(x_i;\mathbf{w}) = \hat{y}(x_i;\mathbf{w}^*) + \mathbf{a}_i'(\mathbf{w}-\mathbf{w}^*) \ ,$$

where $\mathbf{w}^*$ is a global minimum in weight space, and $\mathbf{a}_i$ is the error gradient: $\mathbf{a}_i' = \left(\frac{\partial y_i}{\partial w_1}, \ldots, \frac{\partial y_i}{\partial w_m}\right)$; m represents the number of weights in the network. In other words, small changes to the network weights cause only a linear change in the value of $\hat{y}(x_i;\mathbf{w})$. This is at least approximately true, even for nonlinear networks, provided that $\mathbf{w}$ is close enough to $\mathbf{w}^*$. The result of this assumption is that one can write

$$\mathrm{Var}\left(\hat{y}(x_i;\mathbf{w}^*)\right) = \mathbf{a}_i'\mathrm{Cov}(\mathbf{w}^*)\,\mathbf{a}_i \ .$$

The second assumption involves the shape of the error surface near a local minimum. For a weight vector $\mathbf{w}$ close to a minimum $\mathbf{w}^*$ of the sum-squared error, assume that the error surface is well-approximated by a quadratic "bowl". This allows an estimate of the uncertainty of the weights to be based solely on the inverse of the Hessian:

$$\mathrm{Cov}(\mathbf{w}^*) = \left[\mathbf{H}(\mathbf{w}^*)\right]^{-1} \ .$$

Taking into account that examples in the training set are drawn independently, the following expression represents the variance of $\hat{Y}$:

$$\mathrm{Var}\left(\hat{Y}\right) = \sum_{i=1}^{n} \mathrm{Var}\left(\hat{y}(x_i;\mathbf{w}^*)\right) = \sum_{i=1}^{n} \mathbf{a}_i'\left[\mathbf{H}(\mathbf{w}^*)\right]^{-1}\mathbf{a}_i \ .$$

Simple substitution into equation (3.2) results in:

$$\mathrm{MSEP}\left(\hat{Y}\right) \approx \frac{\mathrm{SSE}}{n} + \sum_{i=1}^{n} \mathbf{a}_i'\left[\mathbf{H}(\mathbf{w}^*)\right]^{-1}\mathbf{a}_i \ .$$

There are a few potential drawbacks to this approach. The first is that in order for the analysis to hold, the predictions of the network must be approximately unbiased. In other words, the neural network must be able to approximate the "true" underlying function that supposedly generated the data arbitrarily well. One could at least try to alleviate this drawback by using a lot of hidden units and performing backward elimination search over input variables. The only apparent problem is that when one start with all possible input variables, the number of network weights will be large. As a result, $\mathrm{Var}\left(Y-\hat{Y}\right)$ will be large for all candidate networks with some input variable removed. This may cause the elimination of an input variable that otherwise would be deemed relevant if the number of network weights were smaller. This is the same drawback to SBP and ARD, as it will be shown in Chapter 7.

Another potential drawback is that the two simplifying assumptions must hold at least approximately. The first assumption is not too strict, provided one uses a weight optimization algorithm that gets close to a local minimum. The second assumption may be more strict: the quadratic error surface assumption can be violated by strongly asymmetric error surfaces, which can occur in nonlinear networks.

Currently, there are no tests of this method's performance against others. I simply present it as one possible approach that extends naturally from the extensive treatment of input variable selection in the linear regression literature. Tests of this method are left to possible future research.

[8] A prime ($'$) indicates vector transposition in the following discussion.

3.2 Model Free Approaches

Another line of research has examined input variable selection as a separate step from network architecture selection and training. The underlying assumption to these approaches is that it is important to find input variables with some predictive information first, and to worry about how to extract this information only after input variables are chosen.

The two approaches reviewed here use nonparametric density estimation to express a relationship between input and output variables from a data set. Also, they use mutual information as a measure of the predictiveness of input variables. After reviewing the methods, I discuss critical differences from the method explored in this thesis.

3.2.1 Fraser's Histogram Algorithm and Battiti's Approach Fraser (1988) has proposed a method for estimating the joint density $p(X_1, X_2)$ for two potential input variables in continuous time series analysis, in order to find the first two time lags that are independent; the algorithm is also applied in (Battiti, 1994) for performing input variable selection for connectionist networks. Figure 3.1 provides high level pseudocode for the algorithm. It is a recursive algorithm that creates more histogram splits in areas where the points have finer structure, and it creates fewer splits where there is no evidence for structure.

The algorithm, as it stands, suffers from a few problems. The most serious problem is that because it creates bins, a great deal of data is required for density estimates involving more than two dimensions. Fraser himself warns against performing mutual information estimates for more than two variables unless one has "millions" of data points. The problem is exacerbated by the fact that splitting occurs over the **marginal** distributions of each variable, rather than over existing bins within the joint distribution. As a result, there is a limited sense in which the algorithm can "zoom in" on a region of the data space; many extraneous bins can be created in the process, which hurts estimation accuracy.

There are several other potential problems. First, there is an unjustified, 20% confidence level for testing the presence of structure in a bin split. Also, the greedy nature of the algorithm causes it to terminate early sometimes, even when there is still structure present in the data. The algorithm determines whether splits should occur by counting the number of points in a set of prospective bins. If the numbers of points are roughly equal across the bins, then the split is not performed. Although this approach of looking for unequal bin counts is a reasonable way to test for structure in the data, it contains no information about the distribution of points through further splitting. Figure 3.2 shows that it is possible to have prospective bins with equal counts, while subsequent splits within each prospective bin would have highly unequal bin counts.

Independently from my own work, Battiti (1994) has proposed selecting inputs using nonparametric density estimate and mutual information. His approach differs from mine in two critical ways. First, he relies on either straightforward histogramming or Fraser's binning algorithm for density estimation. This effectively prevents the estimation of relevance for more than one input variable at a time. Second, he relies on a heuristic approximation of relevance for collections of input variables.

To understand the heuristic, assume that some set of input variables has been selected already; at the start, this set is empty. Suppose that the number of selected variables is currently m. Then, from the unselected variables, the input X is selected that maximizes

$$I(X;Y) - \beta \sum_{j=1}^{m} I(X;X_j) .$$

The parameter β is intended to quantify the importance of pairwise dependencies between input variables relative to the dependency of the candidate input variable and the output. The apparent aim of this measure is to discourage the selection of redundant input variables. Battiti provides no principled choice for the value of β, which can be considered an additional shortcoming of the method.

The fallback heuristic is unavoidable given the choice of density estimate technique, which effectively

```
SPLIT(bin)
    for $i \leftarrow 1$ to Num-Dimensions(bin)
        newbins[i] ← a split of bin[i] into two bins with an
                equal number of points;
    return(newbins);

FLAT(bin)
    testbins ← SPLIT(bin);
    if CHI-SQ-REJECT(testbins) then
        return(True);
    else
        return(False);

RECURSE-SPLIT(bin)
    if FLAT(bin) then
        return(bin);
    else
        return(RECURSE-SPLIT(SPLIT(bin)));

MAIN
    bin ← all data points;
    return(RECURSE-SPLIT(bin));
```

Figure 3.1. Fraser's histogram algorithm. If there are s input variables and t output variables, then every bin has $(s+t)$ dimensions. The function CHI-SQ-REJECT is not provided explicitly; it returns **True** if the given set of bins have roughly an equal number of points in them; more precisely, it returns **True** if the null hypothesis that the bins form a flat multinomial distribution is rejected with a confidence level of 20%.

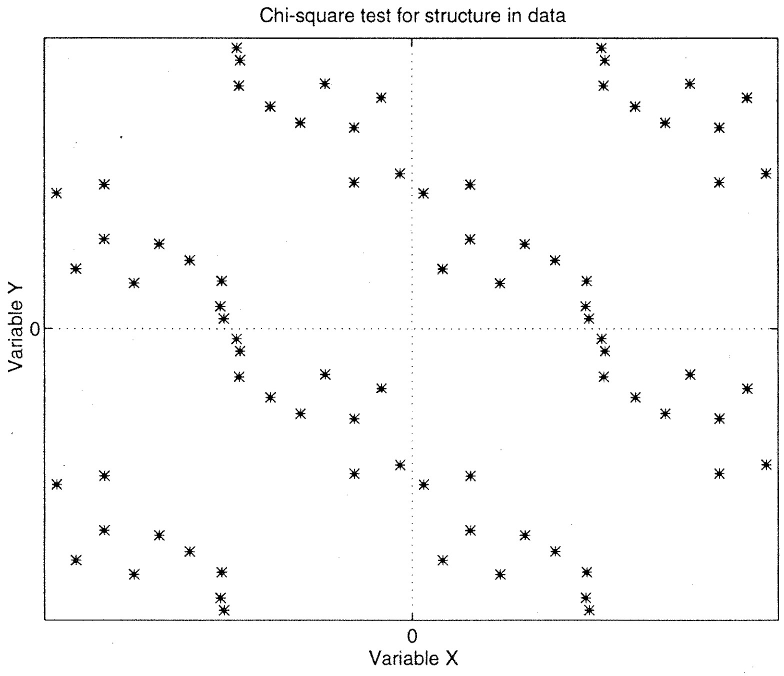


Figure 3.2. A weakness in Fraser's χ^2 test for bin splitting. The proposed split along the dotted lines would not be performed because the number of points in each resulting bin is equal. If the split were to be performed, however, a χ^2 test on splitting any of the four resulting bins would indicate that further splitting is needed. Hence, the stopping criterion terminates early, prevents the algorithm from finding the structure present in smaller bins, and the algorithm underestimates the mutual information.

limits the number of variables involved in a mutual information calculation to two. It does, however, make selection of truly informative input variables impossible if there are strong nonlinearities in the data. Consider, for example, the function $Y = \text{sign}\,(X_1 \cdot X_2)$, where the inputs X_1 and X_2 are i.i.d. uniform in the range $[-1, 1]$. Individually, each input is completely uninformative. Battiti's heuristic does nothing to indicate that the pair of input variables is informative, when in fact their combined information is very high.

On all four data sets he explored, he was unable to show that networks trained with his input variables generalized better than networks with all possible input variables included. Part of the reason for this negative result may be due to the data sets themselves: Battiti mentions that a network trained with all input variables has trouble "overlearning", or overfitting, the data. For such data sets, it is possible that including all possible input variables does not hurt generalization performance greatly. The data sets considered in this thesis are not generally of this type; unlike Battiti, I carefully choose data sets in which it can be shown that eliminating input variables actually makes a difference in generalization performance.

3.2.2 Overcoming Shortfalls of Fraser and Battiti I extend the work of Fraser and Battiti in four general ways. First, I apply a different density estimation technique based on continuous kernels, which avoids accuracy problems that histograms face in estimating high dimensional density functions. Also, because Battiti does not estimate densities of different sizes, he does not have to deal with the issue of comparing mutual information estimates across different input variable subset sizes; this is an important issue in my research. Third, I discuss methods for selecting a network architecture once a set of input variables is chosen. Lastly, I provide extensive comparisons to other methods to show under what conditions the approach works best. Details of the method appear in Chapters 4 and 5, and comparative results appear in Chapter 7.

3.3 Searching for the Best Subset of Input Variables

Several of the methods for input variable selection reviewed so far require a procedure for searching among input variables. These include the NS_p algorithm and Battiti's approach; the method proposed in this thesis requires a search procedure also. If there are k candidate input variables, the potential requirement is that all $(2^k - 1)$ subsets of the k variables must be examined, in order to find the best subset. Even for simple input variable evaluation methods, where evaluation takes roughly a second, the largest number of input variables that can be considered using exhaustive search is between 20 and 30.

It is common to try local search and other heuristic search techniques for finding the best subset of input variables. The remainder of the chapter reviews several of the most common procedures. First, a framework for understanding search algorithms is introduced, which helps clarify tradeoffs among different procedures. The framework shows that some procedures are more computationally demanding than others but are also more likely to find high quality subsets of input variables.

3.3.1 A Framework for Combinatorial Search Procedures for searching among input variables can be viewed as combinatorial search algorithms, since the size of the search space explodes combinatorially as a function of the number of possible input variables. Any procedure can be cast so that it includes the following features:[9]

- A finite set of possible input variable subsets S.
- An objective function $f : S \to \mathbf{R}$, which maps subsets to real numbers. The real numbers represent the "quality" of a subset.
- A **neighborhood structure** N, which defines for each subset $i \in S$ a collection of solutions which are in some sense "close" to i.
- A simple, local search algorithm, which starts with some prespecified subset and then continually tries to find better subsets by searching neighborhoods. The algorithm terminates when the qualities of all subsets in a neighborhood are no higher than the current subset.

[9]The framework is adapted from (Aarts et al., 1996).

Neighborhood Sizes			
Algorithm	Largest	Smallest	Average
Forward Selection	k	1	$k/2$
Backward Elimination	k	1	$k/2$
Stepwise Selection	k	k	k
Sequential Selection	$k^2/4$	k	$k <$ size $< k^2/4$

Table 3.1. Neighborhood sizes for different local search algorithms. The number of available input variables is k.

This framework is mentioned to make one important point: there is a tradeoff between search time and solution quality, which is realized in the choice of neighborhood structure. For some search algorithms reviewed here, average neighborhood structure sizes are on the order of k, the maximum possible number of input variables (Table 3.3.1).

3.3.2 Forward Selection and Backward Elimination In forward selection, the starting point of the search is the solution involving no input variables, or an input with a constant value, if such an input is considered. A member of the neighborhood of some subset of input variables is obtained by adding some unselected input variable to the current subset. The local search algorithm examines the entire possible neighborhood for the current subset. Therefore, input variables are added in a greedy fashion: the variable whose addition causes the greatest increase in subset quality is added. Variables are never deleted.

In backward elimination, the starting point is the solution with all possible input variables selected. For this algorithm, a member of the neighborhood of a subset of input variables is obtained by deleting some member of the current subset. The search behavior of this algorithm is the opposite of forward selection; variables are deleted in a greedy fashion. Variables are never added.

For both algorithms, if there are k possible input variables, the neighborhood structure size starts at k and decreases by one at every optimization step. Therefore, the maximum number of input variable subsets that are examined by either algorithm is $k(k+1)/2$; being $O(k^2)$, this is a tractable value even when k is in the hundreds or thousands. It should be noted that there is no requirement on either algorithm that this many subsets must be explored; it is an upper bound. Either algorithm may stop because the addition of any subset in the current neighborhood does not improve overall quality.

3.3.3 Stepwise Selection This algorithm, in its original form, was presented in (Efroymson, 1960) for linear networks. The search optimized a particular objective function based on a statistical test of significance for linear network input variables. A more general form of the algorithm can be stated, however, for any objective function.

The starting point for the algorithm is the same as for forward selection: no input variables are selected, or perhaps one variable with a constant value is included. Each optimization step adds an input variable in a greedy fashion, as with forward selection, but then the algorithm also determines, after adding a variable, if solution quality can go up further yet by deleting any currently selected input variable. Therefore, at any point in the optimization, the neighborhood structure size is k. Note that the neighborhood size for the stepwise selection algorithm is larger, on average, than the average neighborhood sizes for forward selection or backward elimination. Therefore, one can expect the stepwise algorithm to run longer but find better solutions. In general, it is not possible to determine how many optimization steps will be performed before the algorithm converges, although it is possible to prove convergence if the objective function being minimized has a lower bound.

3.3.4 Other Possible Local Search Algorithms Many other search algorithms besides the above ones are possible; modifiable parameters include starting location and neighborhood structure. For example, one could modify the stepwise selection algorithm by making the starting location random; this would enable us to obtain different solutions for different algorithm runs, and then the best solution from a set of runs could be taken.

For stepwise selection, another possible neighborhood structure is obtained by considering the **replacement** of any selected input variable with any unselected variable. Miller (1990) refers to this class of algorithms as **sequential replacement** algorithms. If the number of currently selected variables is s, then the neighborhood size is $s\,(k-s)$, which is somewhere between k and $k^2/4$, depending on the value of s. Neighborhood structure size can be made larger still by considering replacement of pairs of variables, rather than single variables. With such neighborhood structures, one can reliably expect longer running times than stepwise selection, but better solutions. Supporting evidence for this belief, where input variable selection is performed for linear networks, is given in (Miller, 1990).

3.3.5 Branch and Bound Search Algorithms For some problems, the number of input variables may be almost small enough to make exhaustive search tractable, or the described local search algorithms may not yield acceptable solutions. In these cases, it is sometimes possible to take knowledge of the search problem into account to prune the search space.

The most commonly assumed knowledge relates to sum-squared error objective functions (Furnival and Wilson, 1974). For sum-squared error, one can show that eliminating input variables can only cause the error to worsen; hence, it is possible to determine a bound on maximum possible improvement in the objective function when input variables are eliminated. This bound can be compared against the best solutions found so far to determine if a branch in the search tree can be pruned. One example of a pruning method is outlined below.

To perform the search, the form of the search tree and the search method must be specified first. For example, let the root of the search tree be the solution with all k input variables selected; let a node at depth d in the tree represent a certain subset of $(k-d)$ input variables. Assume a depth-first search procedure. If, for example, the C_p statistic for linear networks is being optimized, then one stores the lowest value of C_p for each input variable subset size (Mallows, 1973). At any node in the search tree, one can compute C_p for this node, plus the bounds on C_p for all nodes below the current one. If the bounds are all larger than the smallest values of C_p already found, the entire search tree branch can be pruned.

The potential amount of pruning to the search tree can be quite large; a lower bound on the number of nodes to consider is $2k$. However, in practice, the savings need not be this large. The actual amount of pruning depends strongly on the specific search problem and the order in which nodes in the search tree are traversed.

Although the properties of sum-squared error have been exploited the most in branch-and-bound techniques, other objective functions share similar properties. For example, it can be shown that mutual information between a set of input variables and an output variable can only go up as more inputs are added:

$$I\left(Y;X_1,X_2\right)=I\left(Y;X_1\right)+I\left(Y;X_2|X_1\right)\geq I\left(Y;X_1\right).$$

Here, the term $I\left(Y;X_2|X_1\right)=H(Y|X_1)-H(Y|X_1,X_2)$, and it is always positive for proper probability mass functions or densities. This fact can be used to derive a search tree pruning technique much in the same way that the technique for sum-squared error is derived. Unfortunately, Chapter 5 shows that this inequality does not always hold when nonparametric density estimation is used to estimate the probability distributions involved in the mutual information computation. It is possible that a different approach to nonparametric density estimation than the one I explore would cause the inequality to hold. This could be a topic for future research (Chapter 8). Because the density estimate method I explore is most accurate for small groups of variables, forward selection is a natural choice that I rely upon throughout the thesis. Results for the effectiveness of forward selection appear in Section 5.4.

CHAPTER 4

MUTUAL INFORMATION AS A MEASURE OF RELEVANCE

It does not follow immediately that mutual information is a natural indicator of which input variables will result in good generalization performance. There are many possible indicators, including the common choice of "mean squared error" on a set of predictions. This chapter argues that:

(1) Mutual information is, in fact, a natural measure of how predictable an output value is for a given set of input variables.

(2) Under certain conditions, the relevance measure of mutual information $I(X;Y)$ and the relevance measure of expected conditional variance $E_x[\mathrm{Var}(Y|X)]$ are related directly. A simple counterexample shows that these two measures are not always related.

(3) While mutual information is hard to compute analytically in practice, it has an unbiased, sample based approximation that converges asymptotically to the true value.

The next section argues that mutual information is an appropriate measure of input variable relevance because of its close ties to predictability of the output value. Section 4.2 presents some well known properties of mutual information and expected conditional variance, and it then draws a connection between these measures from the properties. It then shows what can happen if the properties are violated. Section 4.3 shows that while computing mutual information analytically is difficult in practice, there is a sample-based approximation that converges to the underlying value on a simple test problem.

4.1 Why Use Mutual Information?

Mutual information is a natural way of expressing the relevance of input variables for prediction. It is possible to show that for a set of input variables $\mathbf{X}$ and output variable Y, high mutual information $I(\mathbf{X};Y)$ implies that there is little uncertainty about the output value when those input variables are known. To see this, we write the formula for mutual information as

$$\begin{aligned} I(\mathbf{X};Y) &= H(Y) - H(Y|\mathbf{X}) \\ &= -\sum_{y\in\mathcal{Y}} p(y)\log p(y) + \sum_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x}) \sum_{y\in\mathcal{Y}} p(y|\mathbf{x})\log p(y|\mathbf{x}) \ . \end{aligned}$$

Because the choice of output variable is constant over possible input variables, the leftmost summation is a constant. The rightmost term represents the expected value, over the inputs, of the conditional entropies $p(y|\mathbf{x})$. Therefore, high mutual information for some input variable subset implies low expected conditional entropy $H(Y|\mathbf{X})$. Low conditional entropy is good for prediction; there is more certainty of the output variable for a given input value (Figure 4.1). Although the figure shows Gaussian conditional distributions, the argument of low conditional entropy implying high predictability holds for a much broader class of conditional distributions than the Gaussian case, including multimodal densities.

4.2 Relationship to Mean Squared Error

An alternative approach might be to find input variables that minimize a different measure, such as Mean Squared Error (MSE). In fact, this is the measure used by Sensitivity Based Pruning (Section 3.1.3). MSE, rather than being a function of a joint density (as with mutual information), is a measure over a finite

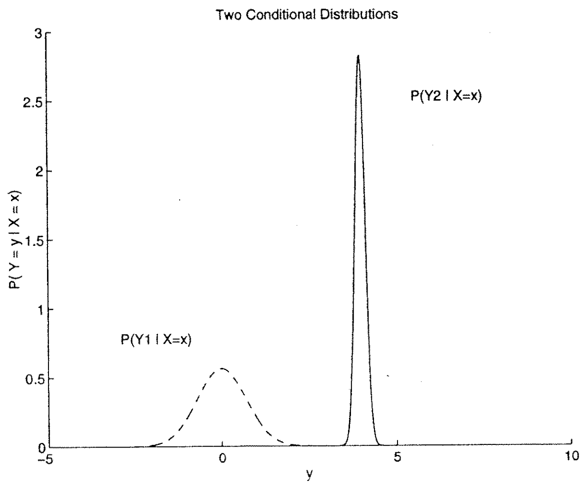


Figure 4.1. Entropy of continuous distributions. Two normal distributions, representing the probabilities of producing an output value given a particular input value x, are plotted. In the continous case, entropy is defined much like in the discrete case: $H(Y|\mathbf{X}) = \int p(\mathbf{x}) \int p(y|\mathbf{x}) \log p(y|\mathbf{x}) dy d\mathbf{x}$. Entropy for the left distribution is higher than the right one, which corresponds to the fact that it is harder to predict the value Y_1 from x than predicting the value of Y_2 from x.

sample:

$$\text{MSE} = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \hat{y}(x_i))^2 \ .$$

It is, however, an estimate for a more analytic measure, which, like mutual information, is a function of the joint density. The analytic measure is known as **expected conditional variance**. To derive the measure, assume for the moment that a fixed value x' is used to generate the $\hat{y}(x_i)$. Then MSE is an unbiased estimate of the variance of $p(Y|X = x')$: MSE $\approx \text{Var}(Y|X = x')$. Since different x_i are used, however, MSE is an unbiased estimate of the expected value of the variance or, in other words, the expected conditional variance:

$$\text{MSE} \approx E_x\left[\text{Var}\left(Y|X\right)\right] \ .$$

An important question is whether expected conditional variance will order subsets of input variables in the same way as mutual information. If so, then the two measures can be used interchangeably to select the best subset of input variables. As a partial affirmative answer, the following arguments will establish sufficient conditions for the correspondence of the orderings.

Lemma 4.1 (More Inputs Increase Mutual Information) For the three random variables X_1, X_2, and Y, $I(X_1;Y) \leq I(X_1, X_2; Y)$, with equality iff $X_1 = X_2$. In other words, adding input variables cannot decrease mutual information.

A proof is given in (Cover and Thomas, 1991), Chapter 2. It hinges on the following facts. First, $I(X_1, X_2; Y)$ can be rewritten as $I(X_1, X_2; Y) = I(X_1; Y) + I(X_2; Y|X_1)$, where the second term is the relative entropy

$$D(p(x_1, y \mid x_2) \,||p(x_1|x_2)p(y|x_2)) \ .$$

Since mutual information can be represented through relative entropy, and relative entropy is always nonnegative (Section 2.1.2), $I(X_2; Y|X_1) \geq 0$. Hence, $I(X_1;Y) \leq I(X_1, X_2; Y)$. □

A second lemma shows that adding input variables cannot increase expected conditional variance.

Lemma 4.2 (More Inputs Decrease Variance) For the two random variables X and Y, $E_x\left[\text{Var}(Y|X)\right] \leq \text{Var}(Y)$. In other words, the variance of a random variable is always greater than or equal to the expected variance of the random variable conditioned on another variable.

A simple proof is based on a formula for conditional variance.[1] For two random variables X and Y,

$$\text{Var}(Y) = E_x[\text{Var}(Y|X)] + \text{Var}(E_x[Y|X]) \ .$$

Since variance is always non-negative, we have $\text{Var}(E_x[Y|X]) \geq 0$; therefore, $E_x[\text{Var}(Y|X)] \leq \text{Var}(Y)$. □

Hence, we have the following theorem:

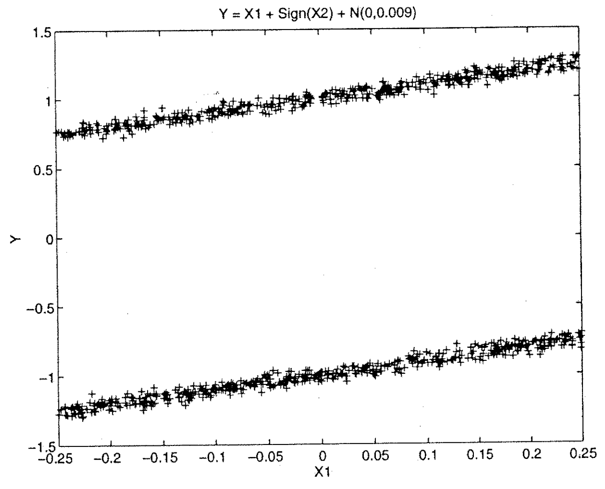
Theorem 4.1 Given a sequence of sets of input variables $\{\mathbf{X}_i\}$ where $\mathbf{X}_i \subseteq \mathbf{X}_{i+1}$. Then mutual information increases monotonically over the sequence, and expected conditional variance decreases monotonically over the sequence.

This follows directly from the two lemmas.

As a special case of the theorem, suppose all available input variables are i.i.d., where the output variable is some symmetric function over the inputs; for example, let $Y = f(\sum_i X_i)$, where $f()$ is any continuous function. Then any sequence of input variable sets will apply in the theorem, provided that simply the **size** of subsets in the sequence decreases monotonically. Any two subsets of the same size will have the same mutual information and expected conditional variance.

The theorem does not cover the case of an arbitrarily constructed sequence of input variable subsets. We cannot expect both measures to order such a sequence in the same way. Figure 4.2 illustrates a situation

[1] See, for example, (Ross, 1991).



