UNIVERSITY OF CALIFORNIA, SAN DIEGO

Selecting Training Exemplars
for Neural Network Learning

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science and Engineering

by

Mark Plutowski

Committee in charge:

Professor Halbert White, Co-Chairperson
Professor Garrison Cottrell, Co-Chairperson
Professor Paul Kube
Professor Ronald Getoor
Professor Walter Savitch

1994

The dissertation of Mark Plutowski is approved, and it is acceptable in quality and form for publication on microfilm:

Co-Chair

Co-Chair

University of California, San Diego

1994

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acknowledgements

Hal's skill for brief expression of comprehensive material permeates not only his written communications, but his spoken word as well. His skillful guidance and command of the material illuminated my path clearly, making my studies both enjoyable and productive. If Hal lit the way, then Gary helped me navigate through and around its obstacles. His clear insight into what is at the heart of the mechanism behind the various learning algorithms made it easier to track a field of research that expanded quickly during my tenure as a graduate student. His sensible insights into the way of scientific research were even more valuable. Gary also tirelessly helped translate my documents and presentations into plain yet clear language, making them understandable to a wider audience. I would like to thank the remainder of my committee as well, especially Paul Kube, whose interest in my work has made it all the more interesting to me. I also thank Paul for being from Grand Forks, North Dakota.

I'd like to thank the CSE Technical Support Staff, including Rick Ord, Glen Gilbert, Eric Anderson, Steve Hopper, Cindy Paloma, Glenn Little, Dave Hutches, and Dave Wargo. I am fortunate indeed that, at any