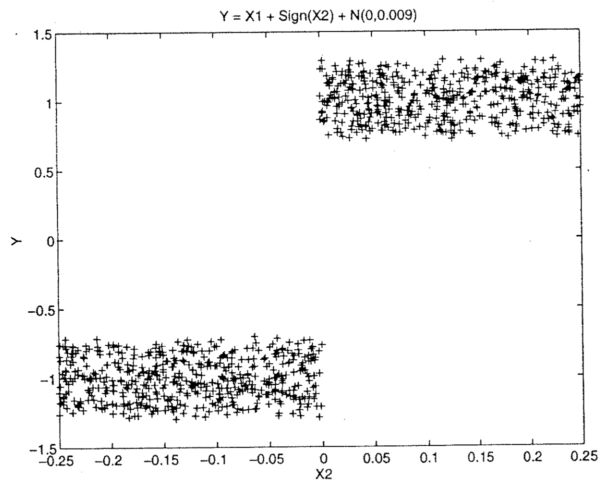


Figure 4.2. When mutual information and expected conditional variance do not correspond. Output values are generated using the equation $Y = X_1 + \text{Sign}(X_2) + \epsilon$, where ϵ is a normal random variable with mean zero and variance 0.009. The top and bottom figures plot the two input variables against the output. The variable X_1 (top) is more predictive of the output than X_2 (bottom) in an information theoretic sense, but its predictions of the output have a higher variance than X_2.

where the information content of an input variable can be very high, but the expected conditional variance is also high. Hence, mutual information will order the input subsets as $\{\{X_2\}, \{X_1\}, \{X_1, X_2\}\}$, from lowest to highest information, whereas expected conditional variance will order them as $\{\{X_1\}, \{X_2\}, \{X_1, X_2\}\}$, from highest to lower variance. This is a simple counterexample to illustrate that the two measures do not always correspond. Chapter 6 goes on to suggest that expected conditional variance is less general than mutual information: the former assumes that $p(Y|\mathrm{X})$ is unimodal, whereas mutual information does not.

4.3 Approximating Mutual Information from a Sample

A common procedure in this thesis involves estimating the mutual information between an output variable and different subsets of the possible input variables. While it is usually possible to compute mutual information analytically or approximate it accurately with numerical methods when the joint density and all marginal densities of interest are known explicitly, such procedures are rarely possible when only the joint density is known. This is due to the fact that for mutual information, it is necessary to perform two integration steps: one for producing marginal density functions, and another to compute mutual information. Suppose $\mathbf{X}_1$ is a subset of input variables, and $\mathbf{X}_2$ is the subset of remaining input variables. Then the required integral is written

$$I(\mathbf{X}_1; Y) = \int \left(\int p(\mathbf{x}_1, \mathbf{x}_2, y)\, d\mathbf{x}_2 \right) \log \left(\int p(\mathbf{x}_1, \mathbf{x}_2, y)\, d\mathbf{x}_2 \right) d\mathbf{x}_1\, dy\,.$$

Therefore, even when the entire joint density is known, it is sometimes required that mutual information be approximated from a finite sample, since the most that can be produced from estimates of marginal densities are probability values at specific points. Luckily, there is an unbiased estimate of mutual information that empirically converges to the true value as a function of the sample size, provided that probability values are exact. Since mutual information can be written as the expected value of a function,

$$I(\mathbf{X}; Y) = \sum_{\mathbf{x} \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(\mathbf{x}, y) \log \frac{p(\mathbf{x}, y)}{p(\mathbf{x})p(y)} = E_{\mathbf{X},y} \left[\log \frac{p(\mathbf{x}, y)}{p(\mathbf{x})p(y)} \right],$$

an unbiased estimator of mutual information is

$$\hat{I}(\mathbf{X}; Y) = \frac{1}{n} \sum_{i=1}^{n} \log \frac{p(\mathbf{x}_i, y_i)}{p(\mathbf{x}_i)p(y_i)}\,. \tag{4.1}$$

This is the equation I use for all sample based estimates of mutual information in the thesis. Comparisons of estimated and exact mutual information are made in Chapter 5.

4.3.1 Empirical Convergence Properties

I test the convergence properties of the estimator in equation (4.1) on the one problem I could find where marginal densities can be computed analytically, and where it is also possible to compute mutual information analytically. This problem consists of a multivariate normal density where input variables are independent, and the output variable is equal to the sum of input variables plus a normal noise variable. If there are m independent normal input variables with marginal variances σ_j^2, $j = 1 \ldots m$, and the noise term has variance σ_ϵ^2, the covariance matrix of the multivariate normal density has the form

$$\begin{array}{cccc}
\sigma_1^2 & 0 & \cdots & \sigma_1^2 \\
0 & \sigma_2^2 & 0 & \sigma_2^2 \\
\vdots & 0 & \ddots & \vdots \\
0 & \cdots & 0 & \sigma_m^2 \\
\sigma_1^2 & \cdots & \sigma_m^2 & \sum_i \sigma_i^2 + \sigma_\epsilon^2
\end{array}$$

For this case, the mutual information between any subset of the input variables and the output variable is given by

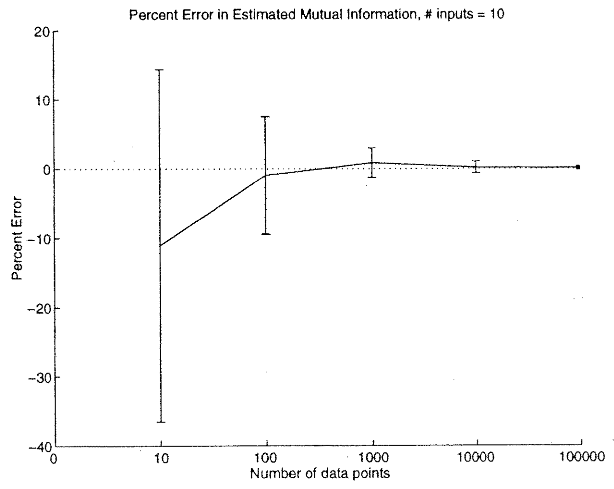
$$I(\mathbf{X};Y) = -\frac{1}{2}\log\left(1 - \frac{\sum_j \sigma_j^2}{\sum_i \sigma_i^2 + \sigma_\epsilon^2}\right) , \tag{4.2}$$

where the top summation is over the input variables included in the subset, and the bottom summation is over all possible input variables. The next paragraph derives this formula.

The above result for mutual information is an extension of the well known equation for the mutual information between two random variables in a bivariate normal distribution:[2] if r is the correlation coefficient between the two variables, then $I(X;Y) = -\frac{1}{2}\log(1-r^2)$. Equation (4.2) can be derived from this formula by making two observations. First, the covariance matrix for a subset of the input variables is obtained simply by deleting the rows and columns of the input variables that are not in the subset. Second, a composite random variable equal to the sum of the included input variables can be created, so that a two by two covariance matrix can be derived between the composite random variable and the output variable. The correlation coefficient is precisely the square root of the quotient in equation (4.2).

Figure 4.3 compares actual mutual information against estimates based on equation (4.1). The mean of the joint density is the zero vector, and all marginal variances equal 0.01. Probability values are calculated exactly, thanks to the simplicity of the multivariate normal distribution. As the top plot indicates, the estimate is approximately unbiased and converges to the true mutual information value. The bottom plot shows that the level of accuracy in the estimate is not strongly dependent on the dimensionality of the joint density; when thirty input variables are involved, the accuracy of the mutual information estimate is about the same as when only five input variables are involved. The slightly larger error bars for less than five are simply due to the plot involving a percentage error. Exact mutual information for less than five input variables is smaller than for ten, which makes the percent error slightly larger in these cases.

[2] See, for example, (Cover and Thomas, 1991), page 237.



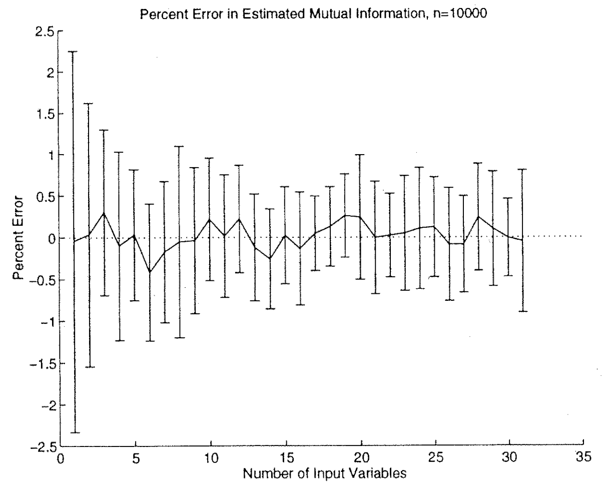


Figure 4.3. Convergence properties of estimated mutual information. Error bars indicate the variation of estimates across twenty different samples. The top plot indicates that the estimate is unbiased and converges to the true, underlying value. The bottom plot shows that accuracy of estimates does not depend strongly on the number of input variables.

CHAPTER 5

PRACTICAL CONSIDERATIONS IN SELECTING INPUT VARIABLES

The previous chapter shows that in theory, mutual information is a natural measure for selecting relevant input variables. However, analysis so far has not confronted two problems that occur when only a set of observations is available for determining relevance. First, probabilities must be estimated using some density estimation technique; it is not clear that density estimates can be accurate enough to distinguish relevant from irrelevant input variables. Second, it is not always practical to test relevance for every possible subset of input variables; some kind of search technique must be employed, beyond simple exhaustive search. This chapter explores the effects of finding relevant input variables using Epanechnikov product kernels for density estimation and forward selection as a search method. I attempt to explore the effects of these choices as independently as possible.

Highlighted throughout the chapter are two possible ways of estimating mutual information. The first way simply computes mutual information according to equation (4.1); I call this the **raw** mutual information estimate. This may, at first, appear to be the proper formula for mutual information estimates. However, the fact that underlying probabilities are estimated implies that raw estimates may no longer be unbiased.

As a way of controlling the bias of mutual information estimates, a second method is also explored. The second method subtracts a baseline value from the raw estimate; the baseline value is obtained by shuffling the target output values and estimating mutual information when the relationship between input and output is essentially random. Note that the baseline value must employ the same kernel width as the raw estimate, or one is subtracting off something different from a baseline mutual information estimate of "zero". I call the result the **adjusted** mutual information estimate.[1]

When either estimate is used in conjunction with forward selection, there is a natural stopping point in adding input variables. Adding input variables, or dimensions, to a density estimate lowers the density of data points in the estimate. This tends to make the points appear more random, or in other words, to contain less information. At some point, adding an input variable to the density will cause mutual information to worsen if the variable does not provide a sufficient amount of additional information.

Section 5.1 begins by determining how mutual information estimates are affected by Epanechnikov product kernel density estimation. The testbeds are simple prediction problems where exact probability calculations are still possible. The results show that while both the raw and adjusted estimates of mutual information deteriorate as a function of the number of input variables, the correct ordering of relevance estimates is still preserved. The number of input variables is small enough for these problems that exhaustive search over the variables is still possible. I also show how relevance estimates are affected by the presence of irrelevant and redundant input variables. In general, raw estimates are more susceptible to selecting dependent or redundant input variables than adjusted estimates.

Section 5.2 goes on to examine more complex prediction problems where exact probabilities are no longer available as a point of comparison. The level of noise in the output variables for these prediction problems is varied to determine what interaction may exist between predictability of output values and accuracy of relevance estimates. The results show that noise can make relevance ordering more difficult when data the number of data is small, but the effect diminishes as the number of data increases. Except in cases where prediction is inherently difficult, raw and adjusted mutual information estimation correctly distinguish relevant

[1]This "adjusted" technique for mutual information estimation is also explored in (Optican et al., 1991) for removing bias from estimates involving discrete data.

from irrelevant input variables; however, adjusted estimation requires more data on problems where the number of relevant input variables is large.

Section 5.3 then varies data set size to determine how data set size affects the ability of mutual information estimates to select all relevant input variables. Forward selection is used to search the large space of possible input variable subsets. It is shown that data sets must grow exponentially with the number of relevant input variables. However, it is still possible to select on the order of five to ten relevant input variables with around ten thousand observations on the problems studied.

Lastly, Section 5.4 shows whether forward selection search runs into local minima as it tries to find relevant input variables. The results show that forward selection using adjusted estimates is less susceptible to local minima, although this effect is lessened when raw estimates involve the same amount of computation as adjusted estimates.

It should be noted that all adjusted mutual information estimates in this chapter are derived using an **average** baseline estimate of "zero". Because individual shuffles of the output data can lead to a noisy estimate of the baseline, ten different shuffles are performed, and the average of the ten baseline estimate is subtracted from the raw estimate. This is the standard procedure used throughout the thesis unless otherwise indicated.

5.1 Effects of Nonparametric Density Estimation on Mutual Information Estimates

In Section 4.3.1 from the previous chapter, it was shown that accurate, sample based estimates of mutual information are possible if probabilities from the joint density are known exactly. This section shows what happens when the probabilities are estimated instead. I examine what happens to both the mutual information estimates themselves and the empirical relevance orderings of input variable subsets.

Results are based on three problems, where it is possible to marginalize out arbitrarily many input variables. A key characteristic of these problems is that the output variable is a function of the sum of relevant input variables. The summation allows the input variables to be replaced by a composite random variable with the same distribution as the sum of inputs. In computing a marginal density, this reduces the number of analytic or numerical integrations to just one. The problems are:

(1) **Linear**: the target output values are generated according to $Y = \left(\sum_{i=1}^{5} X_i\right) + \epsilon$, where input variables are i.i.d. normal with mean zero and variance 0.01.

(2) **Sign**: the target output values are generated according to the equation $Y = \text{Sign}\left(\sum_{i=1}^{5} X_i\right) + \epsilon$, where input variables are i.i.d. uniform with bounds $\left[-\frac{1}{2}, \frac{1}{2}\right]$.

(3) **Piecewise**: the target output values are generated according to $Y = f\left(\sum_{i=1}^{5} X_i\right) + \epsilon$ (where $f(x) = x + 2.5$ if $x < 0$ and $f(x) = x + 2.5$ otherwise), and input variables are i.i.d. uniform with bounds $\left[-\frac{1}{2}, \frac{1}{2}\right]$.

For all problems, the noise term ϵ is normal with mean zero and variance 0.01.[2] As described in Section 4.3.1, the first problem admits an analytic expression for the mutual information between the output variable and any group of input variables. For the other two problems, symbolic and numerical integration packages from Mathematica 2.1 are used to marginalize out input variables. Each mutual information value is estimated from a sample of 1000 points; the probability of each point is accurate to fifteen decimal places. Therefore, while it is not possible to calculate exact mutual information values for the last two problems, reported values are approximately correct with a small margin of error.

Figure 5.1 shows how mutual information compares when probabilities are estimated versus when probabilities are known. The six plots are organized so that the left column involves raw estimates, and the right column involves adjusted estimates. From top to bottom, the plots correspond to the Linear, Sign, and Piecewise problems. Estimated values are plotted with solid lines, and exact values are plotted with dotted

[2] While it is uncommon to add noise to the output of a classification problem as in (2), it is done here to make the true mutual information a finite quantity. The output noise is small enough to preserve the problem's classification characteristics.

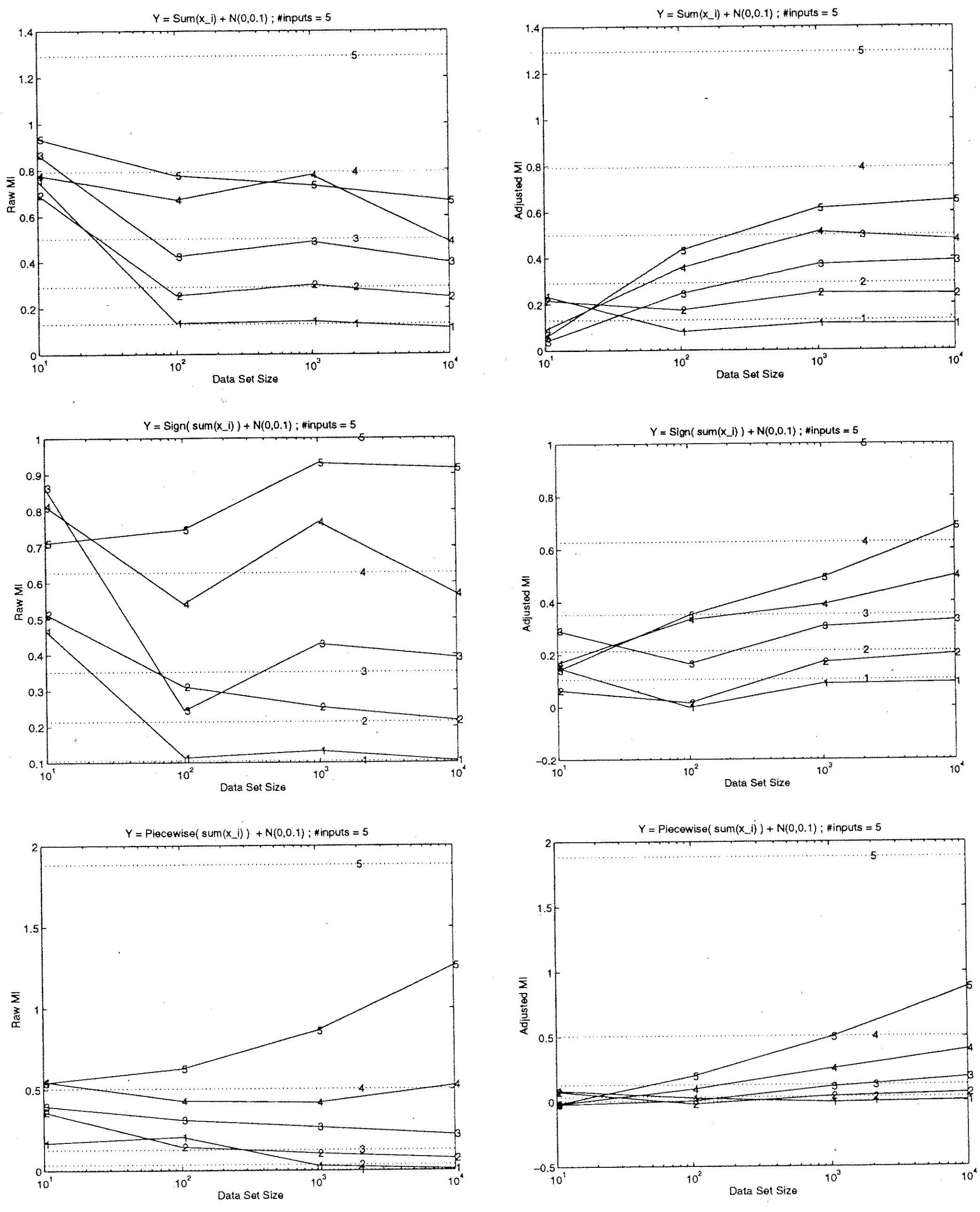


Figure 5.1. Estimated and actual mutual information for three problems. The first, second, and third row of figures correspond to the Linear, Sign, and Piecewise problems, respectively. The first column uses raw estimates, and the second column uses adjusted estimates. Each line is marked with the number of input variables involved; input variables are all equally relevant, which reduces the number of truly distinct subsets of input variables from 31 to 5. Estimated values are plotted with solid lines, and actual values appear with dotted lines.

lines. Even though each problem has 31 possible subsets of input variables, results for only 5 are presented because input variables are i.i.d.; there are only 5 truly distinct input variable subsets.

First, all plots show a general trend where estimated mutual information for small input variable subsets is more accurate than for larger subsets. An important difference between large and small subsets is the density of observations involved in an estimate: the density of points drops off exponentially as more input variables are considered. The plots also show a trend where estimates become more accurate as the data set size grows. In other words, it appears that the accuracy of estimated mutual information depends on how densely the observations populate the density estimate. Notice, also, that raw mutual information estimates appear at least as accurate as adjusted estimates for the four and five input variable cases.

Happily, the relevance orderings of input variable subsets are correct on all three problems: estimated relevance of five input variables is consistently highest, and estimates decrease as the number of input variables decrease. However, these results do not address whether the presence of irrelevant or dependent input variables may change the estimated orderings significantly. The following sections address these issues.

5.1.1 Effect of Irrelevant Input Variables As a further test of what happens when nonparametric density estimation is used, I explore the effect of including irrelevant input variables. Analytically, the mutual information between input and output variables should remain unchanged when irrelevant input variables are added. However, when kernel density estimation is used, the presence of irrelevant input variables affects mutual information estimates.

Figure 5.2 demonstrates the effects for both raw and adjusted estimation methods. The plots are organized as they were in Figure 5.1. To create these plots, a certain number of irrelevant input variables are added to all five relevant input variables, resulting in input variable subsets that vary in size from five to thirteen. Lines in the plots are labeled with the number of irrelevant input variables involved.

On all three problems, raw estimates of mutual information fluctuate around the original estimate as the number of irrelevant input variables are increased. The magnitude of fluctuations tends to lessen as the sample size grows, but even this behavior decreases the desirability of raw estimation method, since it implies that the addition of irrelevant input variables can unpredictably increase or decrease the estimated predictiveness of a set of input variables.

In the same figure, adjusted mutual information estimates are better behaved. For each problem, the addition of irrelevant input variables causes mutual information estimates to decrease, provided the number of observations is not too small. This fact will, in practice, prevent irrelevant input variables from ever appearing to contain predictive information, since the removal of irrelevant input variables increases the apparent relevance of the other input variables. The plots suggest that input variables selected using adjusted mutual information estimation will never contain irrelevant input variables.

5.1.2 Effect of Dependent Input Variables I also examine what effect the presence of dependent input variables has upon mutual information estimates. In order to simulate the presence of dependent variables, I copy values from a single relevant input variable a certain number of times. I also explored performing monotonic nonlinear transformations on these duplicate variables, such as the exponential, logarithmic, and square root functions, and I found that such transformations make little or no difference in the mutual information estimates themselves.[3] Figure 5.3 displays the plots in the same format as the other figures: mutual information estimates based on raw estimation appear to the left, and plots using adjusted estimation appear to the right. Plots for the Linear, Sign, and Piecewise problems appear from top to bottom.

The first noticeable trend across all problems is that mutual information estimates using the raw method tend to increase as dependent input variables are added, and estimates using the adjusted method tend to decrease as dependent input variables are added. As with the irrelevant variables, raw estimates have the unfortunate property of making it appear as if the addition of unimportant input variables increase the

[3]In each case, I rescaled the values to have the same mean and variance as the untransformed values.

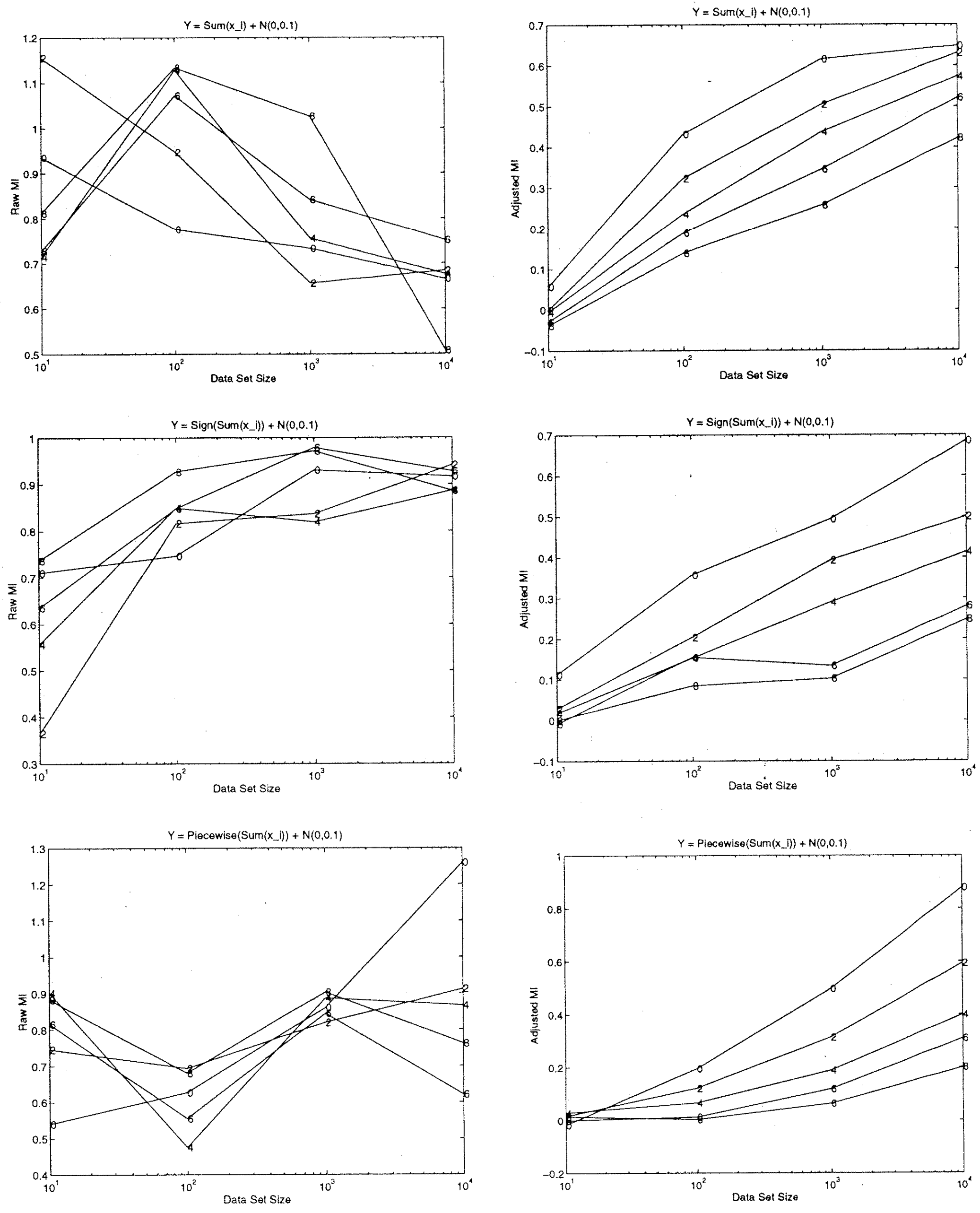


Figure 5.2. Effect of irrelevant input variables. The first, second, and third row of figures correspond to the Linear, Sign, and Piecewise problems, respectively. The first column uses raw estimates, and the second column uses adjusted estimates. All mutual information estimates involved the five possible input variables and a certain number of irrelevant input variables with the same distribution. Each line is marked with the number of irrelevant input variables involved.

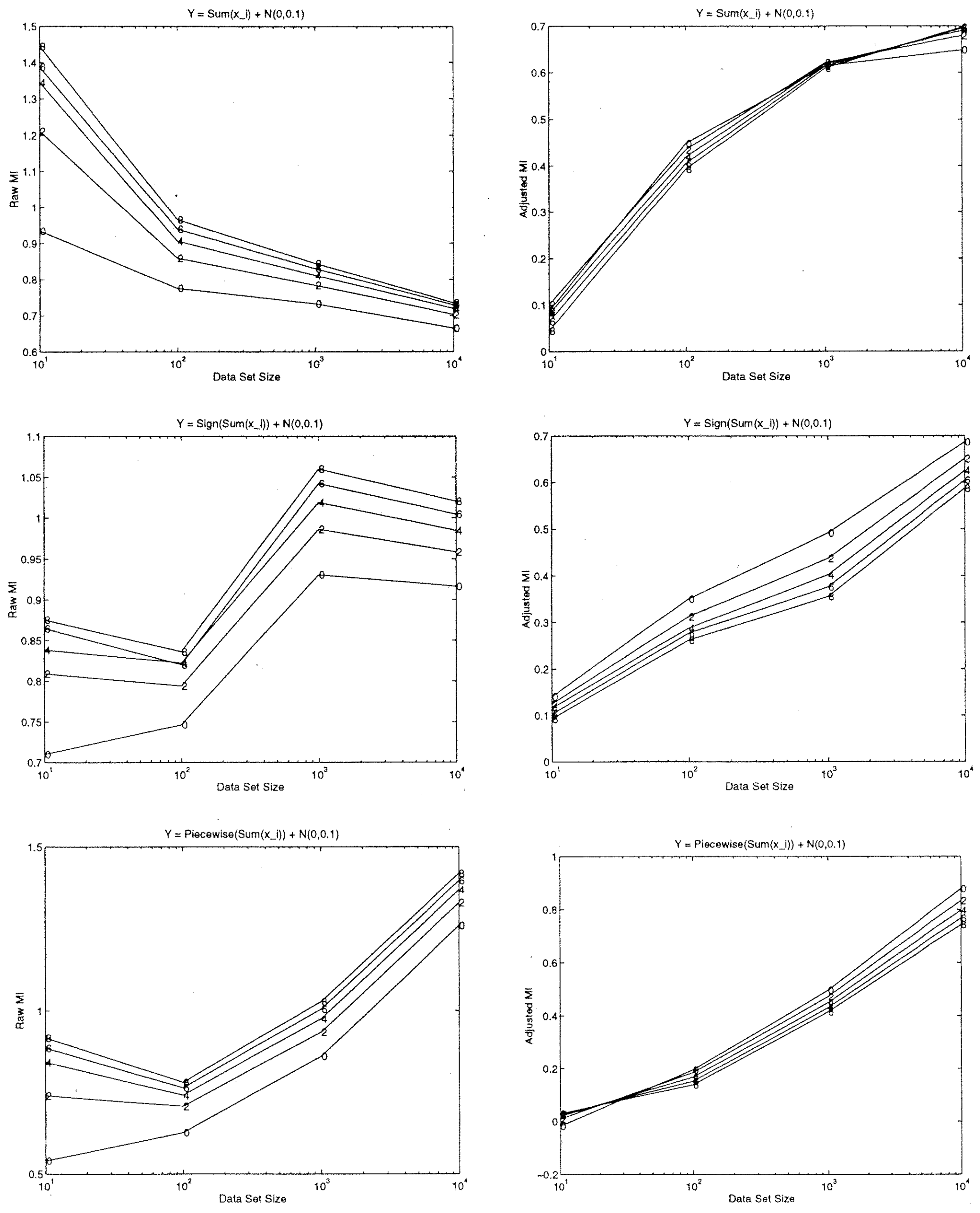


Figure 5.3. Effect of dependent input variables. The first, second, and third row of figures correspond to the Linear, Sign, and Piecewise problems, respectively. The first column uses raw estimates, and the second column uses adjusted estimates. All mutual information estimates involved the five possible input variables and a certain number of dependent input variables, which are all duplicates of a single relevant input variable. Lines are labeled with the number of dependent input variables involved.

Label	Structure	Noise Variance
linid	$\sum_{k=1}^{5} \alpha_k x_k$	$\{0.7, 0.8, 0.9\}$
linsin	$\sum_{k=1}^{5} \sin(\alpha_k x_k)$	$\{0.0, 0.3, 0.6\}$
rbfnet	Sum of RBF's	$\{0.0, 0.3, 0.6\}$
prodid	$\prod_{k=1}^{3} x_k$	$\{0.0, 0.1, 0.2\}$
prodsin	$\prod_{k=1}^{3} \sin(\alpha_k x_k)$	$\{0.0, 0.1, 0.2\}$
signet	25/8/1 Network	$\{0.0, 0.3, 0.6\}$

Table 5.1. Summary of problems explored for estimated relevance orderings. For the linid problem, the coefficients α_k have the form $\alpha_k = (1 - (k-1)\,0.2)$, so that the relevance of input variables decreases linearly. For the linsin problem, the α_k have the form $\alpha_k = 2^{k-1}$ to create complex nonlinearities in the output function. For the prodsin problem, the coefficients have the same form as for linsin.

predictiveness of the entire set of input variables. As it will be shown in the next section, however, there is an advantage to the raw estimation method: it has the ability to order the relevance of large input variable subsets more correctly than the adjusted estimation method.

The top right plot of Figure 5.3 shows an interesting effect: as the sample size grows, the ordering of estimates reverses itself, and estimates with dependent variables appear slightly more relevant than those without dependent input variables. Therefore, it seems that neither raw nor adjusted estimation is totally immune to selecting dependent input variables. The problem in the top right plot can be understood by examining equations (2.1) and (4.1). Adding dependent input variables does not cause the kernel width in density estimates to change, but equation (2.1) shows that adding redundant inputs lowers probability values slightly. The more redundant input variables, the lower the probabilities. Lower probabilities imply higher mutual information estimates; to see this, notice the three different density estimates involved in equation (4.1). The lowering is slightly less severe for $p(\mathbf{x}_i, y_i)$ than for $p(\mathbf{x}_i)$, due to the presence of the output variable y in the former. The density estimate for $p(y)$ remains constant for whatever number of redundant input variables are present.

One would think that the baseline that is subtracted to produce an adjusted estimate would be raised an equal amount as more redundant input variables are added. However, for the Linear problem, this does not appear to be true when the size of the data is large enough. Whether there is some way of overcoming this problem is left for future research.

5.2 Further Tests of Input Variable Relevance Orderings

The previous section shows that estimating mutual information gets less accurate for more input variables, but that raw and adjusted mutual information still can correctly order the predictive relevance of input variable subsets. However, the results cover a limited set of problems, where the output is a function of the sum of input variables. This simplifies analysis by making all subsets of the same size have equal relevance. This section explores problems where input variables have a more complicated relationship with the output variable. For some problems explored here, all possible subsets of input variables are distinct and must be examined separately. This poses a much more difficult test for raw and adjusted estimation to order estimated subset relevances correctly. The problem characteristics are described first, followed by the results.

5.2.1 Description of Prediction Problems The problems selected for this exploration are carefully chosen to have a wide variety of structural characteristics, so that results are less dependent on any particular kind of problem. To avoid confounding the study with the choice of an input variable search algorithm, relevance estimates are performed for all distinct subsets of relevant input variables. Further tests involving forward selection appear in Section 5.4.

Table 5.1 summarizes the problems explored. The first five are introduced in (Finnoff et al., 1993). For each problem, the table lists the name used to denote the problem in subsequent analysis, the general form of the output variable as a function of input variables, and the variance of additive, uniform noise added to the output variable to create target values.

For the first five problems in the table, there are ten possible input variables, all i.i.d. uniform with values in the range $[-1, 1]$. Output values are produced by using the equations in the table. Modified output values $\tilde{y}_i$ are then created by adding uniform noise with variances equal to the rightmost column in the table (three values for a problem imply three separate prediction problems), and then by applying a nonlinear transformation. For the linid and linsin problems, the transformation has the form

$$y_i = \tanh\left(\frac{\tilde{y}_i - \bar{y}}{\sigma_y}\right) , \tag{5.1}$$

where the $\tilde{y}_i$ are the untransformed targets, $\bar{y}$ is the average of the $\tilde{y}_i$, and σ_y is the sample standard deviation of the $\tilde{y}_i$. For the prodid and prodsin problems, the transformation has the form

$$y_i = \text{sign}\left(\tilde{y}_i\right) .$$

For both transformations, the resulting y_i are the actual target values used in prediction.

The rbfnet problem has an output function of the form

$$y_i = \sum_{l=1}^{8} (-1)^l \exp\left(\sum_{k=1}^{5} \frac{\left(\alpha^{k,l} - x_i^k\right)^2}{2\sigma^2}\right)$$

There are, in other words, eight Radial Basis Function (RBF) units and five relevant input variables involved. No additional transformation is applied to the outputs y_i. A value of $\sigma^2 = 0.3$ is chosen so that the prediction problem has highly local properties; according to Finnoff et al., values larger or smaller than this result in trivial prediction problems.

The signet problem involved 25 input variables, eight tanh hidden units, and a single linear output unit. Input variables are i.i.d. uniform in the range $[-1, 1]$. Exhaustive evaluation of all input variable subsets is not possible with this problem; however, the number of distinct subsets is reduced by giving the input variables equal relevance. This is accomplished by selecting a single vector of weights between an input unit and the hidden unit layer, and then permuting these weights randomly from input variable to input variable. This reduces the number of distinct input variable subsets from $(2^{25} - 1)$ to 25. The vector chosen for permutation is $\{-3, -3, -3, -3, 3, 3, 3, 3\}$. The hidden-to-output weights are equal to $\{-1, -1, -1, -1, 1, 1, 1, 1\}$. Finally, the target outputs are produced using the transformation of equation (5.1).

5.2.2 Discussion of Results For the linid, linsin, rbfnet, and signet problems, all possible subsets of the relevant input variables are evaluated and ranked. This ranking is compared to the actual relevance ranking of the input variable subsets, and a correlation coefficient is computed. It should be noted that finding the actual ranking of input variable relevance is not always easy for prediction problems, even when the form of the equations generating the data are known. In particular, the linsin problem consists of a sum of sinusoidal functions, where the relative contributions of each sinusoid is not easily quantified. In the analysis presented here, I assume that each input variable has roughly equal relevance for the linsin problem.

For the linid, linsin, and rbfnet problems, 31 input variable subsets are ranked. Because there are only three input variables of equal relevance for the prodid and prodsin problems, the relevant input variable subsets of size one, two, and three are compared, along with the possible addition of zero, one or two irrelevant inputs, for a total of nine subsets to be ranked for these problems.

Figures 5.4 and 5.5 show the correlation coefficients of estimated relevance ranking against actual relevance ranking, obtained as a function of the sample size. The left three plots in each figure contain results

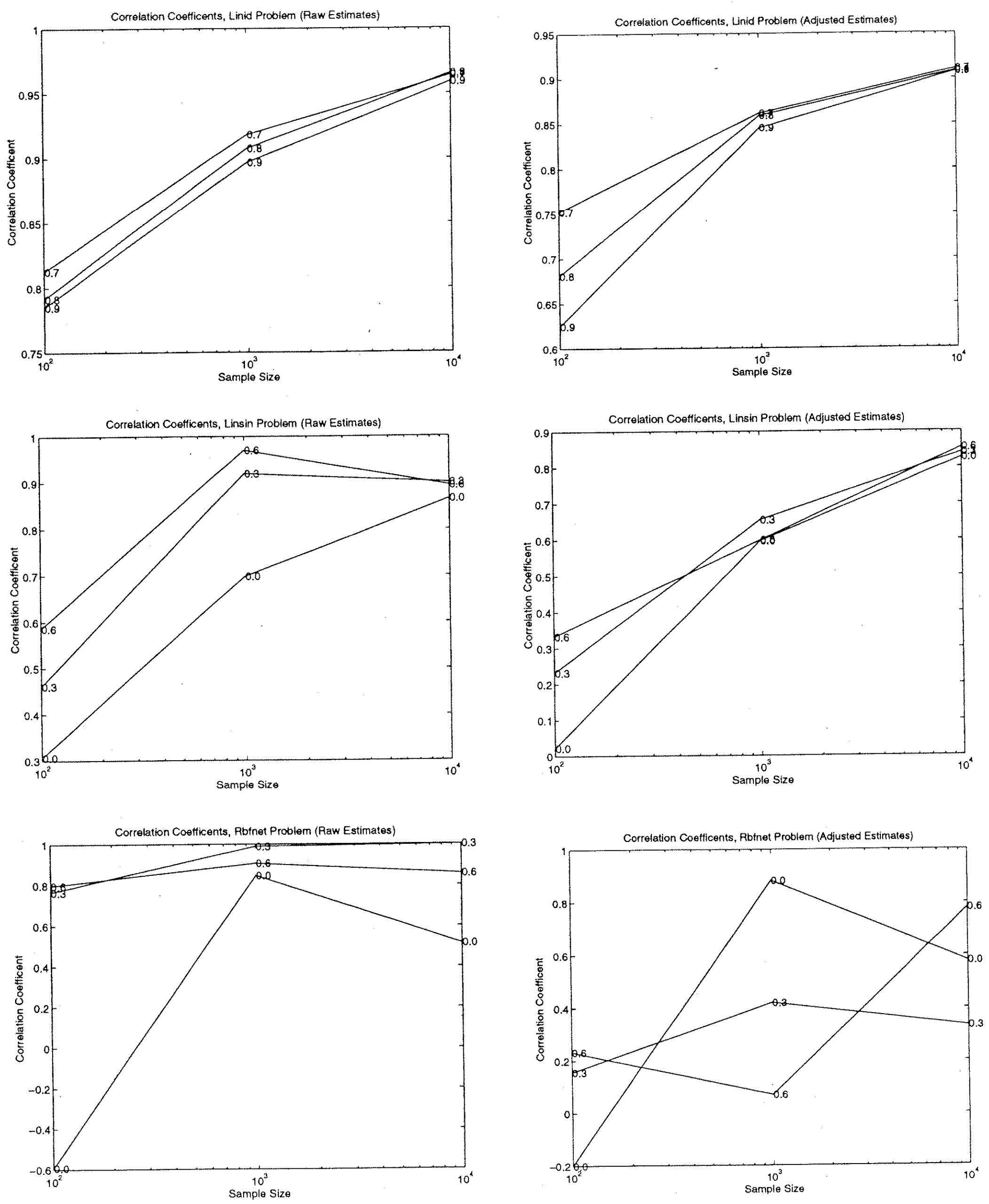


Figure 5.4. Input variable relevance orderings. The first, second, and third row of figures correspond to the linid, linsin, and rbfnet problems, respectively. The first column uses raw mutual information estimates, and the second column uses adjusted estimates. Rank ordering correlations are plotted as a function of sample size. Lines are labeled with the output variable's additive noise level.

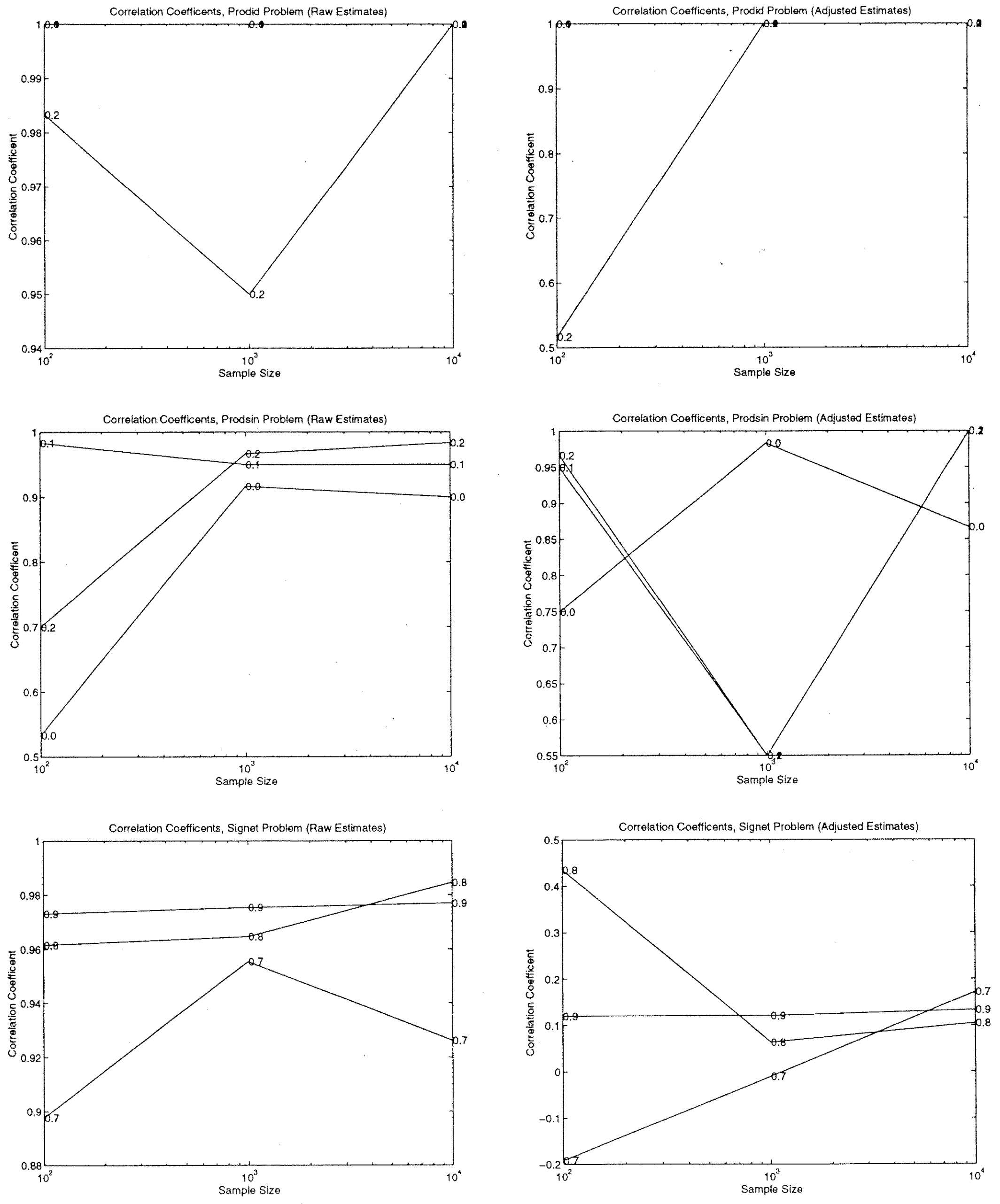


Figure 5.5. More input variable relevance orderings. The first, second, and third row of figures correspond to the prodid, prodsin, and signet problems, respectively. The first column uses raw mutual information estimates, and the second column uses adjusted estimates. Rank ordering correlations are plotted as a function of sample size. Lines are labeled with the output variable's additive noise level.

for raw mutual information estimation; the right three contain results for adjusted estimation. From top to bottom, Figure 5.6 contains results for the linid, linsin, and rbfnet problems, and Figure 5.6 contains results for the prodid, prodsin, and signet problems.

For the linid, linsin, prodid, and prodsin problems, the rank of estimated mutual information corresponds well with the actual rank. Increasing the the sample size causes an overall increase in correlation. Generally, increasing noise in the output variable causes the correlation to decrease, but this effect is diminished as the sample size increases. Results for the linsin problem are a possible exception; it appears that noise increases correlation for raw estimates. This may actually be due to an incorrect assessment of how input variable subsets are actually ordered. In the plot, it is assumed that the more input variables involved, the better; however, because the output is a sum of nonlinear functions, it is difficult to tell how much information is contained in subsets of input variables; it may actually be that certain nonlinear combinations of fewer inputs provide more information than other nonlinear combinations of more inputs. In any case, it is still true that the effect of noise diminishes for large sample sizes. For the prodsin problem, there is also a greater variation in correlations than for the other problems. This is most likely caused by ranking only nine distinct subsets, where small variations in ranking cause larger changes in the correlation coefficient.

For the rbfnet problem, a first attempt at estimating mutual information resulted in estimates of zero for all input variable subsets (up to 5 decimal places), due to the existence of outliers in the output values. Most output values are small, but for a handful of instances, output values are huge because the corresponding input values lie close to an RBF center. As a result, the density estimation algorithm is forced to choose large kernel widths in order to span the points, as required by likelihood cross validation (Section 2.2). Hence, mutual information estimates were originally close to zero. Results in Figure 5.4 are reported by first eliminating the top two percent of points whose L-infinity distance to a nearest neighbor is largest. This procedure eliminates the problematic outliers, but it also eliminates information about the location of RBF centers. Hence, this problem is especially difficult for the relevance estimation approach, in that most data points contain very little information, and the remaining points consist of large values. As I report in Chapter 7, however, all input selection methods have difficulty with this problem. Thus, the low correlations for this problem can be seen as a result of the problem, and not the estimation method.

For the signet problem, the number of relevant input variables is much larger than for the other five problems. The difficulty of searching over all possible subsets of input variables is circumvented by making all input variables equally relevant. This problem reveals a weakness in the adjusted mutual information estimation approach: relevance estimates for more than about seven input variables plateau and then decrease as an increasing number of relevant input variables are considered, even with a sample size of 10000. Presumably, this plateau effect lessens with ever-increasing sample sizes, but nevertheless, the prospect of finding structure in points, when more than about seven relevant input variables are involved, looks remote when using adjusted mutual information estimation. As a result, the correlation between ranks of estimated and actual mutual information values is low for adjusted estimates, whereas it still remains high for the raw mutual information estimation approach.

Up to this point, results for raw and adjusted estimates have shown a number of things. Both appear to work for most of the problems explored. Raw estimates work better when a large number of input variables are involved, as shown by the signet problem. However, adjusted estimates appear to avoid selecting irrelevant or dependent input variables to a greater extent, as indicated by results from Section 5.1. It is not clear that one works better than the other, so the problems I study in the remainder of the thesis will continue to involve both estimation schemes to be safe.

The following section shows how well the two estimation approaches work in conjunction with the forward selection algorithm. Such a selection algorithm is necessary as a way of avoiding the evaluation of too many input variable subsets, which grow exponentially as a function of the number of possible input variables.

Results from the next section will show how much data are required to select a certain number of input variables, and whether forward selection encounters local minima in practice. The results are obtained using arrays of prediction problems where the number of relevant input variables is varied, in addition to the size of the data set. The problems are most complex than those studied so far, in that forward selection is used to select input variables, and both irrelevant and dependent input variables are present among the candidate inputs.

5.3 Interaction Between Problem Size and Data Size

The previous section showed that estimated relevance orderings are highly correlated with actual relevance orderings for most of the problems studied; however, each problem has an abundance of data for selecting relevant input variables. This section examines problems with many more input variables, in order to better determine how much data are required to select a certain number of relevant input variables. Here, forward selection is used to search the large space of possible input variables. The results demonstrate that the amount of data needed increases roughly exponentially as a function of the number of relevant input variables. While this may seem disastrous at first, it still appears possible, for some problems, to select between five and ten relevant input variables with around ten thousand data points, regardless of how many candidate input variables there are.

Both a linear and a nonlinear prediction problem are studied. The results for both problems are similar, which may seem surprising at first. With hindsight, however, it is clear that kernel-based nonparametric density estimation is sensitive to **local** structure in the data, and locally, smooth nonlinear functions such as the one studied here are approximately linear. Hence, results for both problems should be similar.

5.3.1 Description of the Data Sets The output functions for the linear and nonlinear prediction problems have the respective forms

$$y_i = \sum_{k=1}^{m} x_k$$

and

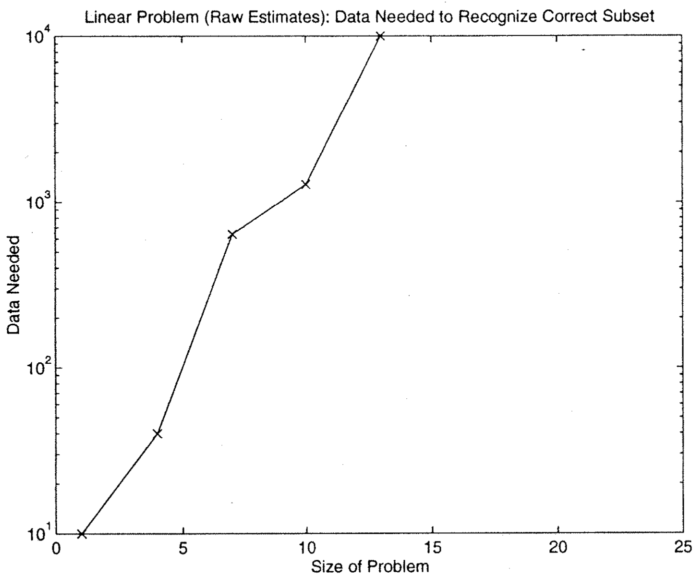
$$y_i = \tanh\left(\sum_{l=1}^{8} (-1)^l \tanh\left(\sum_{k=1}^{m} w_i x_k\right)\right) .$$

The values for the w_i in the second equation are chosen randomly from $\{-3, 3\}$, which made the m input variables involved in the calculation roughly equally relevant, but not exactly equal, as with the signet problem of the previous section.

The number of relevant input variables m for each problem is varied from 1 to 25 in increments of three, and the number of data points available for input variable selection varied along the values {10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 10000}. To avoid an extra confounding variable, no noise is added to the output values y_i. This results in a (9×11) array of prediction problems for each of the two output functions. For each subproblem in the array, three redundant input variables and three irrelevant input variables are also added.[4] In the cases with only one relevant input variable, only one redundant input variable is created, and two additional irrelevant input variables are included. All input variables are i.i.d. uniform in the range $[-1, 1]$.

5.3.2 Results Figures 5.6 and 5.7 show, for the linear and nonlinear problem respectively, the amount of data needed to select all relevant input variables as a function of problem size. The roughly straight line on the log-linear plots indicates an exponential increase in required data set size. Clearly, a great deal of data would be required to select every relevant input variable when there are more than about fifteen present. However, even for prediction subproblems with greater than fifteen relevant input variables, sometimes a large percentage of the relevant ones are selected (Figures 5.8 and 5.9). For example, nineteen relevant input variables are selected from a set of twenty five in the top plot of Figure 5.8. In practice, using a large subset of the most

[4]The redundant input variables are created by taking the exponential of the first three relevant input variables. The mean and variance of these variables are adjusted to match those of the original input variables.



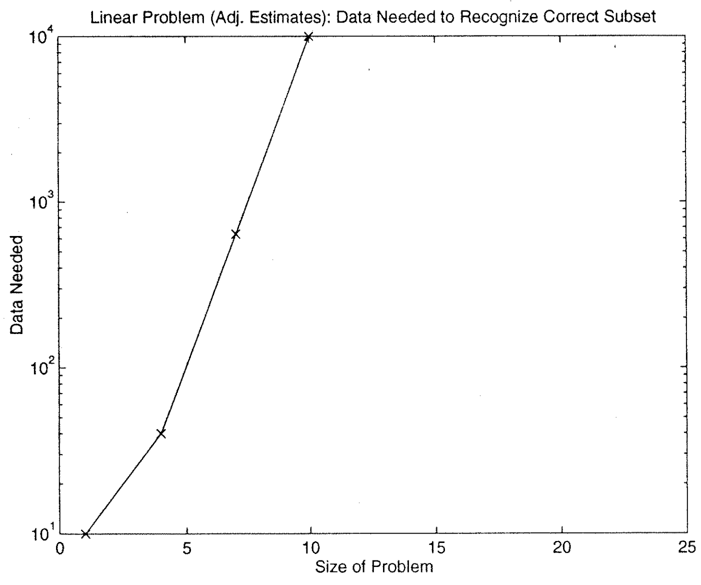
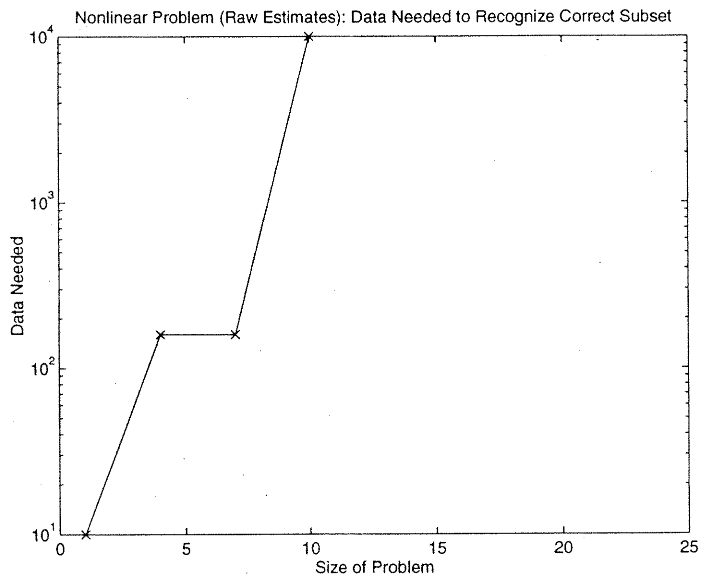


Figure 5.6. Linear problem: amount of data needed to recognize all relevant input variables. Plotted is the amount of data that is required before all relevant input variables are selected. The data requirements are a little more stringent when adjusted mutual information estimates are used (bottom plot) as opposed to raw estimates (top plot).



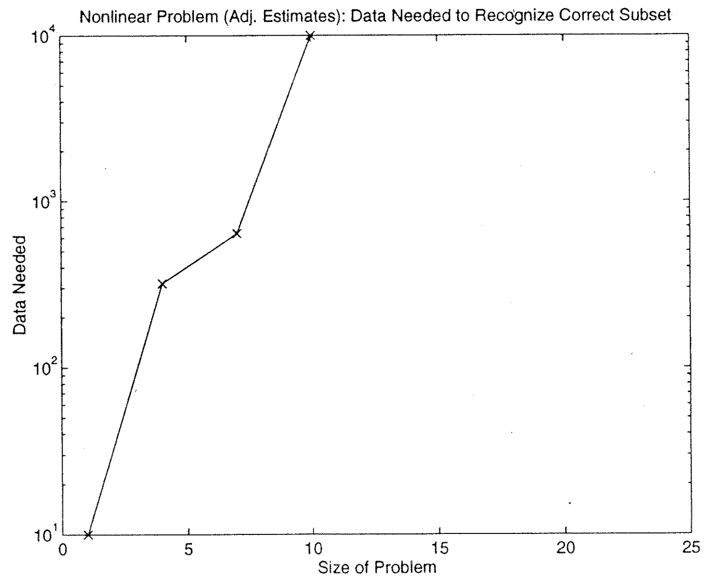
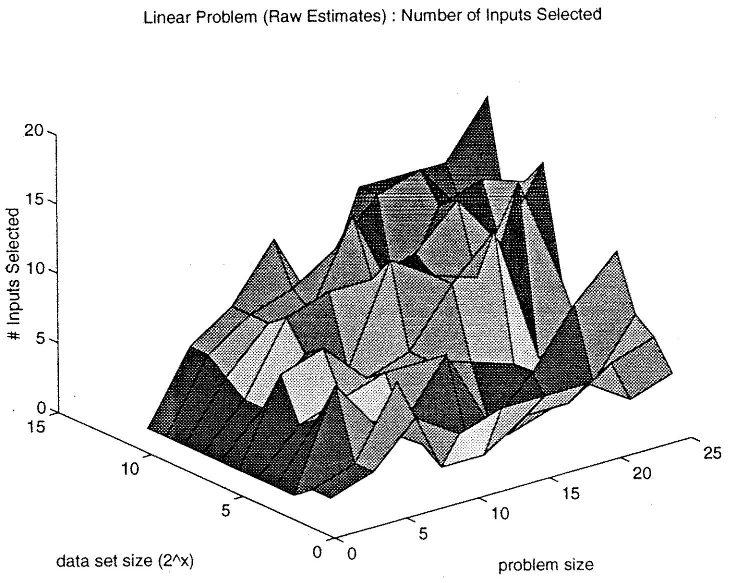


Figure 5.7. Nonlinear problem: amount of data needed to recognize all relevant input variables. Plotted is the amount of data that is required before all relevant input variables are selected. The data requirements are a little more stringent when adjusted mutual information estimates are used (bottom plot) as opposed to raw estimates (top plot).



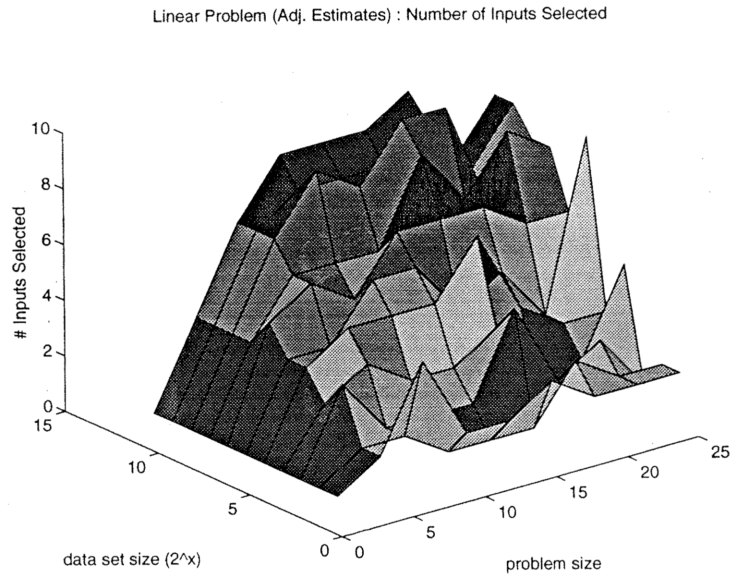
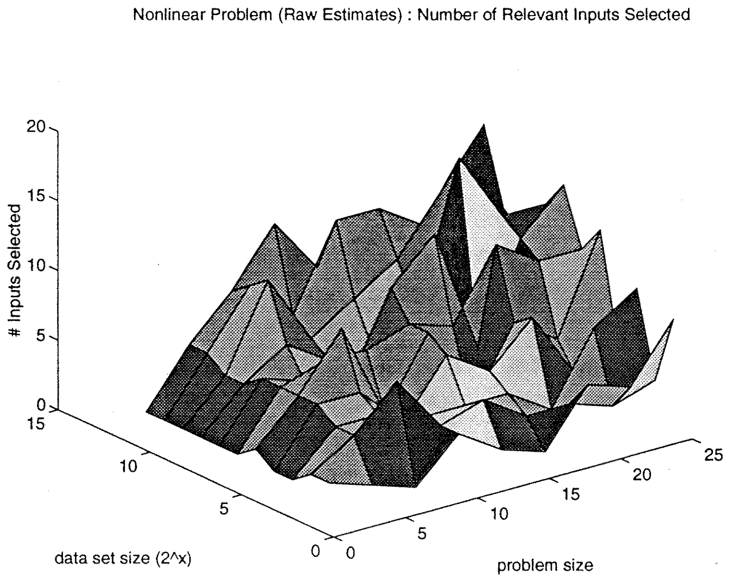


Figure 5.8. Linear problem: number of relevant input variables selected as a function of problem size and data set size.



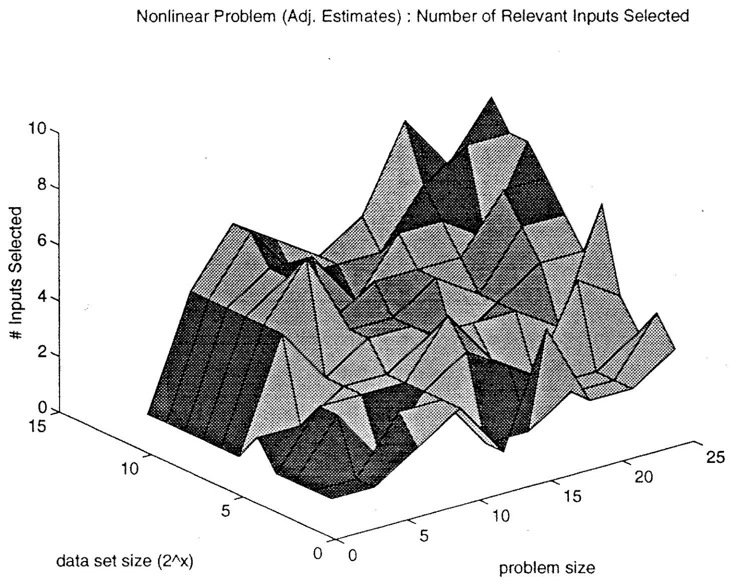


Figure 5.9. Nonlinear problem: number of relevant input variables selected as a function of problem size and data set size.

relevant input variables could still result in good predictions. If the focus of input variable selection is finding the best possible set, however, then the data set size requirements are demanding.

It can also be seen from Figures 5.8 and 5.9 that the number of selected, relevant inputs is not a smooth function of data set size and problem size. This can be alleviated to some degree by bootstrapping the data and basing the relevance of an input variable on more than one raw or adjusted mutual information estimate. This is not performed for these problems because of the already steep computational requirements.

Figure 5.10 shows the behavior of the algorithm in selecting irrelevant input variables on the nonlinear prediction problem. Clearly, the raw mutual information estimation approach, represented in the upper plot, is more susceptible to selecting irrelevant input variables. For both approaches, however, the severity of the problem decreases as the amount of data grows. Results for the linear problem are qualitatively similar.

I also show, in the nonlinear case, for which problems the forward selection algorithm selects dependent input variables (Figure 5.11). The results are consistent with those in Section 5.1, where raw estimates uniformly give higher relevance to dependent input variables, but adjusted estimates do so only as the number of data grows. Results for the linear problem are qualitatively similar.

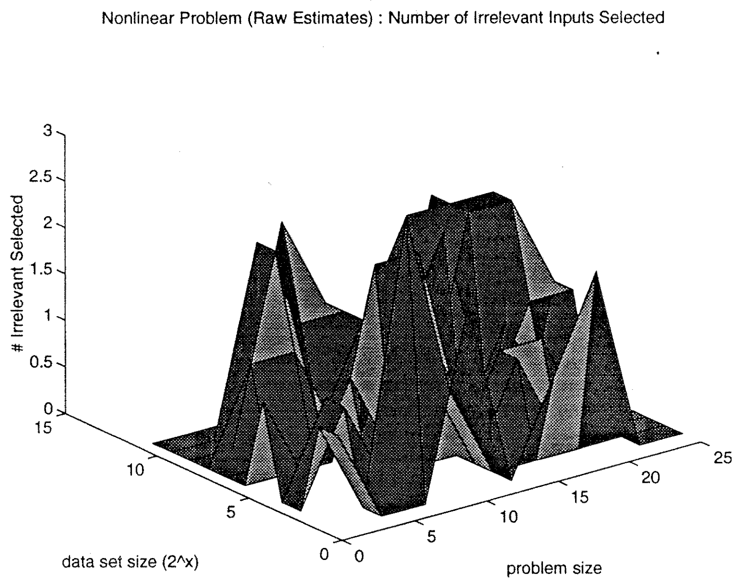
The dependent input variables are always selected last and, in retrospect, may be selected partly because their strength of redundance is so large (it technically does not "hurt" predictions to include these input variables, since the dependent input variables are non-noisy functions of each other and therefore introduce no extra uncertainty into predictions). If even small amounts of noise are added to the dependent input variables, it may be that dependent input variables would not be selected as readily. The interaction between degree of redundance and susceptibility to selection could be explored in future research.

5.4 Forward Selection and Local Minima

Finally, I test the susceptibility of forward selection search to local minima. For each choice of problem size and data set size, I estimate the relevance of the best possible set of input variables and compared it to the highest relevance estimate found by the forward selection algorithm. The forward selection algorithm is deemed to be in a local minimum only if its relevance estimate is lower than the estimate given to the best possible subset. This means that forward selection is not always at a local minimum if some relevant input variable remains unchosen: it is possible for additional relevant input variables to actually decrease an estimate of relevance for the entire subset of input variables.

Figure 5.12 presents the results for the linear problem as two matrices; one each for raw and adjusted mutual information estimates. Each square in a matrix represents a different prediction problem. If a square in the matrix is white, then the forward selection algorithm did not reach a local minimum. A shaded square indicates a local minimum, and the darkness of the shade indicates the level of severity of the local minimum. Raw estimates, which are based only on a single number, tended to give the forward selection algorithm more trouble; more than a third of the problems have local minima. The black square in the top plot, representing the worst local minimum, has an absolute difference of 0.723 from the corresponding estimate given to all relevant input variables. In contrast, only three problems resulted in local minima for adjusted estimates. The black square in the lower plot represents the maximum absolute difference of only 0.0508.

To see whether the difference in performance between the two estimates is due solely to greater computation for adjusted estimates, I reran the forward selection algorithm, where each raw mutual information estimate is based on the average of ten bootstrapped calculations, rather than a single calculation. Figure 5.13 shows that extra computation does, in fact, help forward selection in avoiding local minima. However, the severity of the worst local minimum, once again represented by a black square, is still greater than for adjusted estimates, with a value of 0.3076.



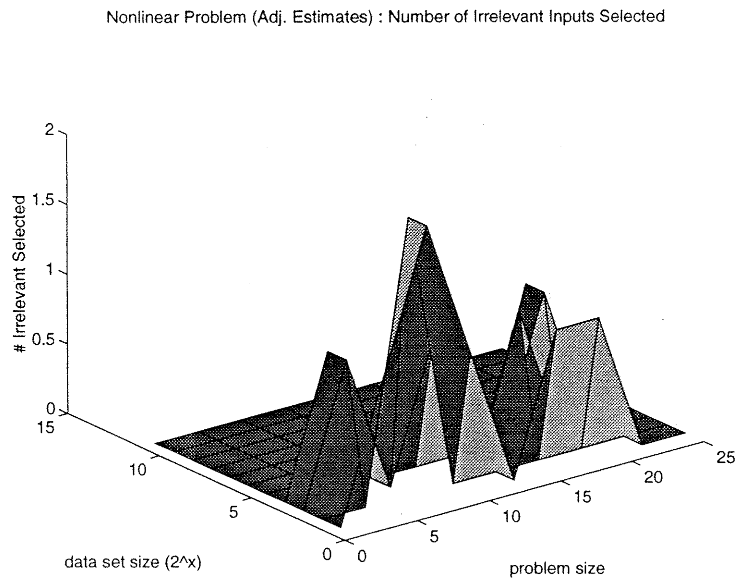
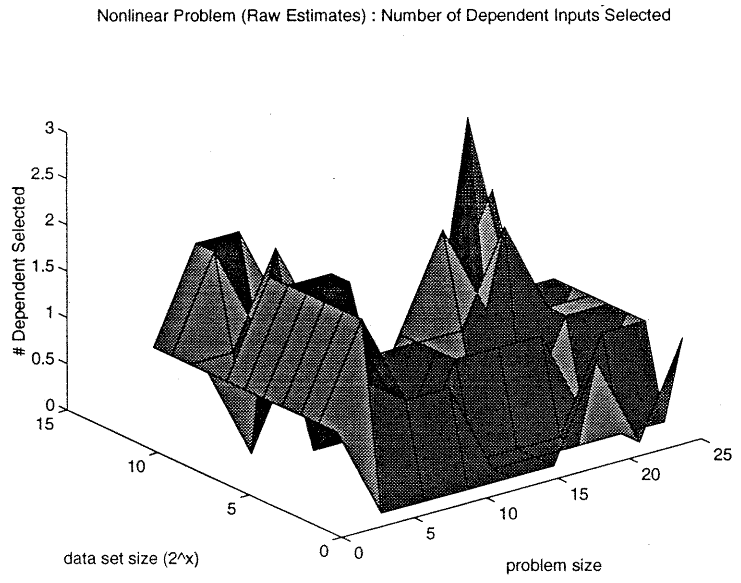


Figure 5.10. Nonlinear problem: number of irrelevant input variables selected as a function of problem size and data set size.



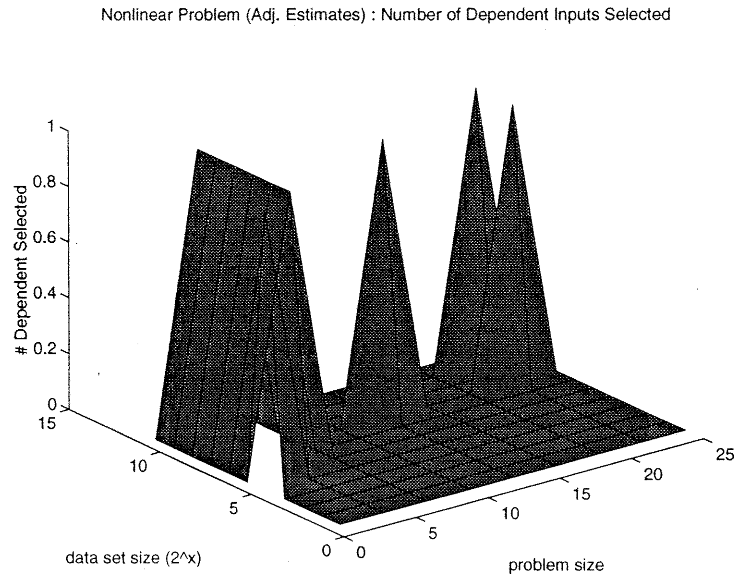
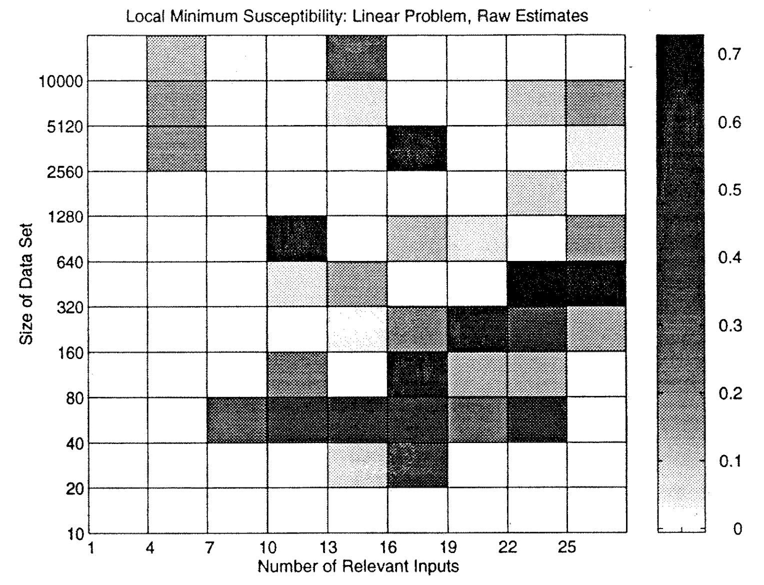


Figure 5.11. Nonlinear problem: number of dependent input variables selected as a function of problem size and data set size.



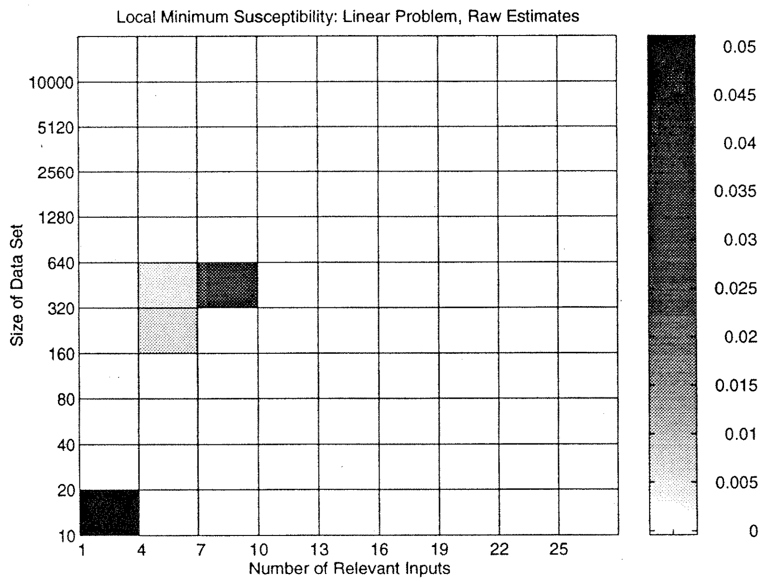


Figure 5.12. Linear problem: susceptibility of forward selection to local minima. The plots represent, for each prediction subproblem, the severity of the difference between the mutual information estimate for the search minimum and the estimated mutual information for all relevant input variables. The top plot uses raw mutual information estimates, and the bottom uses adjusted estimates. When the search minimum's value is better, the squares appear white; otherwise, the square is shaded. A darker shade indicates larger absolute difference between the two estimates.

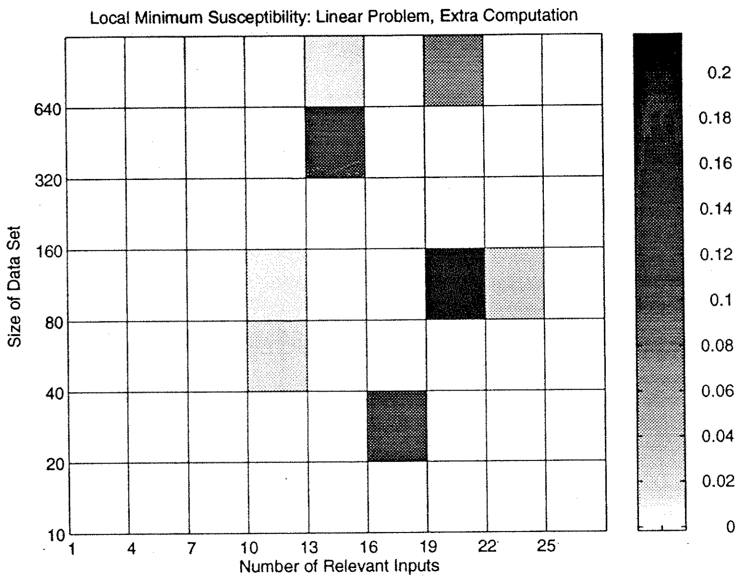


Figure 5.13. Linear problem: results with extra computation. The severity of differences between local and actual minimum are not as severe. The dark black square represents the maximum absolute difference of 0.3076.

5.5 Conclusions

This chapter has explored the ability of raw and adjusted mutual information estimates to order the relevance of input variable subsets. Sections 5.1 and 5.2 explored this ability, independently of the forward selection algorithm, for a range of problems. Section 5.3 then showed how well forward selection works in conjunction with these two estimation methods. Results indicate that both have advantages and drawbacks: raw estimates can select a greater number of relevant input variables for the same amount of data. However, it is more susceptible to selecting irrelevant or dependent input variables than adjusted estimates. The decision to use one or the other is based, ideally, on the number of relevant input variables that one believes to be present. If many are present, then input variables selected using raw estimates are likely to result in better performance. If only a handful of relevant input variables are present, then adjusted estimation is more likely to select them. Since the number of relevant input variables is usually not known in advance, the safest course is to select two sets of input variables; one set is selected using each method. It is possible, however, that slight changes to the way raw and adjusted estimates are performed would change the behavior of the two algorithms and make one the obvious choice. The conclusions in Chapter 8 discuss possible modifications to the two estimation methods.

CHAPTER 6

DETERMINING THE NUMBER OF MODES IN THE CONDITIONAL OUTPUT DISTRIBUTION

The two previous chapters have established that kernel density estimation and mutual information can be used to select predictive input variables. Chapter 4 showed that mutual information is a natural way of quantifying input variable predictiveness. In fact, it argued that mutual information is more general than error based measures, such as mean squared error, used in many connectionist approaches. Figure 6.1 helps emphasize the greater generality of mutual information. Shown in the figure are two joint densities between an input variable X and output variable Y. While mutual information is able to capture the information content of the input variable X in both cases, an error based measure underestimates the information content of an input variable in the bottom density because the conditional output distribution $p(Y|X)$ is bimodal.

However, the extra generality of mutual information comes with a price: after selecting a set of input variables, one does not know whether the the conditional output distribution is of the top or bottom type in Figure 6.1. If the distribution is unimodal, as in the top plot, then a standard connectionist network can be used effectively as a statistical model for the selected input variables. If the distribution is of the second type, however, a more sophisticated statistical model will be required. Possible models for the second case are discussed in (Bishop, 1994; Weigend and Srivastava, 1995).

As it turns out, all of the problems explored in this thesis involve selected input variables with at least an approximately unimodal conditional output distribution. This implies that standard connectionist networks work well for the input variables. Because I could not know this in advance, however, experiments presented in this chapter explore ways for determining the number of modes in the conditional output distribution. The main contribution of this work involves extending two mode estimation algorithms to cases where the input space is multidimensional. Up to now, the algorithms explored here were developed only for univariate data.

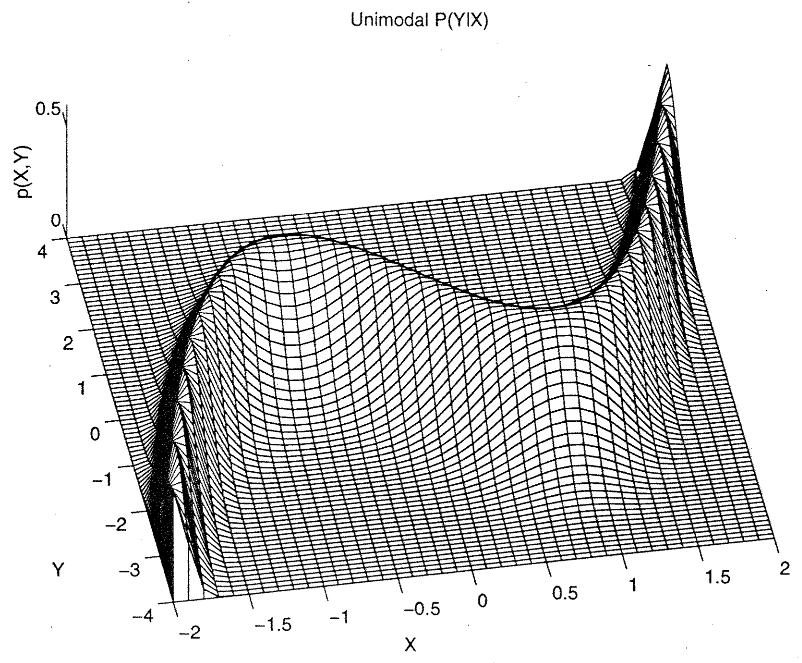
The two algorithms I extend are a kernel based test (Silverman, 1981), and the DIP test for multimodality (Hartigan and Hartigan, 1985). These algorithms and my proposed multivariate extensions are outlined in Sections 6.1 and 6.2. Sections 6.3 and 6.4 present results for the two respective algorithms. The results show that accurate mode estimation is possible in the multivariate case, although accuracy decreases as the number of input variables increases. They also show that it is better to perform initial density estimates with kernel widths selected using likelihood cross validation as opposed to using fixed kernel widths.

6.1 Kernel Based Test

The traditional domain for this test is a set of univariate data. Intuitively, Silverman's kernel based test works by varying the kernel width used in a univariate density estimate to determine how wide the kernels must be to achieve a given number of maxima, or modes, in the distribution. If the kernel widths must be very wide to achieve a unimodal distribution, for example, then there is strong evidence that the supporting data are at least bimodal. This argument can be extended to any number of modes of interest.

The algorithm is described generally as a test for the presence of k modes. Assume that the null hypothesis states that the underlying density has k modes, as opposed to more than k modes. Define the **k-critical window width** h_{crit} as the smallest possible value of the kernel width such that the density estimate has at most k modes; making h any smaller will cause the density to have more than k modes. Large values of h_{crit} will reject the null hypothesis.

The algorithm works by first finding the value of h_{crit} and then using the density estimate with h_{crit} as the kernel width to repeatedly generate "artificial" data sets. For each artificial data set, an "artificial" kernel



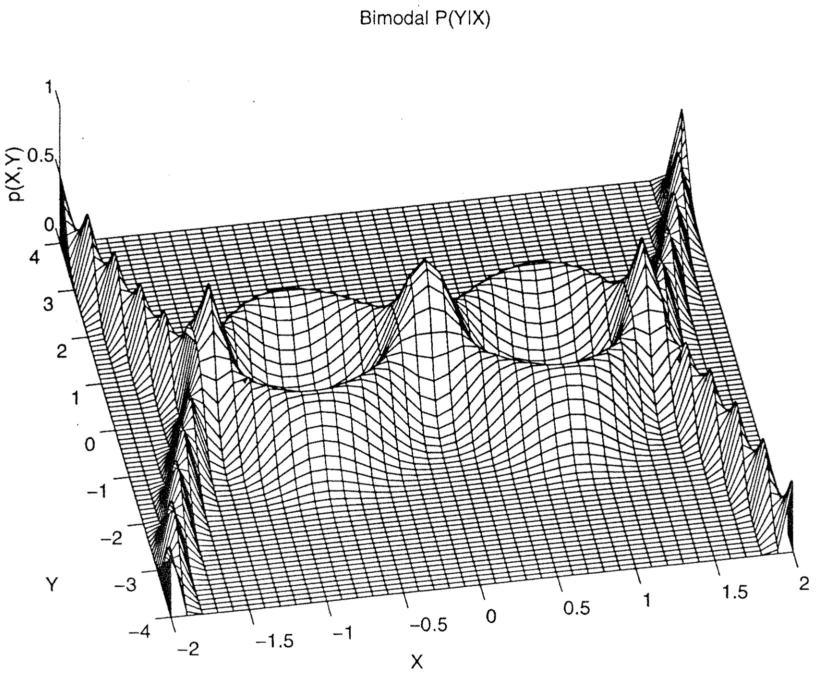


Figure 6.1. Unimodal and bimodal conditional output densities. In the top plot, the distribution $p(Y|\mathbf{X})$ is unimodal. In the bottom plot, $p(Y|\mathbf{X})$ is bimodal.

density estimate is formed, where h_{crit} is once again used as the kernel width. The number of maxima are counted.[1] If h_{crit} is large, the artificial estimates will, in general, have k modes. If, however, the value of h_{crit} somehow reflects the original density estimate well, then chance will cause the artificial estimates to have more than k modes on some occasions. Typically the resampling and mode counting procedure is repeated many times to obtain a significance level for the number of modes.

There are efficient ways to find h_{crit}, although only for certain kernel function choices. The Gaussian kernel is one of the few where one can guarantee that the number of modes in the density estimate cannot increase when the kernels in a density estimate are widened. Consequently, one can find h_{crit} using binary search when Gaussian kernels are employed. With another type of kernel, search for h_{crit} is more difficult and could end up in a local minimum.

Silverman points out that this kernel based method for modality testing does not always give a clear indication of the number of modes. The significance estimates can fluctuate as an increasing number of modes are considered, so that it may appear that the data have, say, either two or four modes with the same degree of significance. Silverman suggests fortifying the statistical test with visual analysis to help alleviate this problem. Needless to say, this hampers efforts to automate deciding the number of modes in the data.

6.1.1 Extending to Multivariate Data There is no simple extension of the method to determining the number of modes in $p(Y|\mathbf{X})$. As the bottom of Figure 6.1 shows, the number of modes in $p(Y|\mathbf{X})$ can vary as a function of $\mathbf{X}$. The best one can estimate is the expected number of modes, where expectation is taken over $\mathbf{X}$.

Toward this end, I extend the algorithm by first estimating the joint density $p(\mathbf{X}, Y)$, then selecting a set of random input values $\{\mathbf{x}_i\}$, and forming a set of estimates for $p(Y|\mathbf{X} = \mathbf{x}_i)$ from the cross section of $p(\mathbf{X}, Y)$ at each $\mathbf{x}_i$. Selecting the input values randomly guarantees that their distribution at least roughly reflects the actual distribution of inputs. I apply the kernel-based test to each of these estimates and histogram all significance estimates together for visual inspection. In order to apply the univariate algorithm to each estimate of $p(Y|\mathbf{X} = \mathbf{x}_i)$, no data need to be generated; the density estimates can be applied directly to the estimates for $p(Y|\mathbf{X} = \mathbf{x}_i)$, and the traditional algorithm's initial univariate density estimation step can be skipped. Weighted Epanechnikov kernels, which comprise each estimate, are replaced with equally weighted Gaussian kernels, the value for h_{crit} is found, and 100 sets of artificial data are created from the resulting estimate. The number of points in each artificial data set is equal to 1000. Results on the 100 artificial data set are used to establish a significance level for the number of modes in the conditional output density estimate.

To make the artificial data sets as similar as possible to the points determining the original density, I rescale each artificial data set to have the same variance as the generating density, as suggested by Silverman. The variance of the original density is calculated approximately by replacing weighted kernels with weighted points and computed a weighted variance. If there are n kernels in the density with univariate locations $\mathbf{x}$ and weights $\mathbf{w}$, where $\sum_i w_i = 1$, the weighted variance is equal to

$$\text{Var} = \sum_{i=1}^{n} w_i \left(\bar{x} - x_i\right)^2 \quad ; \quad \bar{x} = \sum_{i=1}^{n} w_i x_i \ .$$

From the 100 artificial data sets, the number of data sets with more than k modes is tallied. This becomes the estimate of the significance for h_{crit}. The higher this significance level, the more likely the data reflect a distribution with at least k modes. For the purposes of the thesis, comparing the tests for $k = 1$ and $k = 2$ are the most important, since the main reason for mode estimation is to verify that a simple connectionist network is appropriate for a selected subset of input variables.

[1]A practical method for finding the number of maxima in a density estimate involves placing a fine grid of points over the estimate and looking for local maxima according to the points on the grid (Silverman, personal communication). I have found that setting the number of grid points to ten times the number of kernels works well in practice.

6.2 DIP Test for Unimodality

Unlike the kernel based test, the DIP test is designed to answer a more specific question: "Is the set of points sampled from a unimodal distribution?" There is no natural extension of the method to some higher number of modes. It is, however, potentially useful for deciding whether a standard neural network architecture is appropriate for the prediction task.

The traditional starting point for the DIP is, once again, a set of univariate data. The algorithm begins by creating the empirical cumulative distribution function (CDF) of the data. It then uses a procedure to find the stairstep, unimodal distribution that "fits" this empirical distribution the best. The fit of a unimodal distribution is defined as the maximum absolute difference between itself and the empirical distribution on the real line. The best fitting unimodal distribution is the one that attains the smallest fit value (the closest fit). The DIP test statistic is equal to the fit value of the best fitting unimodal distribution. The larger the value of the DIP statistic, the more likely the null hypothesis of unimodal data is rejected. Clearly, the greater the size of the sample, the more accurate the empirical CDF estimate DIP test, and the more accurate the DIP test is likely to be. The authors provide empirical significance levels for the statistic based on 9999 samples from a uniform distribution. The uniform distribution is chosen because in theory, it generates the largest DIP values of all possible unimodal distributions.

The authors provide a Fortran-based version of the algorithm for finding the best fitting unimodal distribution. The algorithm can be obtained through the **Statlib** statistical software repository.[2]

6.2.1 Extending to Multivariate Data Similar to the kernel based method, extending to multivariate data begins with a joint density estimate for $p(\mathbf{X}, Y)$ and a collection of randomly chosen input values $\{\mathbf{x}_i\}$. For each input value, the estimate $p(Y|\mathbf{X} = \mathbf{x}_i)$ is formed. The DIP statistic requires a set of data, so the next step produces a set of 200 points from the conditional output density estimate using smoothed bootstrapping; (Silverman, 1986) discusses an efficient way of generating data from Epanechnikov kernel density estimates. The size of 200 points is chosen because it is the maximum number of points for which empirical significance levels are reported in (Hartigan and Hartigan, 1985). From these data, the DIP statistic is computed. The DIP statistics for all inputs are then histogrammed to get a sense of the overall modality of the conditional output distribution $p(Y|\mathbf{X})$.

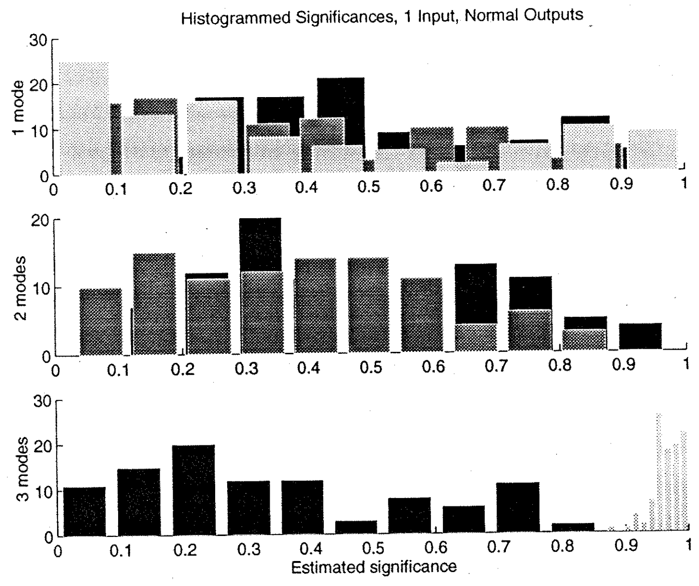
6.3 Results with Fixed Kernel Width Density Estimation

The extended algorithms are tested with six artificial data sets. Each data set consists of 1000 points, and included five input variables and one output variable. The inputs are all i.i.d. uniform, and to simplify the analysis, the conditional output density function is unvarying across the entire input space. This test may seem overly simple at first, but the actual relationship between input and output variables is not as important as the form of the conditional output density, since the tests are based solely on estimated conditional output densities for different input values.

The output variables for the first three data sets have density functions $N(1, 0.1)$, $\frac{1}{2}N(1, 0.1) + \frac{1}{2}N(2, 0.1)$, and $\frac{1}{3}N(1, 0.1) + \frac{1}{3}N(2, 0.1) + \frac{1}{3}N(3, 0.1)$, respectively. Here, $N(\mu, \sigma)$ indicates a normal distribution with mean μ and standard deviation σ. The output variables for the other three data sets have density functions $U(1, 1.5)$, $\frac{1}{2}U(1, 1.5) + \frac{1}{2}U(2, 2.5)$, and $\frac{1}{3}U(1, 1.5) + \frac{1}{3}U(2, 2.5) + \frac{1}{3}U(3, 3.5)$, respectively; $U(a, b)$ indicates a uniform distribution with starting point a and ending point b. The data for all input and output variables are rescaled to have mean zero and variance one. The choice of between one and five input variables and one of six output variables creates a set of (5×6) problems over which to apply the modality estimation methods. Because of the constant output functions, the only thing that varies by changing the number of input variables is the density of data points involved in each test of the modality for the conditional output distribution.

As a first try, I chose to base all joint kernel density estimates on a fixed kernel width of 0.2. The

[2] `http://lib.stat.cmu.edu`



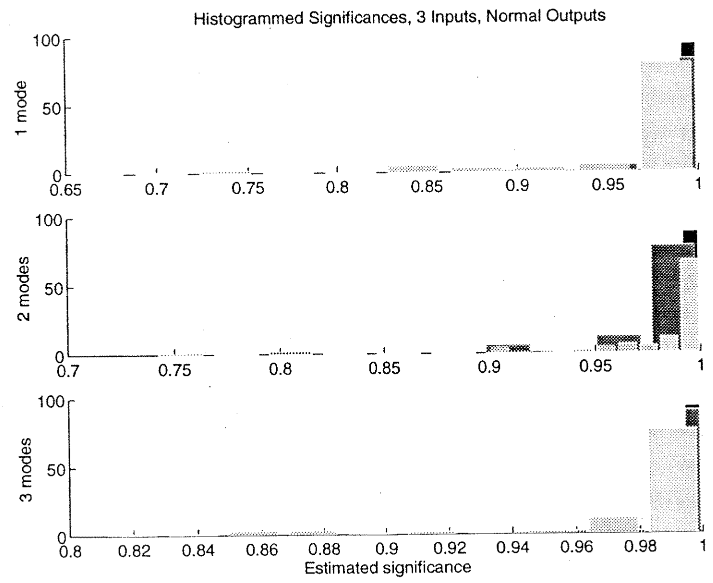


Figure 6.2. Kernel based test results, fixed kernel widths. Kernel widths for all density estimates were fixed to 0.2. Histograms show significance levels computed for estimated h_{crit} values; the lightest histogram shows values for a unimodal test, and the darkest shows values for a trimodal test.

original thinking behind this choice was that the number of inputs should be the experiment's only independent variable, and all other experimental variables, such as the kernel width involved in the density estimate, should remain fixed.

6.3.1 Kernel Based Test Figure 6.3 shows results for the kernel based test on the first three data sets (those with output variables derived from normal mixtures). The y axis label of each plot shows the number of modes that the output variable had for that plot. The top group of plots shows histogrammed significance values when one input variable is involved in the joint density estimate, and the bottom group shows significance values when three input variables are involved. Three superimposed histograms in each plot show significance levels obtained for three separate values of k, the number of modes being tested for. The lightest shaded histogram denotes significance levels for a test for one mode, the darker shaded histogram denotes levels for a test for two modes, and the black histogram denotes levels for a test for three modes.

Focusing on the top group of plots, one sees that when the output variable is in fact unimodal, significance levels for all three tests are roughly uniformly distributed between zero and one. When the output variable is bimodal, tests for $k = 2$ and $k = 3$ are still roughly uniform, whereas the test for $k = 1$ results in all significance values equal to one (no corresponding histogram was shown for this reason). When the output variable is trimodal, the significance values for $k = 3$ are roughly uniformly distributed, all values for the $k = 2$ test become one, and values for the $k = 1$ test are distributed between 0.9 and 1.0. From this, one can infer a general rule that when significance levels for a test $k = m$ are high, there are at least m modes present in the data.

Focusing on the bottom group of plots, which involve three input variables in each joint density instead of one, it is obvious that the power of the tests to distinguish the number of modes has deteriorated. Significance levels for all three tests in each case are distributed roughly equally in the range $[0.95, 1.0]$, with a few smaller outliers for the test for $k = 1$. The mean number of kernels involved in each conditional density estimate dropped from a mean of 116 in the one input variable case to a mean of 3.5 in the three input variable case. Clearly, the power of the test depends on the number of kernels involved in each conditional density estimate, which drops off exponentially with the number of input variables if the kernel widths are fixed. A joint density estimate with a high number of input variables must, in general, involve wider kernel widths than estimates involving just a single input variable, in order to keep the number of kernels involved in each conditional density estimate roughly constant as the number of input variables vary.

6.3.2 DIP test Results on the DIP test for unimodality show similar behavior across input variables (Figure 6.3). Plots here are organized differently from Figure 6.2; the top group shows results for data sets with the three normally mixed output variables, and the bottom group shows results for data sets with the three uniformly mixed output variables. A single plot summarizes tests for a given number of input variables (shown by the y axis label), and each histogram represents a test for a certain data set. The lightly shaded histograms show test results for unimodal output variables, through to the black histogram, which shows test results for trimodal output variables. The vertical dotted lines show where 0.9 significance levels for the test lie: for data generated from a uniform distribution, a dip test's value will fall to the left of this line 90 percent of the time.[3] Significance values shown in the plots are based on setting the sample size equal to the number of kernels used in the conditional density estimate. These significance values can be used as a basis for a unimodality test: If a computed dip statistic lies to the right of a dotted line, then we can reject the null hypothesis of unimodality with a confidence of 0.90.

There are several important observations based on the plots. When one input variable is involved (the first and fourth plots from the top), the test succeeds in rejecting the unimodality hypothesis for the bimodal and trimodal output variables. For the unimodal, normal output variable, roughly ten percent of the

[3] Each significance level value is reported in (Hartigan and Hartigan, 1985) and is based on empirical tests run over dip calculations on 9999 uniformly distributed samples.

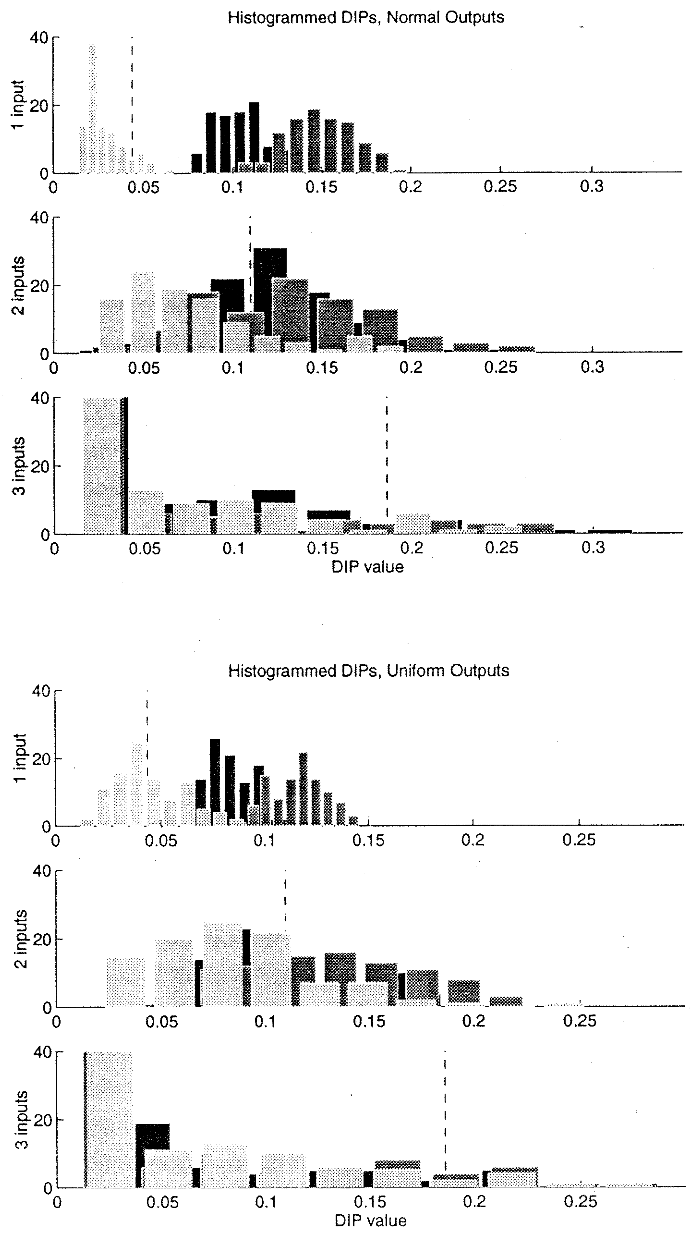


Figure 6.3. First DIP test results, fixed kernel widths. Kernel widths for all density estimates were fixed to 0.2. Histograms show how 100 DIP calculations were distributed for each test; the lightest histograms show DIP estimates for unimodal output variables, and the darkest show values for trimodal output variables.

dip calculations lie to the right of the significance line, as expected.

Also, for the unimodal, uniform output variable, slightly more than fifty percent of the dip calculations lie to the right of the line. This is unexpected, but it may be due to the use of kernel based, smoothed bootstrap calculations to generate samples for the DIP test. It may be necessary to place less weight with the significance levels found in (Hartigan and Hartigan, 1985) when smoothed bootstrap samples are employed. Also, the difference in results between the normal and uniform output variable results suggests that unimodality is more easily recognized in normal random variables than in uniform random variables.

Also, there appears to be a weakness in the way the statistic is computed, which is evident in results for the trimodal distributions. For trimodal data, the histogram of dip values overlaps significantly with the dip values of unimodal data, making it hard to determine if the data is truly unimodal or not. This is due to the fact that the "valleys" in the trimodal distribution are shallower than for the bimodal density; both density functions have modes with roughly the same width, but not the same height. Hence, the existence of multiple modes is harder to detect in the trimodal case when the DIP statistic is used.

Finally, the deterioration in the power of the tests is again evident as the number of input variables increases. For this reason, I reran the tests and allowed the kernel widths for each joint density estimate to be selected automatically through likelihood cross validation. The results for these tests are presented in the next section.

6.4 Results with Variable Kernel Width Density Estimation

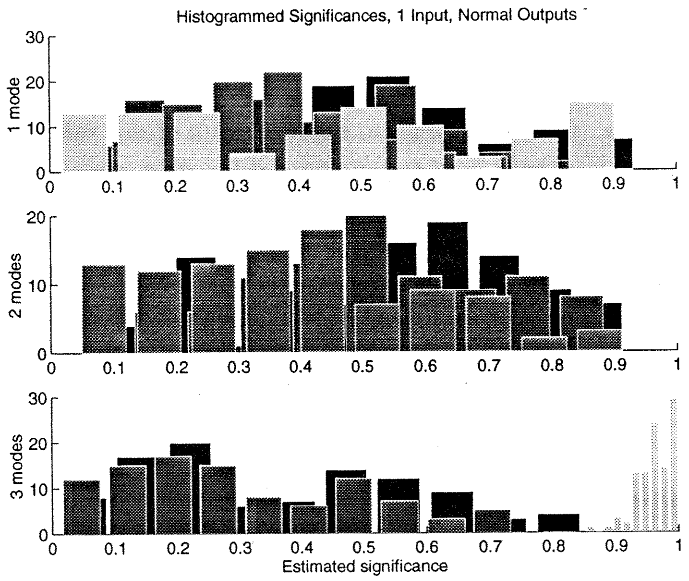
Because the power of the modality tests depend so strongly on how many kernels span each conditional density estimate, I allowed the kernel widths for the density estimate $p(\mathbf{X}, Y)$ to be determined automatically through likelihood cross validation (Section 2.2). This procedure for kernel width selection not only matches the one used for all prediction problems in this thesis, but it also tends to make kernels wider when more input variables are involved in a kernel density estimate. Hence, the number of kernels involved in a conditional output distribution calculation should remain closer to constant as more input variables are considered.

6.4.1 Kernel based test Figure 6.4 shows the results for the kernel based test. For completeness, four histograms are superimposed in each plot; the histograms for $k = 1$ to $k = 4$ are shaded by colors that become darker for larger values of k. The top group of plots look much the same as in Figure 6.2. The bottom group, however, where five input variables were involved in the joint density estimate, look much better than in Figure6.2. In the case of a trimodal output variable (bottom plot), test values for $k = 3$ and $k = 4$ are roughly uniformly distributed in the range $[0, 1]$, whereas values for $k = 1$ are distributed above 0.6, and values for $k = 2$ are distributed above 0.9. It appears that the deterioration in estimated significance values decreases only slightly as a function of the number of input variables when kernel widths are chosen through likelihood cross validation.

6.4.2 DIP test Figure 6.5 shows the results of the DIP test when kernel widths are chosen through likelihood cross validation. Once again, the 0.90 confidence levels are indicated with vertical dashed lines, where the levels are based on the mean number of kernels involved in each conditional density estimate.[4]. As the number of input variables increases, the test has a harder time distinguishing between unimodal and multimodal data. Progressively wider kernels apparently smooth out multiple modes to the point that they cannot be easily discerned with the DIP test.

As a final attempt to increase the power of the DIP test, I tried sampling points for the artificial data sets using straight bootstrapping, rather than smoothed bootstrapping, on the theory that the presence of modes would be more pronounced. Figure 6.6 shows the results. The data appear less unimodal, but there is still no clear separation between results for unimodal and trimodal output variables, and significance values do

[4]The number of kernels actually decreases for the data sets with bimodal and trimodal output variables, which pushes the 0.9 significance values even further to the right on the plots for the darker histograms. This only worsens the already poor results for three and five input variable cases



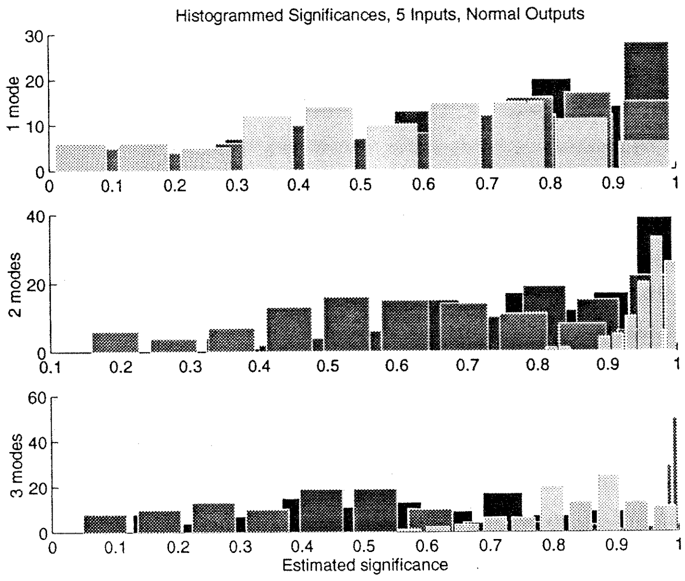


Figure 6.4. Kernel based modality estimation results. Kernel widths for the joint density $p(\mathbf{X}, Y)$ are chosen through likelihood cross validation.

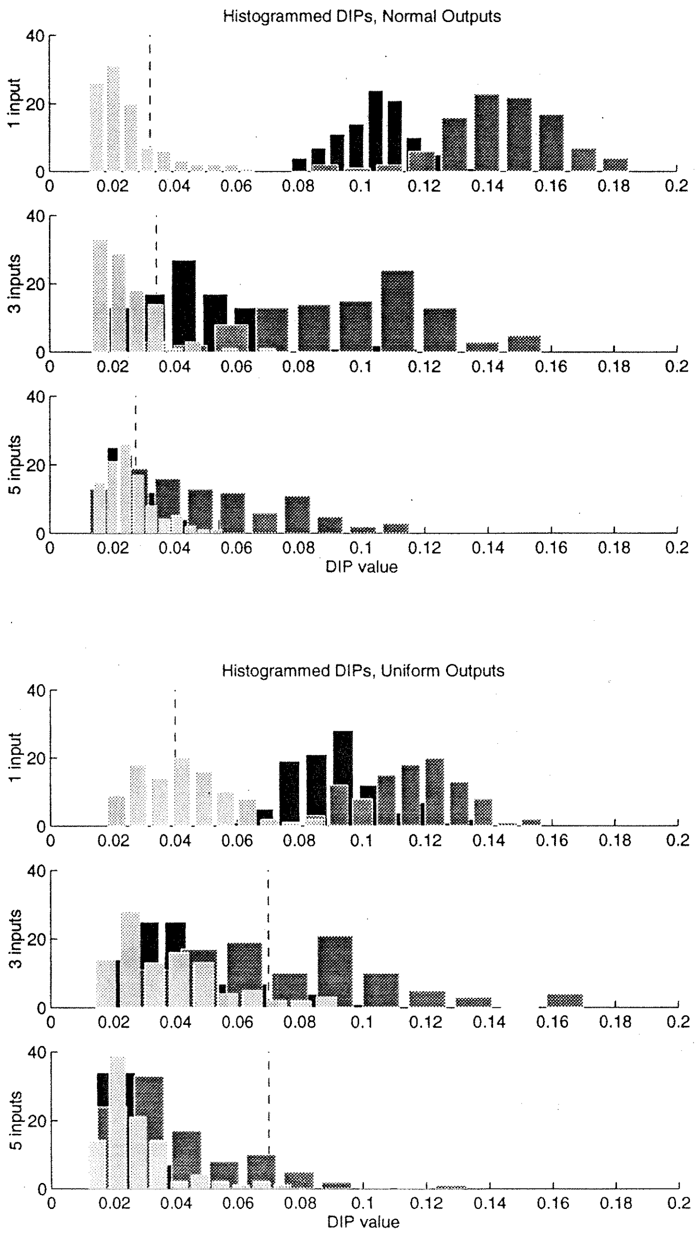
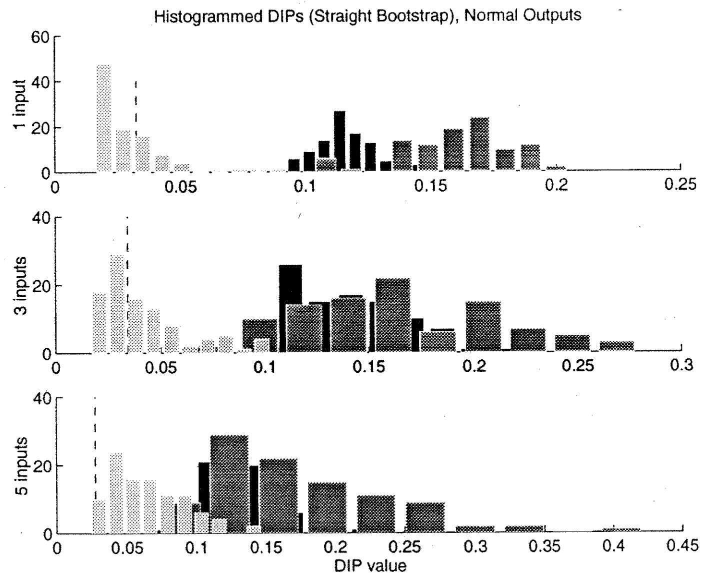


Figure 6.5: DIP test results with variable kernel widths.



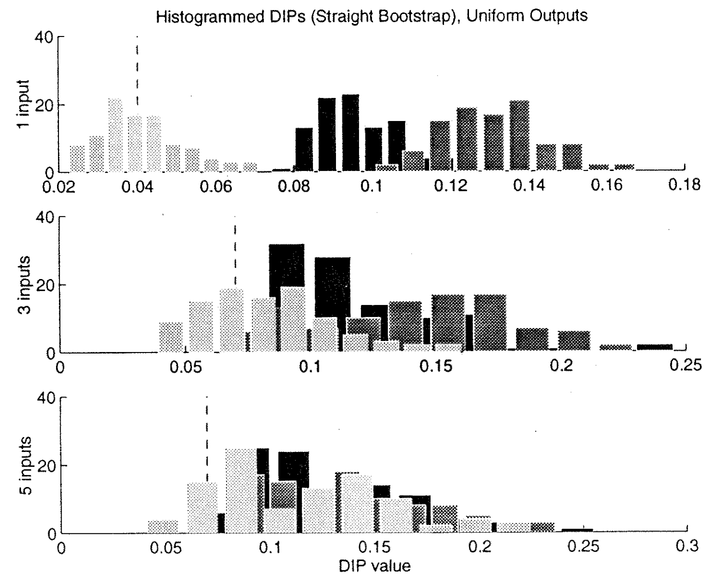


Figure 6.6. DIP test results with straight bootstrapping. Previous plots have shown results where smoothed bootstrapping was used to obtain data points for significance tests.

not appear to correspond: the null hypothesis based on the 0.9 confidence levels would likely be rejected even for the unimodal output data, as the number of input variables grows.

6.5 Conclusions

From the experiments performed in this chapter, I conclude that Silverman's kernel based test would work best for determining the number of modes in the conditional output distribution, especially when more than two or three input variables are involved. This conclusion is based on well-behaved data, however; it may be possible to defeat the test in cases where the data are not as well-behaved. One clear problem occurs when the number of modes in the conditional output distribution varies across the input space. In this case, my approach to Silverman's test will give no clear indication of the maximum number of modes that appear. In order to accomplish this, it would be necessary to perform a series of tests that more systematically cover separate parts of the input space.

CHAPTER 7

COMPARISON OF INPUT VARIABLE SELECTION METHODS

This chapter compares three methods for input variable selection on a range of prediction problems. The methods include the one proposed in this thesis, SBP, and ARD. The goal of the comparisons is to show for which problems a particular method is best suited.

Results in this chapter show that the proposed method is most appropriate when there are many possible input variables, and only a small handful are actually relevant for prediction. The other two methods are more appropriate for problems with small data sets or problems where a larger number of inputs variables are relevant. Figure 7.1 depicts the appropriateness of the methods as regions in a three-dimensional space. The axes for the space are the size of the data set, the number of independent input variables, and the number of relevant input variables. Note that one axis is the number of **independent** input variables, rather than the number of **possible** input variables. The distinction between independent variables and simply all possible input variables is important: the underlying dimensionality of the points is more essential to determining the difficulty of input variable selection than the actual number of variables used to express it. Note also that the regions in the figure intersect when both the number of possible input variables and the number of relevant ones are small; a few of the real-world problems studied here fall within this intersection, and all methods work well in selecting relevant input variables.

The comparisons begin with an artificial prediction problem as a way of confirming the validity of Figure 7.1. Analysis continues with comparisons on the artificial problems proposed by (Finnoff et al., 1993), and finish with real-world prediction problems. Results for the problems proposed by Finnoff et al. suggest that they are not appropriate for comparing different input variable selection methods. The real world problems include prediction of energy consumption in a building, prediction of the heart rate in a patient with sleep apnea, and prediction of the wind's angle of attack in a wind turbine. Results on the real world problems support results on the first artificial problem, and they suggest additional strengths and weaknesses of the input variable selection methods.

7.1 "Fair" Method Implementations

Throughout this chapter, I attempt to choose implementation parameters that make comparisons of the methods as fair as possible. For example, the number of hidden units are fixed across methods for every prediction problem. However, a few implementation differences makes straightforward comparison of the methods difficult. For example, Neal's completely Bayesian framework and inclusion of priors over all network parameters, including the hidden unit to output unit weights, tends to improve the performance of ARD over its two competitors, which employ more straightforward, less powerful, connectionist assumptions. Early stopping, which is used for the methods competing with ARD, also biases more toward linear solutions than ARD. For this reason, the ARD network architecture also includes direct, input to output weights to help compensate for this bias. To make the comparison easier on certain problems, some of Neal's Bayesian machinery is used in place of standard connectionist learning.

Another difficulty involves deciding a fair computational limit for raw and adjusted mutual information estimates. These estimates allow the freedom of **bootstrapping**,[1] which results in more accurate mutual information estimates at the expense of additional computation time. Instead of basing the relevance of an

[1] Bootstrapping involves creating a collection of data sets from the original by resampling the observations randomly with replacement.

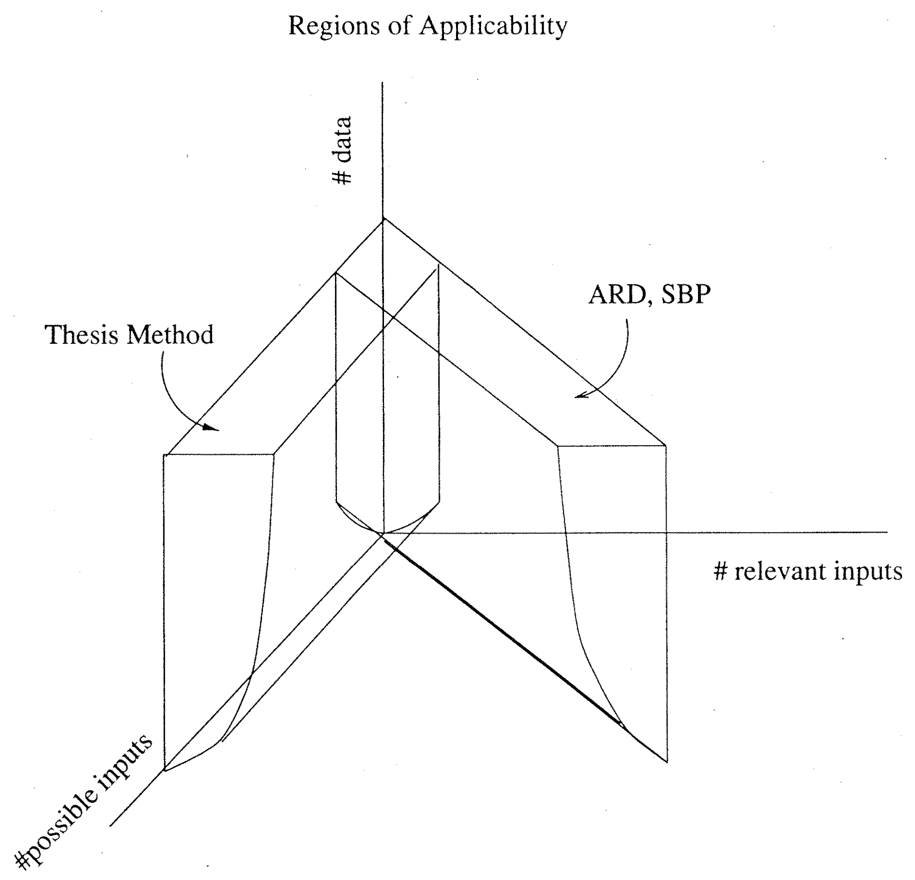


Figure 7.1: Regions of applicability for three input selection methods.

input variable on a single raw or adjusted estimate, the data can be bootstrapped m times, and relevance can be based on a function such as the average of the m mutual information estimates. For most problems, forward selection without bootstrapping is at least ten times faster than SBP or ARD, so I often base relevance on ten bootstrapped, raw estimates or five bootstrapped, adjusted estimates.[2]

Typically, ARD is run for 100 iterations on each problem, and evenly spaced networks are taken from the last third of the iterations, in order to reflect independent networks with high posterior probability (Neal, personal communication). These are compared against networks using input variables derived from raw and adjusted mutual information estimates. Performance is reported for ten networks with different initial weights. For the SBP algorithm, performance is reported only for a single network, since results for SBP are supposedly valid only for a particular set of weights (Section 3.1.3). To properly get an idea of how SBP operates across ten different choices for initial weights, one would have to run the entire SBP algorithm ten times, which is computationally prohibitive.

The basis of comparison between methods will be Normalized, Mean Squared Error (NMSE), defined as

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

Here, summations are taken over the n available points in the test set: the y_i represent test set output values, the $\hat{y}_i$ represent network output values, and $\bar{y}$ represents the mean of the test set output values. NMSE is simply the mean squared error on the test set divided by the variance of the test set outputs. What NMSE expresses is a ratio of the variance of predicted output values versus the variance of the actual output values. If this ratio is close to 1.0, then the usefulness of a network's output values can be said to be low; one could simply produce the mean of the test set output values $\bar{y}$ on all test examples and achieve a comparable mean squared error. If this ratio is lower, however, then one knows that the network is extracting structure from the problem to produce good output values on the test set.

7.2 Artificial Problem

This problem serves as a testbed for exploring the space of data set characteristics depicted in Figure 7.1. In keeping with the figure, the characteristics varied in this problem include the amount of data, the number of possible, independent input variables, and the number of relevant input variables. The result is an ensemble of related prediction problems, to which each input variable selection method is applied.

I describe an initial prediction problem first, from which the variations in the ensemble are derived. The initial problem's characteristics are chosen so that it falls within the **intersection** of the two regions in Figure 7.1. In other words, it contains a large amount of data, the number of relevant input variables is small, and the number of possible, independent input variables is also small. All methods should perform well on this problem. The initial problem contains twelve possible input variables $\{X_1, \ldots, X_{12}\}$, all i.i.d. normal with mean zero and variance one. The output variable Y is equal to

$$Y = 0.7X_1 + 0.3X_2 + 0.1X_3 + \sin(4X_4 + 2X_5 + X_6) + \epsilon .$$

The random variable ϵ is uniform with mean zero and variance equal to the sample variance of the first four terms of Y. The data set size is 5000, which is split evenly into training and test sets.

From this initial problem, the variations are derived. The variation with less data has a data set size of 1000 instead of 5000. The variation with more independent input variables has 40 irrelevant, i.i.d. inputs instead of six. The variation with more relevant input variables has output variable Y equal to

$$Y = 0.7X_1 + 0.3X_2 + 0.3X_3 + 0.1X_4 + \sin(4X_5 + 2X_6 + 2X_7 + X_8) + \epsilon ,$$

[2]Less computation is given to adjusted estimates because each represents the difference of two estimates (Chapter 5).

where a third more input variables are relevant. The number of irrelevant input variables remains the same, increasing the total number of possible input variables to 14. Again, ϵ for this latter problem is uniform with mean zero and variance equal to the sample variance of the sum of all relevant terms for Y. A final variation involves a large number of possible input variables, but these inputs are no longer independent. For each relevant input variable X_i in the initial problem, there are six equal to $X_i + N(0, 0.5)$. This creates 36 extra input variables, each with a correlation of around 0.90 to some relevant input variable. An extra four independent input variables are added to bring the total number of irrelevant input variables to 40, as with the second variation described above. For the initial problem and its four variations, the data for all input and output variables are rescaled to have sample mean zero and sample variance one.

Before getting to the results, it is necessary to mention a few implementation details. The connectionist networks for ARD and SBP use 20 tanh hidden units. Networks trained using inputs selected from raw and adjusted estimates also employ 20 hidden units. Raw mutual information estimates are based on ten bootstraps, and adjusted estimates are based on five.

Results for the initial problem and its four variations are presented in Figure 7.2. The title of each plot identifies the variation. Along with the ARD, SBP, raw and adjusted performances, a baseline estimate is also presented. The baseline reflects performance using standard connectionist learning where all possible input variables are involved. The top left plot in the figure shows that on the initial problem, ARD, SBP, and the raw estimate input variables all perform better than the baseline. One learning run for the raw estimate gets stuck in a local minimum; this is due to using initial weights that are close to zero, which biases the learning algorithm too heavily toward picking up the linear components of the problem. Rerunning the simulations using initial weights selected from a mean zero, normal distribution with standard deviation 0.5 eliminates the local minimum and results in otherwise indistinguishable performance values. The adjusted estimate does no better than baseline because noise in the output and a medium sized data set conspire to prevent the selection of more than three relevant input variables (Section 5.2). Raw estimates are less affected by this circumstance but still are able to select only five of six relevant input variables. The input variables selected by SBP, raw, and adjusted estimates are $\{X_1, X_4, X_5, X_6, X_2, X_{10}, X_3\}$, $\{X_1, X_6, X_2, X_4, X_5\}$, and $\{X_1, X_2, X_8\}$, respectively.[3]

ARD appears to have the best performance, but some of this is due to additional priors over weights connected to the network's output unit. To confirm that the additional priors are actually an enhancement, I used Neal's learning implementation on the input variables selected with raw estimation, where priors are placed only over the weights connecting to the output unit. The slightly better test set performance appears in the plot under the label "Raw+".

The ARD algorithm represents the selection of an input variable as a continuous value, so a separation between selected and unselected input variables is not immediately clear. Figure 7.3 shows the hyperparameter values for the last twenty training iterations. In this case, training is performed for 50 iterations, instead of the normal 100, because convergence to a good solution happens so quickly. The high values given to variables $\{X_4, X_5, X_6\}$ indicate that these input variables are selected. The linear input variables $\{X_1, X_2, X_3\}$ have low associated hyperparameter values because direct input to output unit weights exist; these input variables influence the output variable by modifying direct weights to the output variable, rather than those that pass through the hidden units. In general, an input variable will be considered selected by ARD if associated hyperparameter values or direct input to output weights have a large enough relative magnitude (greater than ten percent of the largest hyperparameter value) in the last twenty iterations of learning. The relative magnitude is necessary because absolute hyperparameter magnitudes can vary from problem to problem. The choice of ten percent is somewhat arbitrary, but it appears accurate in separating relevant and irrelevant input variables for the artificial problems tested in the thesis.

[3]The ordering given for input variables reflects their order of relevance: the first variable listed is either the first selected, in the case of forward selection, or the last eliminated, in the case of SBP's backward elimination.

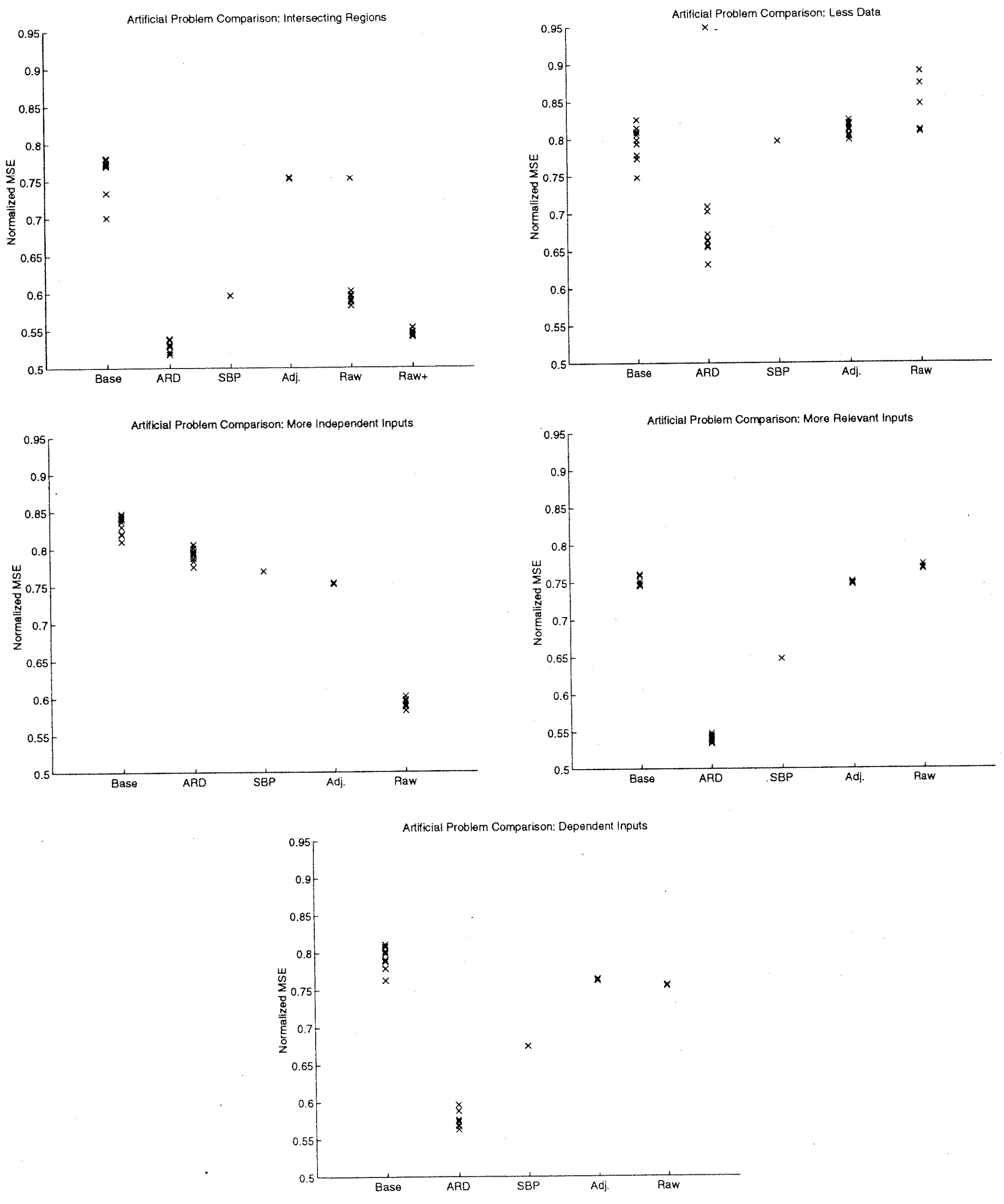


Figure 7.2. Comparative results for the artificial problem. For each method except SBP, performance is shown for ten networks.

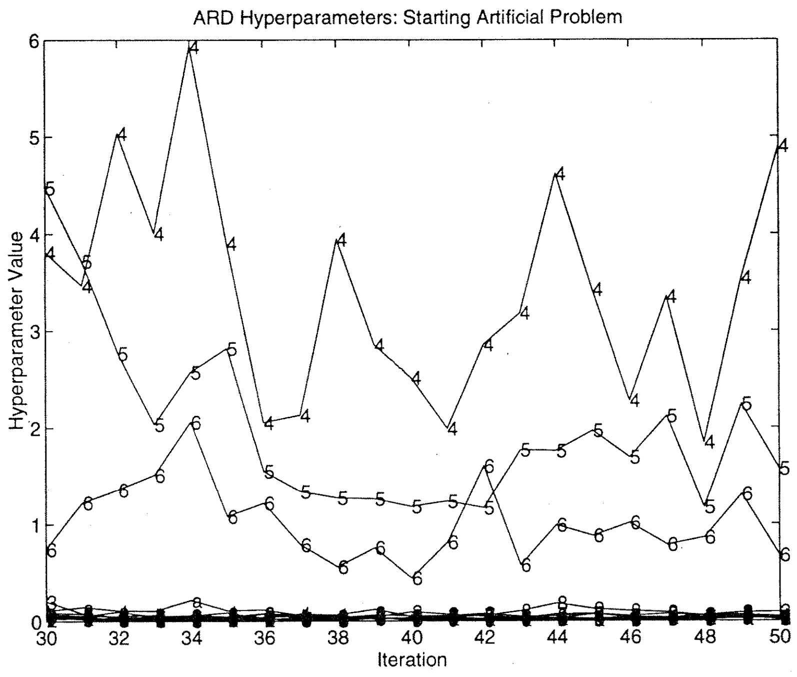


Figure 7.3. Example of ARD hyperparameter values during learning. High values given to the three nonlinear input variables indicate that they are selected. The linear three linear input variables appear unselected, but they still have an affect on the output unit through direct input to output unit connections in the network.

The top right plot shows results for the smaller data sets; for this plot, combined training and test set sizes total 1000 observations. All input variable selection methods perform worse, as one would expect. The ARD algorithm still does somewhat better than the other methods, but a single, high data point for ARD also indicates that its performance has a high variance. The other methods perform no better than the baseline. Raw and adjusted estimation selected the input variables $\{X_1, X_4\}$ and $\{X_1\}$, respectively. Technically, SBP selected all possible input variables but one irrelevant one, although validation performance does not worsen dramatically until ten input variables are removed (Figure 7.4), suggesting that it found the last three input variables to be removed ($\{X_1, X_2, X_4\}$) the most important of the ones it did select. For ARD, the hyperparameters look much like those for Figure 7.3, but the magnitudes and variability from one iteration to another are much larger. It is fair to say that ARD may separate relevant from irrelevant input variables correctly, but this does not help it achieve the same performance it can with more data.

The left middle plot of Figure 7.2 shows results for more independent input variables. Here, the raw and adjusted estimates are roughly the same as in the top left plot, but performance for ARD and SBP have worsened. The presence of the additional input variables has made it difficult for either SBP or ARD to pick out structure in the data, which is a necessary precursor to their training based removal of irrelevant input variables. ARD selects the linear input variables X_1 and X_2 through its direct input to output connections, but the largest hyperparameters in the last twenty iterations are associated with irrelevant input variables. The training algorithm was run for 200 iterations to make sure that this would not change through more training. The behavior of the SBP algorithm is presented in Figure 7.5.

Raw and adjusted estimation selected the same input variables as for the original artificial problem with only 12 possible input variables. However, each relevance estimate for an input variable subset was based on 60 raw or adjusted estimates, instead of just 5 or 10. The increased computation is necessary for the following reason: even totally unimportant input variables have a positive empirical relevance associated with them, which occasionally, by chance, is as large as the relevance estimates for truly important input variables. Because each relevance estimate has a certain empirical variance associated with it, more computation is needed to guarantee that a truly important input variable has an average empirical relevance larger than all other possible input variables.

The right middle plot of Figure 7.2 shows results for two additional relevant input variables. Here, ARD performance is nearly identical to the top left plot. The barely noticeable worsening of performance is probably explained by have a slightly greater number of network weights for the same amount of training data. SBP performance is also slightly worsened, with the selected input variables $\{X_1, X_5, X_6, X_7, X_8, X_2, X_3, X_{10}, X_{13}, X_{12}\}$, listed from last eliminated to first. The slightly worse performance may be due to the presence of three irrelevant input variables in the group, versus only one in the original problem.

Raw and adjusted estimates select the input variables $\{X_1, X_3, X_{10}\}$ and $\{X_1, X_2\}$, respectively. Mean performance for both cases is equal to the baseline, with a small variance. In the case of the raw estimation method, there is no longer enough data to identify the two additional input variables that are relevant.

The bottom plot shows results for the case where there are many irrelevant input variables, but they contain redundant information. The ARD and SBP algorithm appear only slightly affected by the presence of these extra input variables; their performance appears much better than when the inputs are all independent, as they are in the left middle plot. As with the other plots, ARD appears a little more resistant to extra input variables than SBP. The SBP algorithm selects 21 input variables, which is not necessarily poor behavior; there is likely very little negative effect on test set performance associated with including many highly redundant input variables.

Performance based on raw estimates, however, has deteriorated once again to the baseline. This reflects another weakness of raw mutual information estimation: it is susceptible to selecting redundant input variables (Section 5.1). The variables selected by raw estimation, where dependent inputs are mapped back to their

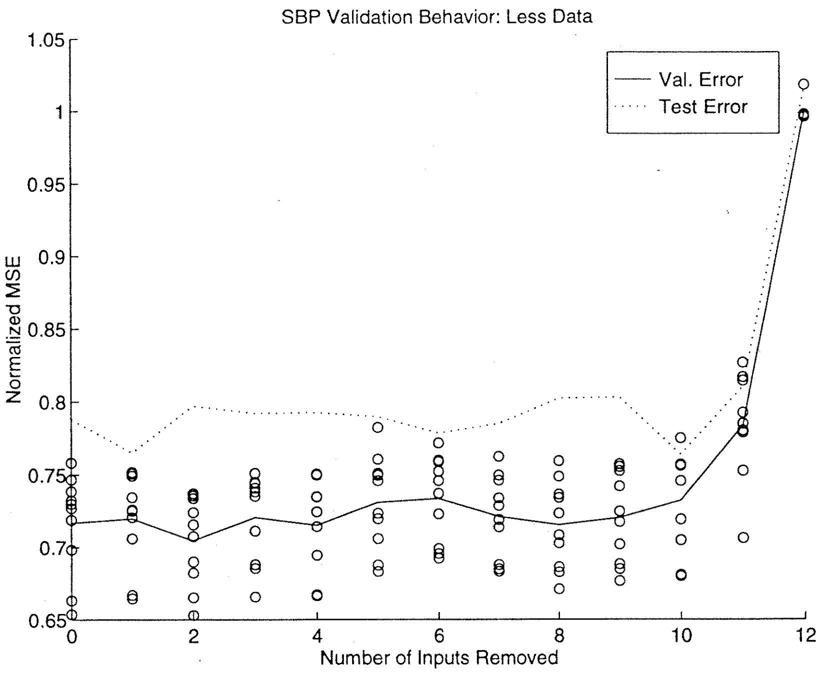


Figure 7.4. SBP validation behavior on the variation with less data. Circles represent cross validation performance on individual networks; the solid line represents the average. Validation and test set performances are relatively constant until more than ten input variables are removed.

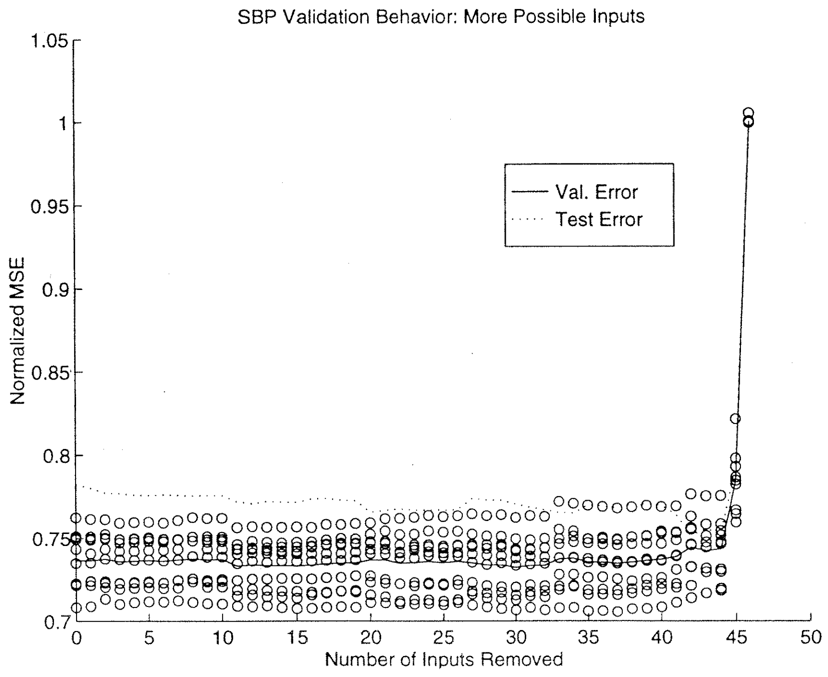


Figure 7.5. SBP validation behavior on the variation with more possible input variables. Validation and test set performances never dip below 0.7 because relevant input variables are removed in the early stages, before enough input variables have been removed to distinguish good from bad.

Label	Structure	Train	XV	Test	Noise Variance
linid	$\sum_{k=1}^{5} \alpha_k x_k$	200	100	1000	$\{0.7, 0.8, 0.9\}$
linsin	$\sum_{k=1}^{5} \sin(\alpha_k x_k)$	400	200	1000	$\{0.0, 0.3, 0.6\}$
prodid	$\prod_{k=1}^{3} x_k$	1400	600	1000	$\{0.0, 0.1, 0.2\}$
prodsin	$\prod_{k=1}^{3} \sin(\alpha_k x_k)$	1400	600	1000	$\{0.0, 0.1, 0.2\}$
rbfnet	Sum of RBF's	400	200	1000	$\{0.0, 0.3, 0.6\}$

Table 7.1: Summary of problems explored.

originals, are are $\{X_1, X_5, X_4, X_1, X_6, X_2\}$, with only the first input variable in the set being noise free. The input variables selected by adjusted estimation are $\{X_1, X_4\}$, again with only the first variable being noise free.

7.3 Artificial Problems of Finnoff et al.

The prediction problems presented in (Finnoff et al., 1993) are also an interesting subject of study. Initially, it seemed that these problems would be ideal for comparing input variable selection methods, since the problems had already appeared in the literature, were the subject of intense study, and claimed to be appropriate for comparing weight pruning algorithms. However, I soon found that they are not ideal for comparing input variable selection methods, for a few reasons. First, the data sets tend to be too small for good raw or adjusted mutual information estimation. More importantly, however, results show that performance for several of the problems does not worsen significantly in the case where all input variables are included over the case where only relevant input variables are included. For at least one problem, the target output is inherently difficult to learn, which makes its usefulness as a basis for comparison questionable.

Table 7.1 shows the characteristics of each problem. It includes a label, the general form of the output function, the size of the training, validation, and test sets, and the variance of additive output noise given to three different instantiations of a problem. More detailed descriptions of the input and output variables for each problem can be found in Section 5.2. Each problem instantiation has six associated data sets, created in order to reduce the effect of any anomalies in a particular data set. Results for each method reflect ten networks run on a data set. All networks for each method involve ten tanh hidden units and a linear output unit. Results for the classification problem are obtained by thresholding the linear output at zero.

Figure 7.6 depicts results for the five prediction problems listed in Table 7.1. Each error bar reflects runs over 60 networks. A few things things should be pointed out. First, error bars for the two cases where all input variables and only relevant input variables are present overlap. Because of the overlap, one can not conclude that performance is significantly enhanced by removing irrelevant input variables, at least without performing many more runs. The outcome of overlapping error bars is not that surprising, given that all problems involve such a small number of irrelevant input variables. The number of irrelevant input variables is not really large enough to show a significant difference in network performance when they are included, without requiring possibly thousands of training runs.

The second fact to be pointed out is that performance on the rbfnet problem (bottom plot) is poor for all input selection methods. This problem is simply inherently difficult to learn; most target output values are close to zero, except for between ten and fifty large outliers. It seems that one should think carefully before reporting any kind of comparative results for a problem such as this.

Lastly, note the slightly worse ARD performance on the linid problem. This is just a confirmation that ARD is less biased toward linear solutions than the other methods, which rely on the method of "early stopping". Such a bias occurs whether or not direct input to output connections exist for the network, which suggests that the bias away from linearity may be due to a difference in learning algorithm implementation, either in the network's initial weights or the optimization technique itself.

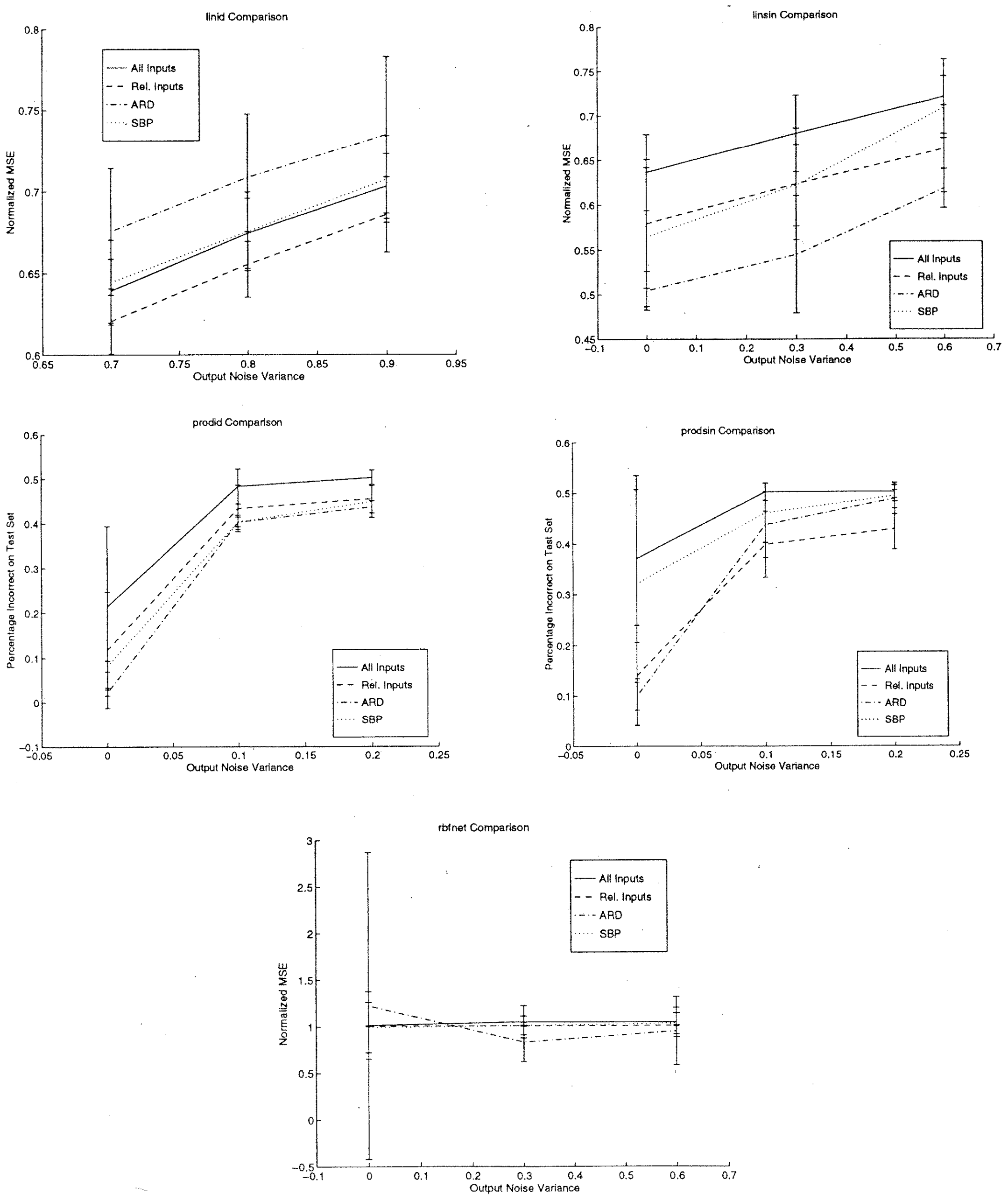


Figure 7.6: Comparative results for Finnoff et al.'s prediction problems.

In order to make performance differences more significant for these problems, without requiring a lot of computation time, two changes may be necessary. First, the number of irrelevant input variables should probably be increased. Second, some of the problems or network architectures will have to be modified to allow for a greater possibility of overfitting. In particular, the linsin and prodsin prediction problems have a complicated target output function that no network with only ten hidden units can hope to approximate. Without the possibility of overfitting, a competition for network resources between relevant and irrelevant input variables will be less likely to occur, and the variance of network error across training runs will not be as great as the case when overfitting is possible.

7.4 Energy Consumption Data

The real world data for prediction represent information gathered about a large, academic building located on the Texas A&M University campus. Sensors measuring energy consumption and related indicators were sampled on an hourly basis, resulting in eight time series with around 4200 examples each. Data from September 1, 1989 through December 31, 1989 comprise the training set, and the test set consists of data from the next 54 days. The available input series are

- hour of the day,
- outside temperature,
- outside humidity,
- solar flux,
- wind speed.

The target output series are:

- whole building electricity,
- chilled water usage,
- hot water usage.

I treat each output series as a separate prediction problem. I create a set of 26 possible input variables by using the six most recent time lags for the temperature, humidity, flux, and wind variables, resulting in 24 possible input variables. I rescale the "hour of the day" variable to be between 0 and 2π and take the sine and cosine, in order to obtain a circular representation of the hour of the day. This accounts for the other two input variables. Labels for the input variables, which are reported below, are numbered from 1 to 26. They are assigned by considering the sets of variables in the order of time, temperature, humidity, flux, and wind. Within each set of variables, the most recent time lag is given the lowest label number.

My approach to input variable creation for this data set is different from MacKay's (1993) , who removes anamalous data from the training set (including the period over Chrismas and New Year's break, most clearly apparent in the top series of Figure 7.7), designs a different set of input variables with time lags taken at 2.5, 24 and 72 hours for each of the environmental input series, includes circular time variables for the hour, day of month, and time of year, and includes a special binary input variable indicating whether the day is a holiday or not. For this reason, his results are not directly comparable to mine.

A fourth prediction problem involves a data set with 3344 hourly measurements of four time series. Three of the four series are measurements of solar flux, and are available for inputs. The last is a measurement of "true beam insolation". Random time chunks from the data set are selected and set aside to comprise the test set; it consists of 900 observations, which leaves 2444 observations for training.

For this problem, I create 20 input variables by taking the six most recent lags of the three solar flux variables and representing hourly time circularly with sine and cosine. When referring to the input variables for this problem, they are labeled from 1 to 20, first considering time, then the lagged variables from the first to third series. Again, within each series, the most recent time lag is given the lowest label number.

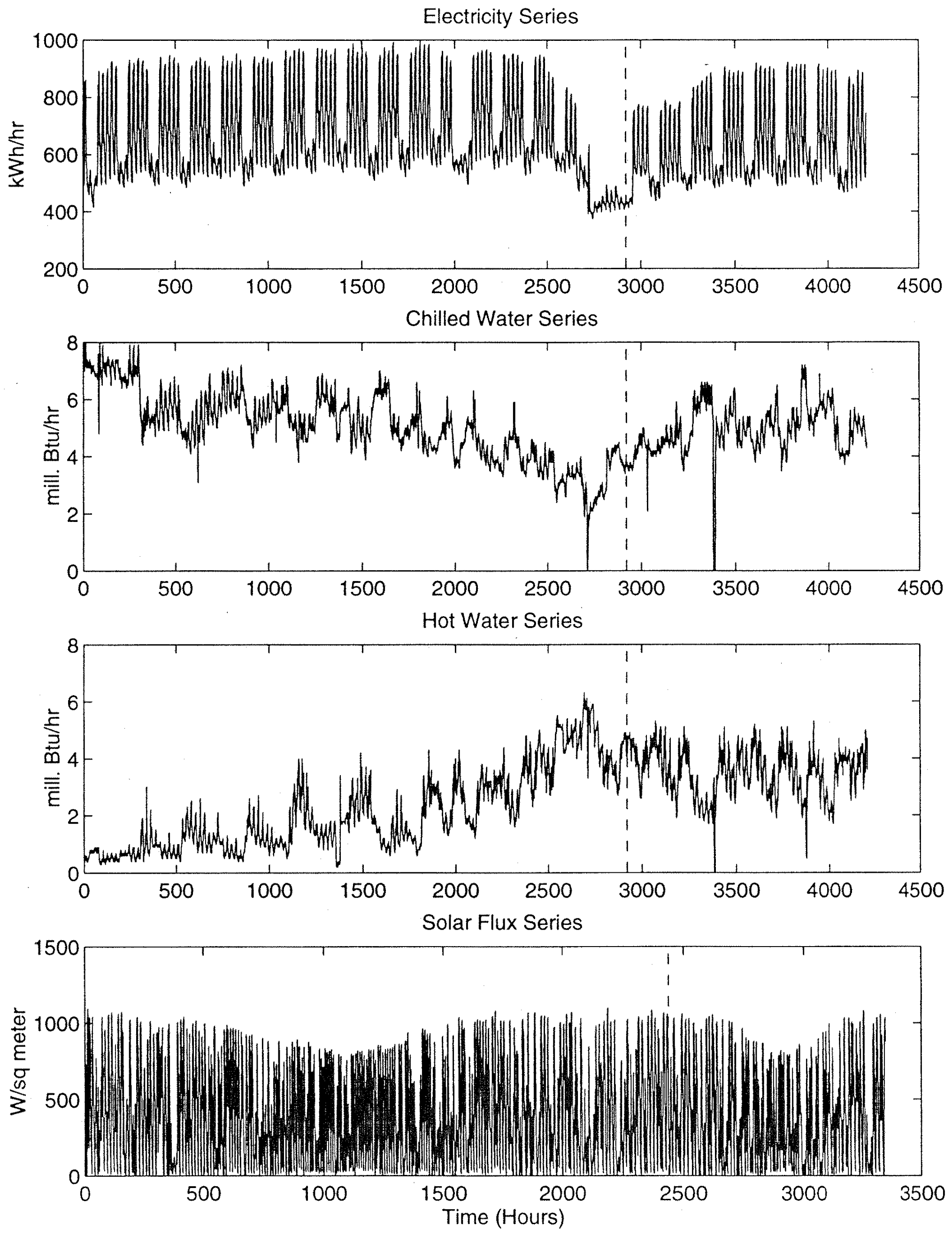


Figure 7.7. Target series for building energy prediction. The dashed lines represent the boundaries between training and test examples: to the left of these lines are the training examples, to the right are the testing examples. Observations in the bottom series are not in exact order, but large chunks of the data are ordered correctly, which gives an accurate feel for the behavior of the series.

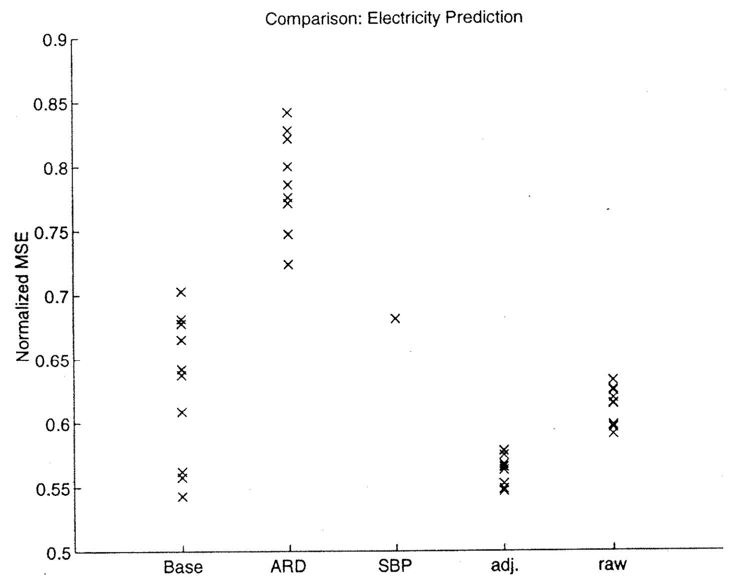


Figure 7.8: Performance of each method on electricity prediction.

Figure 7.7 shows the target series for all four prediction problems. A vertical dotted line on each plot shows the boundary between the training and test sets. All target series exhibit daily periodicity. Weekly periodicity is less evident, except in the top series.

Figure 7.8 shows the performance of all input variable selection methods on the electricity prediction problem. For this problem, SBP selects 21 input variables (unselected variables are $\{X_{25}, X_{21}, X_{26}, X_{24}, X_{17}\}$), raw estimates select $\{X_2, X_1, X_4, X_{14}, X_8, X_5\}$, and adjusted estimates select $\{X_2, X_1, X_5\}$. The performance results for this problem are a bit unusual, in that NMSE values for ARD and SBP actually appear worse than the baseline, where all available input variables are used for prediction. Inputs selected using raw and adjusted estimation do not perform much better than the baseline either, but the variance of the performance is much less, which suggests that the variance of learned weights from run to run is smaller for the selected set of inputs in both cases. This is confirmed by Figure 7.9, which plots weight magnitudes for the four input variables $\{X_0, X_1, X_2, X_5\}$ by raw, adjusted, and baseline networks. The variable X_0 is simply an input variable with a constant value that is included for all network simulations.

The somewhat worse ARD and SBP performances are a bit baffling at first, but a little analysis into the data uncovers an important fact: the prediction problem appears to be nonstationary. then the better performance of raw and adjusted estimates may simply be luck: these methods may simply select more stationary input variables, such as hour of the day, to the exclusions of other input variables which may be relevant in the training data but irrelevant in the test data. Figure 7.10 shows the difference, from training to test set, in means and variances for all input and output variables. Output variables for the electricity, chilled water, and hot water prediction are also presented in the plots; the leftmost 26 variables are the possible input variables, and the three rightmost are the electricity, chilled water, and hot water output variables, from left to right.

Bootstrapping is used to create 1000 training sets and 1000 test sets. Bootstrapped training sets are created from only training data, and bootstrapped test sets only from test data. For each set, a mean and variance are computed, and the difference is taken. Error bars derived from these bootstrapped sets help show the significance of the differences. As a baseline measure, 1000 training sets and 1000 test sets were also created through bootstrapping by first combining both data sets and then sampling from replacement from the combined data. This procedure creates data which should have the same characteristics as the first two sets

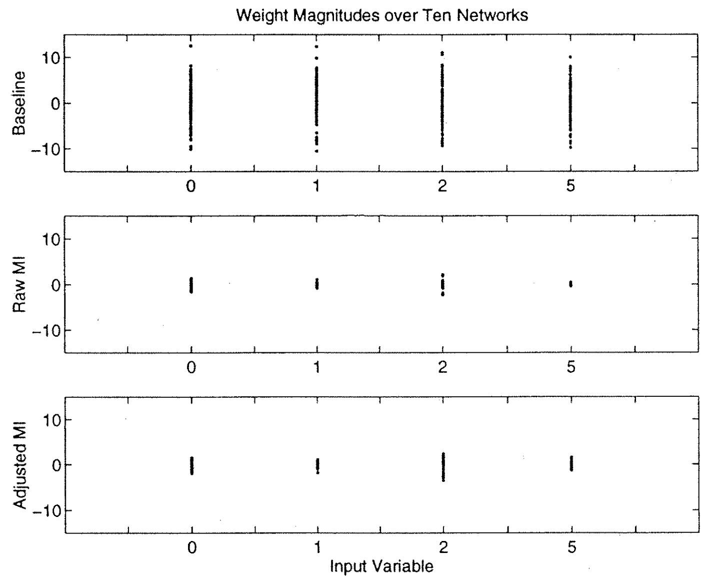
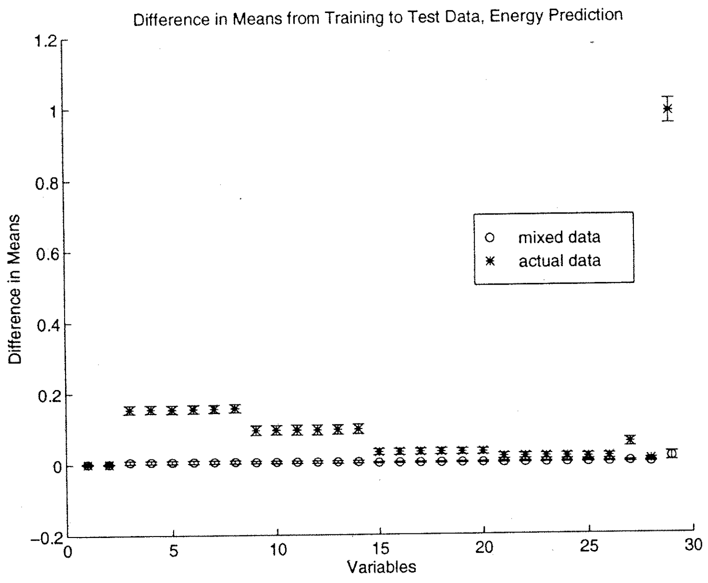


Figure 7.9. Plot of weights for baseline, raw, and adjusted input variables. All weights from the labeled input variables to ten hidden units are plotted. The variances for the weights in the bottom two plots are smaller than for the baseline network, which contains many more input variables.



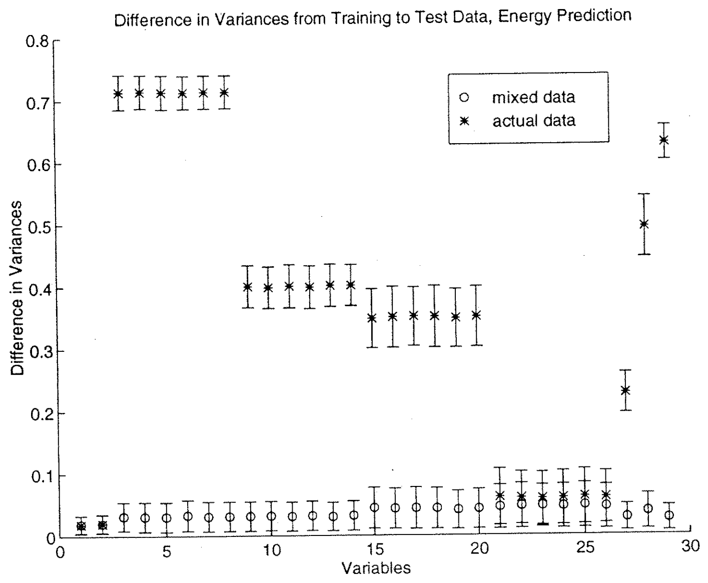


Figure 7.10. Differences in means and variances for all variables involved in energy prediction. In a stationary problem, the error bars for mixed and actual data should overlap for all variables. The fact that they do not for most variables suggests a nonstationary prediction problem.

of bootstrapped data, provided that the prediction problems have stationary distributions. The fact that the differences are large for some variables provides evidence that the problems are nonstationary.

As it turns out, two of the more important input variables for prediction are the two time variables, which appear at the very left of the plots. Using just these two variables for energy prediction results in a mean NMSE of 0.6. In all likelihood, the ARD and SBP algorithms go after additional nonstationary structure present in the training set that ends up not being present in the test set, which causes prediction performance to be poor. The raw and adjusted estimates, on the other hand, select the time variables and only a few of the less stationary input variables, and are more resistant to the time-varying aspects of the data as a consequence. Hence, the performance for these subsets is not as poor as for ARD and SBP. Unfortunately, the fact that the raw and adjusted estimates selected the two time variables cannot be counted upon for other prediction problems, and it has to be seen as simply luck that the performance for raw and adjusted mutual information appear so much better than SBP or ARD.

Similar conclusions apply for the chilled water prediction problem. Figure 7.11 shows how the performance of the different input variable selection algorithms compare. Once again, ARD and SBP do slightly worse than the baseline case. Again, input variables selected using raw and adjusted estimation perform only a little better than the baseline, at best. This time, however, raw and adjusted estimates do not choose the time variables; the variables selected by raw and adjusted estimation are all temperature and humidity variables: $\{X_3, X_{11}, X_8, X_9, X_5, X_7, X_4\}$ for raw estimation and $\{X_3, X_{11}, X_8\}$ for adjusted. Again, the results are best explained as ARD and SBP going after structure present in the training set that is not present in the test set. SBP selected the variables $\{X_8, X_{14}, X_{15}, X_9, X_3, X_7, X_{10}, X_2, X_1\}$.

The results on hot water prediction more clearly reflect the presence of a nonstationary time series. Figure 7.12 shows that all methods, including the baseline, perform quite badly. This is indicated by a normalized mean squared error above 1.0; such results are not possible unless the time series is nonstationary.

So far, results on the first three prediction problems have simply shown that input variable selection methods cannot be guaranteed to help for nonstationary time series, outside of possibly removing some of the nonstationary input variables that can worsen predictions. The next problem, however, shows a different situation. For predicting solar flux, Figure 7.13 indicates that there is not much noise in the target outputs, and predictions are quite good. This is true even when all input variables are involved. Here, results for ARD show that removing inputs can improve the mean error and reduce the variance of the errors also. Results for SBP also show an improved mean, but the variance cannot be evaluated due to the fact that the algorithm is run only once.

Performance for the raw and adjusted estimates are, on the other hand, slightly worse than baseline. It appears that the main reason for worse performance is due to both raw and adjusted estimation stopping before all helpful input variables are selected. The variables selected by raw and adjusted estimation were $\{X_9, X_{15}, X_2, X_1\}$ and $\{X_9, X_3, X_2\}$, whereas SBP selected all but the input variables $\{X_7, X_5, X_8, X_6, X_{14}\}$, with the five most important input variables being $\{X_2, X_{15}, X_3, X_1, X_9\}$. A plot of network performance versus number of input variables (Figure 7.14) shows that raw and adjusted estimation do comparably well to SBP for a given number of input variables, but SBP eventually does better because it selects more variables. A larger data set would enable raw and adjusted selection of more input variables. However, the amount of data may need to be disproportionately large: improvement in performance caused adding a few more input variables may be hard for either algorithm, since the small improvement in performance is overwhelmingly offset by the "spreading out" of data points caused by the addition of the input variable to the density estimates.

7.5 Sleep Apnea Data

This one of the data sets used in the Santa Fe time series prediction competition (Weigend and Gershenfeld, 1994). A sleeping patient was connected to several sensors that monitored heart rate, chest

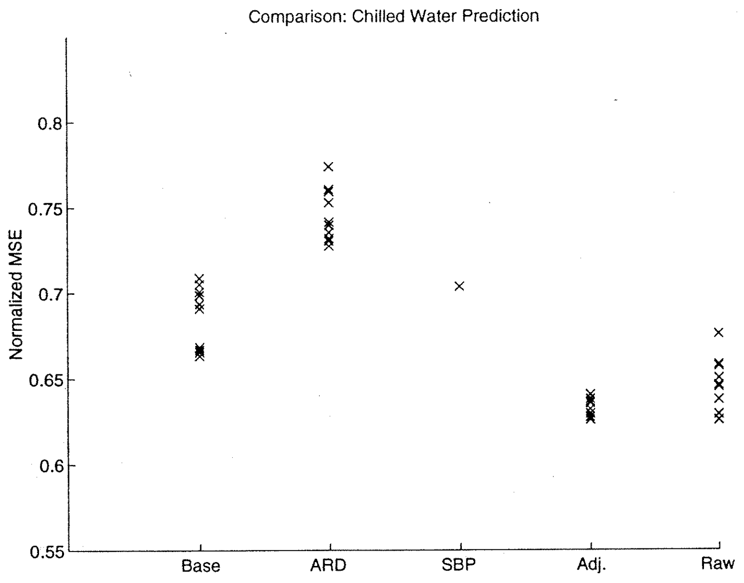


Figure 7.11: Performance of each method on chilled water prediction.

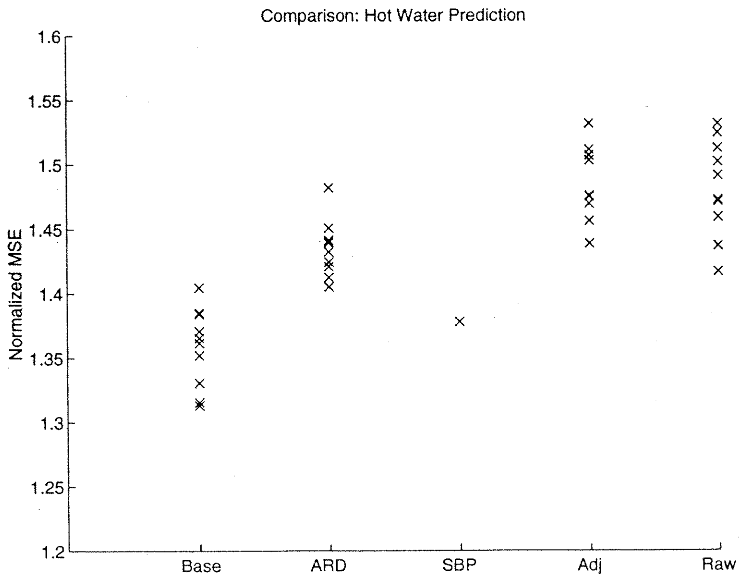


Figure 7.12: Performance of each method on hot water prediction.

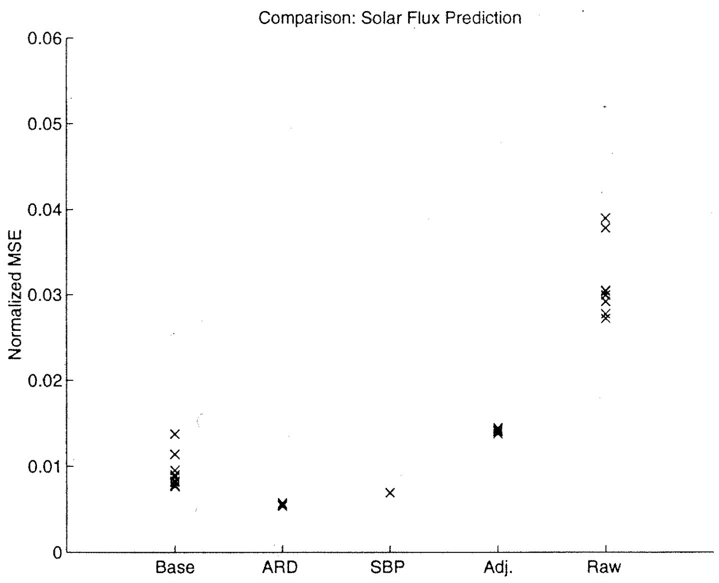


Figure 7.13: Performance of each method on solar flux prediction.

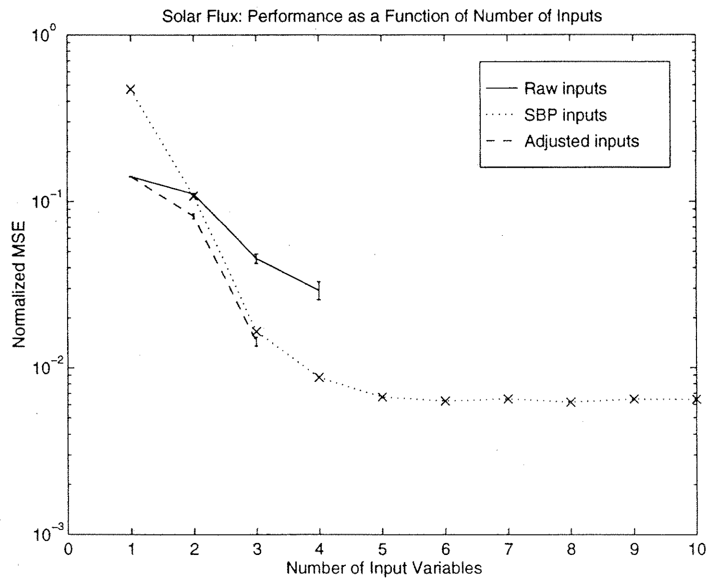


Figure 7.14. Performance of raw, adjusted, and SBP algorithms on solar flux prediction. Error bars for the raw and adjusted methods represent runs over ten networks started with different initial weights. No error bars are reported for SBP, since the algorithm does not allow variable initial weights by construction.

volume (respiration force), and blood oxygen concentration twice every second. The patient was tentatively diagnosed with sleep apnea, which is a condition that results in abnormally long periods of stopped breathing during sleep. This condition is potentially life threatening.

The task is to predict the patient's heart rate from past values from the three time series. For this problem, I take the past ten lags from each time series to create 30 possible input variables. For labeling the input variables, the input variables are labeled in the order of heart rate, chest volume, and blood oxygen, with the most recent lags in each series receiving the lowest numbered labels. For short periods of time, the sensors were disconnected: the related observations are removed by hand. Because of sensor drift, windowed portions of the input variables are rescaled, in blocks of 100 observations, to mean zero and variance one, using a scaling window of 1000 observations. Then the data are permuted randomly and split equally into training and test sets; the final results are 14218 observations for training and 14218 observations for testing.

Figure 7.16 compares results from the input variable selection methods. Once again, SBP shows that performance can be improved by selecting input variables. Raw and adjusted estimates are not quite as good because not enough relevant input variables are selected; SBP requires seven input variables to achieve its performance. The variables selected by raw and adjusted estimation are $\{X_1, X_2\}$ and $\{X_1\}$, respectively; SBP selected $\{X_1, X_2, X_3, X_{11}, X_{21}, X_{28}, X_{16}\}$. Figure 7.15 shows that for the number of inputs that raw and adjusted estimation do select, the performance is comparable with SBP. The slightly worse performance of ARD suggests that the problem is essentially linear, and the bias toward linearity of the other methods cause a slight performance advantage. This is confirmed by doing a linear fit on the seven input variables selected by SBP, which results in a normalized, mean squared error of 0.0575.

7.6 Wind Turbine Data

This data set consists of measurements taken from a wind turbine during a five minute period (McCabe and Fingersh, 1995). The goal is to predict the coefficient of force perpendicular to the wind turbine blade. Toward this end, measurements are taken at a high sampling rate (130.21 Hz), including the local angle of attack of the wind relative to the turbine blade, the wind shear (the difference between wind speeds at the top of the turbine rotor and the bottom), and the current rotational position of the turbine blade. These measurements are converted into nine input variables. The rotational angle of the turbine blade is converted to two input variables: sin(angle) and cos(angle). The angle of attack is processed into six input variables: the angle at the current time, and the change in the angle from time $(t-n)$ to time $(t-n-1)$, $n = 1 \ldots 5$. The wind shear is the last input variable. Measurements are taken for five minutes. To create the training and test sets, I permute the data randomly, take 10000 points for training, and 10000 points for testing. Data are rescaled to have sample mean zero and sample variance one.

Comparative results are presented in the top of Figure 7.17. It shows that performance is not significantly improved by removing input variables. This may be due partly to the fact that there are not that many input variables to begin with. The bottom plot in the same figure also shows that according to SBP, not many input variables can be removed before performance starts worsen. After the removal of three or four inputs, performance decreases steadily. As Section 7.2 would suggest, this it is not a good problem for raw or adjusted mutual information estimation. For the record, both the raw and adjusted algorithms selected $\{X_3, X_2, X_9\}$, and SBP selected $\{X_2, X_3, X_9, X_4, X_8, X_7, X_6\}$.

7.7 Conclusions

This chapter has explored a range of prediction problems, both real and artificial, to determine for which types of problems each input variable selection method works well. All results support the validity of Figure 7.1: the method proposed in this thesis works well when the number of input variables is small and the amount of training data is large. This is supported directly by results from the artificial, solar flux, and sleep apnea problems. Additional variations of the artificial problem help establish that the number of possible,

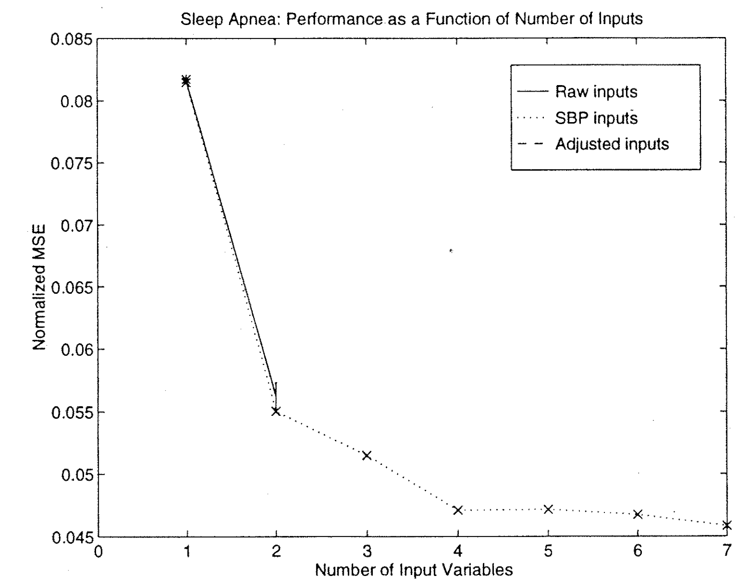


Figure 7.15. Performance of raw, adjusted, and SBP algorithms on the sleep apnea data set. Error bars for the raw and adjusted methods represent runs over ten networks started with different initial weights. No error bars are reported for SBP, since the algorithm does not allow variable initial weights by construction.

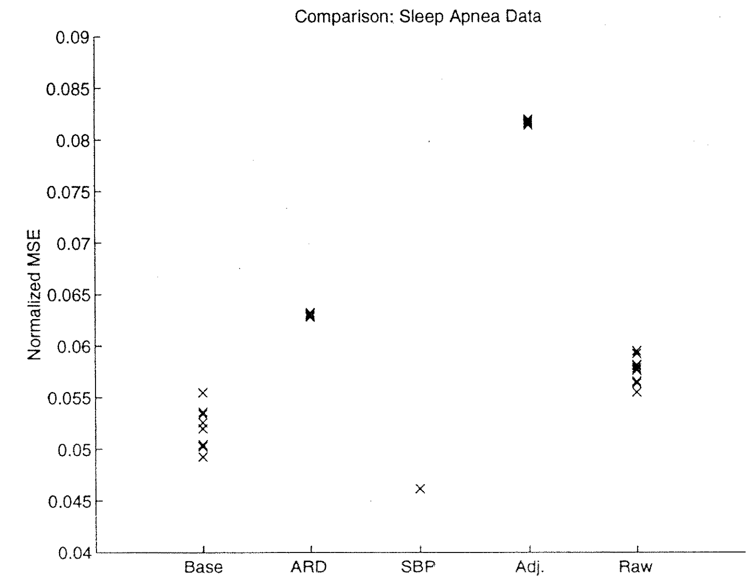
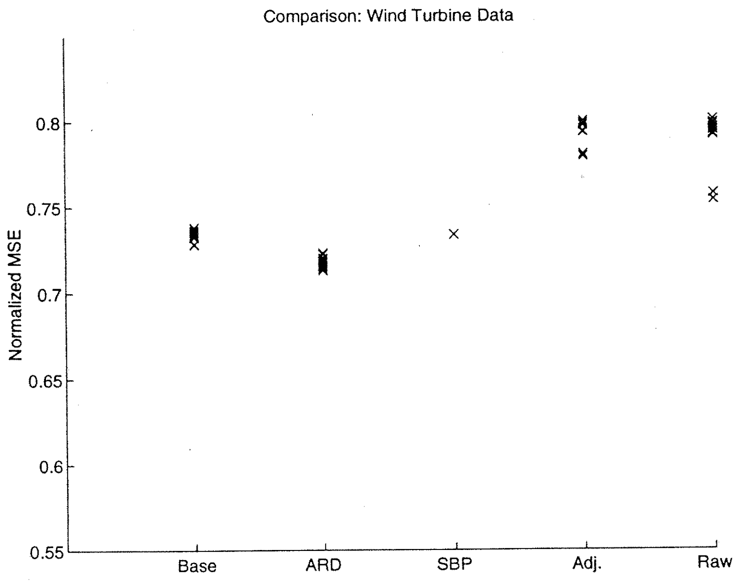


Figure 7.16: Performance comparison: sleep apnea data.



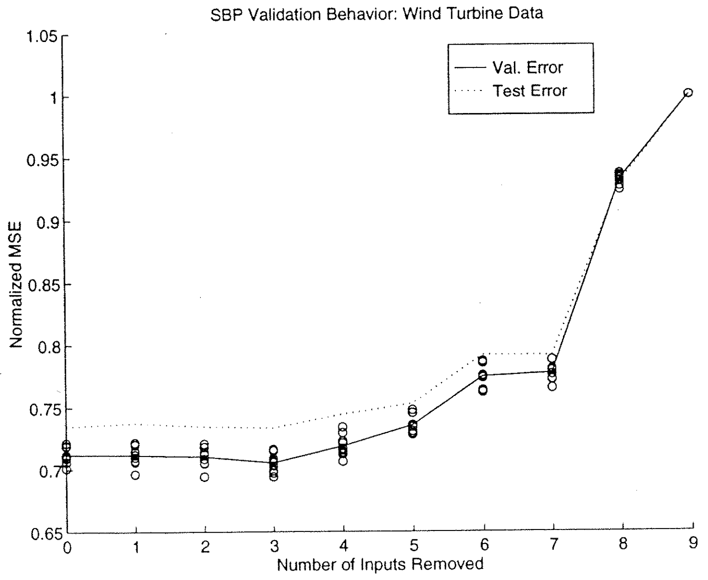


Figure 7.17: Performance comparison: wind turbine data.

independent input variables is not influential in their selection, but the amount of data and the number of relevant inputs are. Results for the wind turbine problem also support the conclusion that the method does not do well when too many input variables are relevant.

All results from this chapter also support Figure 7.1 in showing that SBP and ARD work well when the amount of data are large and the number of possible, independent input variables is small. The most direct evidence for these boundaries comes from the artificial problem, which showed both methods performing poorly with too little data or too many possible, independent input variables. Results on all real world problems are positive for both methods. Even though as many as 30 input variables are available for selection, they are far from independent from each other for the sleep apnea, solar flux, and other energy prediction problems (as simple correlation matrices for all input variables reveal). Therefore, there is not much chance to contradict the claim that SBP and ARD perform poorly with many possible, independent input variables.

In comparing raw and adjusted mutual information estimates, it appears that raw estimation performs better on all problems except solar flux prediction and the nonstationary problems, where performance may be based largely on luck. One possible explanation for raw estimates performing better is that many problems contain output noise, which causes data set size demands to be too great for adjusted estimation (Section 5.2 contains further evidence for this claim). For sleep apnea, however, output noise appears to be low, and yet performance for raw estimates are still better than adjusted. It may be that for this problem, output noise is so low that selecting nearly redundant input variables still boosts prediction performance. Adjusted estimates are less likely to select redundant input variables (Section 5.1), which may explain why its performance is lower than the raw estimates.

Both raw and adjusted estimation, in their current implementations, appear to suffer from stopping before all relevant input variables are selected. This is most apparent in the sleep apnea problem, where raw estimation selects only two input variables, and adjusted estimation selects only one, despite there being 14218 data points. This apparently poor behavior may not be a fundamental problem, but simply a consequence of how the stopping criterion for forward selection is currently implemented. Right now, forward selection is stopped whenever the addition of any possible input variable causes estimated mutual information to decrease. Rather than adhering to the strict idea that mutual information estimates are always ordered correctly, the algorithm could simply be implemented to select the input variable that causes mutual information estimates to decrease the least. The decision of how many input variables to use would then be performed by training several networks with different numbers of input variables.

Another possible change that could have a drastic effect on selection performance involves adjusting the number of bootstraps while the selection algorithm is running. If the currently "winning" input variable is not winning by a high enough significance level, more bootstraps are performed for the winning input variable and all of its close competitors. A winner is not selected until it is deemed significantly better. This approach is termed "racing" (Maron and Moore, 1994).

In summary, a comparison of raw versus adjusted estimates based on all the empirical results of this thesis produces no clear winner. However, a final comparison should involve more explorations of the criteria for stopping forward selection and for selecting an input variable. A more detailed description of additional enhancements is presented in Section 8.1.

CHAPTER 8

CONCLUSIONS

This thesis has explored a new method for selecting input variables for connectionist learning. It contains both theoretical and empirical analysis of the method's components, which lend understanding to the method's behavior and also point to shortcomings and possible improvements. Another portion of the thesis is devoted to extensive comparisons between the method and two competitors from the connectionist literature.

Analysis for the method began in Chapter 4 with an examination of the properties of mutual information as an analytic measure. It was shown that mutual information is a natural measure of input variable relevance, in that it quantifies the reduction in uncertainty, or entropy, of an output variable due to knowledge of input variables. Input variables with high associated mutual information estimates greatly reduce the uncertainty associated with an output variable. It was also shown that while mutual information can sometimes order a collection of input variable subsets in exactly the same way as expected conditional output variance, it is actually fundamentally different from this and other variance based measures. Lastly, it was shown that an unbiased, sample-based approximation to mutual information exists.

Chapter 5 devoted itself to the practical consequences of estimating mutual information from data. Two ways of estimating mutual information were introduced: raw and adjusted estimation. It was shown that the accuracy of both methods relies on how densely populated the points in the estimate are. Both raw and adjusted estimation have their advantages and shortcomings: raw estimation is more susceptible to selecting irrelevant or redundant input variables, whereas adjusted estimates tend to require a larger amount of data to select the same number of relevant input variables as raw estimation. One can view raw and adjusted estimation, respectively, as "headstrong" and "conservative". In general, the amount of data required by both measures is an exponential function of the number of relevant input variables. However, a significant number of relevant input variables may be selected in practice despite this fact.

Chapter 6 was concerned with the issue of estimating the number of modes in the conditional output density for a selected set of relevant input variables. This is necessary to verify that the subset of input variables selected using mutual information has a unimodal conditional output distribution. The best one can hope to estimate is the expected number of modes, since the number can vary as a function of the input. It is shown that kernel based mode estimation works well for a selected set of six problems, and that the accuracy of mode estimates decreases only slightly as a function of the number of input variables. As it turns out, however, none of the mode estimation methods were needed for the comparisons of Chapter 7, since performance with standard networks was satisfactory.

Chapter 7 went on to compare the method against two competitors from the connectionist literature: Sensitivity Based Pruning and Automatic Relevance Determination. Results on an artificial problem clearly demarcate the types of problems for which each method is appropriate. It is shown that the method, unlike its competitors, can handle a large number of possible, independent input variables. However, the amount of available data must be large and the number of relevant input variables must be relatively small before the method is able to identify all relevant input variables correctly.

Results on six real world prediction problems support similar conclusions. For one of these problems, where the number of possible input variables is relatively few, no method improves generalization performance greatly over simply including all input variables. Moreover, the method analyzed in this thesis performs less well on this problem because a relatively large number of input variables have at least marginal relevance. For another two prediction problems, there are a much larger number of possible input variables, and fewer variables are required for good predictions. All methods do well on these two problems, although baseline performance with all variables included is still not beaten significantly by any method because the number of independent inputs is still quite small. Results on three other prediction problems show that none of the input variable selection methods can be expected to perform well on nonstationary time series prediction problems.

One can argue that the method presented in this thesis is most appropriate as part of a completely automated approach to prediction, or as a starting point for understanding data set characteristics. For data sets where observations are collected automatically and many candidate input variables are generated blindly, it is more likely that many irrelevant input variables and only a small number of relevant input variables exist than if the data set is examined thoroughly ahead of time and preprocessed by an expert. The method may be useful to model designers as a starting point for data set understanding: quick identification of the most relevant input variables may suggest possible improvements to data collection or input variable preprocessing.

8.1 Future Research Directions

There is still possible work to be done on the proposed method. The biggest shortcomings of the method include:

- Occasional selection of irrelevant inputs by raw estimation
- Occasional selection of dependent inputs by raw and adjusted estimation
- Large amounts of data required to select even small numbers of input variables.

All of these shortcomings could be remedied by a different approach to kernel width selection within the method. In the current approach, kernel widths for mutual information estimation are determined separately for each candidate input variable subset. The determination is based upon likelihood cross validation. Using likelihood cross validation allows for meaningful comparisons of mutual information estimates across input variable subsets of different sizes. As a result, there is a built in stopping condition for adding more input variables than are necessary.

However, this approach to kernel width selection also suffers from certain problems. The most immediate problem comes when one compares input variable subsets of the same size, in order to select the next most relevant input variable. Likelihood cross validation tends to base its choice for kernel width on the largest nearest-neighbor spacing between any two points. Although, for large data sets, this spacing tends to be similar for different input variables, it does not have to be so when the input variables have markedly different distributions. The bottom line is that likelihood cross validation's choice of kernel width can result in a high variance for mutual information estimates, making the identification of the most relevant input variables more difficult.

A different approach that could solve the robustness problem might involve using the **same** kernel width for all input variable subsets of the same size, rather than choosing a different width each time a different candidate input variable is considered. Differences in data point spacings would no longer throw off mutual information estimates, making the selection of irrelevant or redundant input variables before relevant ones much less likely. In addition, a technique known as **racing** (Maron and Moore, 1994) could be used to make sure that one input variable has a significantly better mutual information estimate before it is selected. Bootstrapping would be performed on an input variable until its average mutual information estimate became poorer than another by a predetermined significance level, such as 95 %.

One drawback to this approach is that forward selection would have to continue until all variables have been selected, since there would no longer be a principled stopping criterion with all kernel widths being the same for all subsets of a certain size. Several networks would have to be trained and compared, where each network contains progressively more input variables in the order determined through forward selection. The network with the best apparent generalization performance would then represent the choice of input variables.

There is also the issue of which fixed kernel width to use. The decision cannot be entirely arbitrary, since mutual information estimates are affected by a bias-variance tradeoff in the same way that networks with too few or too many weights are. Using kernel widths that are too narrow creates structure in the data where there is in fact none, and using widths that are too wide obliterates important structure. A possible way out might be to use likelihood cross validation for one candidate input variable and then use the same width for all successive candidate input variables until one is selected. Then, a new width is selected for the next round of candidate input variables, and the process of selection is repeated until all are selected.

BIBLIOGRAPHY

Aarts, E. H. L., Korst, J. H. M., and Zweitering, P. J. (1996). Deterministic and randomized local search. In Smolensky, P., Mozer, M. C., and Rumelhart, D. E., editors, **Mathematical Perspectives on Neural Networks**. Erlbaum Associates.

Abu-Mostafa, Y. (1995). Hints. **Neural Computation**, 7:639–671.

Allison, J. J. (1994). Explorations of Bayesian input relevance determination in neural networks. Master's thesis, University of Colorado.

Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. **IEEE Transactions on Neural Networks**, 5(4):537–550.

Beale, E. M. L. (1970). Note on procedures for variable selection in multiple regression. **Technometrics**, 12(4):909–914.

Bishop, C. M. (1994). Mixture density networks. Unpublished report, Aston University.

Cover, T. M. and Thomas, J. A. (1991). **Elements of Information Theory**. John Wiley, New York.

Efroymson, M. A. (1960). Multiple regression analysis. In Ralston, A. and Wilf, H. S., editors, **Mathematical Methods for Digital Computers**. Wiley.

Epanechnikov, V. A. (1969). Nonparametric estimation of a multivariate probability density. **Theory of Probability and its Applications**, 14:153–158.

Finnoff, W., Hergert, F., and Zimmermann, H. G. (1993). Improving generalization performance by nonconvergent model selection methods. **Neural Networks**, 6:771–783.

Fraser, A. M. (1988). **Information and Entropy in Strange Attractors**. PhD thesis, University of Texas, Austin.

Furnival, G. M. and Wilson, R. W. (1974). Regressions by leaps and bounds. **Technometrics**, 16(4):499–511.

Härdle, W. (1989). **Applied Nonparametric Regression**. Cambridge University Press, Cambridge, MA.

Hartigan, J. A. and Hartigan, P. M. (1985). The dip test of unimodality. **The Annals of Statistics**, 13(1):70–84.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). **Introduction to the Theory of Neural Computation**. Addison-Wesley, Reading, MA.

MacKay, D. J. C. (1993). Bayesian non-linear modeling for the energy prediction competition. Unpublished report.

Mallows, C. L. (1973). Some comments on C_p. **Technometrics**, 15:661–675.

Maron, O. and Moore, A. (1994). Hoeffding races: accelerating model selection search for classification and function approximation. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, **Advances in Neural Information Processing Systems 6**, pages 59–66. Morgan Kaufmann.

McCabe, T. M. and Fingersh, L. J. (1995). Cluster-based regression using neural networks. Unpublished report, University of Colorado.

Miller, A. J. (1990). **Subset Selection in Regression**. Chapman & Hall, London.

Moody, J. (1994). Prediction risk and architecture selection for neural networks. In Cherkassky, V., Friedman, J. H., and Wechsler, H., editors, **From Statistics to Neural Networks: Theory and Pattern Recognition Applications**, NATO ASI Series F. Springer-Verlag.

Moody, J. E. and Utans, J. (1992). Principled architecture selection for neural networks: application to corporate bond rating prediction. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, **Advances in Neural Information Processing Systems 4**, pages 683–690. Morgan Kaufmann.

Neal, R. M. (1995). **Bayesian learning for neural networks**. PhD thesis, University of Toronto.

Optican, L. M., Gawne, T. J., Richmond, B. J., and Joseph, P. J. (1991). Unbiased measures of transmitted information and channel capacity from multivariate neuronal data. **Biological Cybernetics**, 65:305–310.

Quinlan, R. (1993). Combining instance-based and model-based learning. In **Machine Learning: Proceedings of the Tenth International Conference, Amherst, Massachusetts**, pages 236–243. Morgan Kaufmann.

Ross, S. (1991). **A First Course in Probability**. Macmillan Publishing Company, New York.

Scott, D. W. (1992). **Multivariate Density Estimation, Theory, Practice and Visualization**. John Wiley and Sons, New York.

Seber, G. A. F. and Wild, C. J. (1989). **Nonlinear Regression**. John Wiley and Sons, New York.

Silverman, B. W. (1981). Using kernel density estimates to investigate multimodality. **J. R. Statist. Soc. B**, 43(1):97–99.

Silverman, B. W. (1986). **Density Estimation for Statistics and Data Analysis**. Chapman-Hall, London.

Thodberg, H. H. (1994). Bayesian backprop in action: Pruning, committees, error bars and an application to spectroscopy. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, **Advances in Neural Information Processing Systems 6**, pages 208–215. Morgan Kaufmann.

Thompson, M. L. (1978). Selection of variables in multiple regression: Part I. A review and evaluation. Part II. Chosen procedures. **International Statistical Review**, 46:1–19 and 129–146.

Utans, J., Moody, J. E., Rehfuss, S., and Siegelmann, H. (1995). Input variable selection for neural networks: Application to predicting the U.S. business cycle. In **IEEE/IAFE Conference on Computational Intelligence for Financial Engineering**.

Weigend, A. S. and Gershenfeld, N. A., editors (1994). **Time Series Prediction: Forecasting the Future and Understanding the Past**. Addison-Wesley, Reading, MA.

Weigend, A. S. and Srivastava, A. N. (1995). Predicting the probability distributions: A connectionist approach. **International Journal of Neural Systems**, 6(2):109–118.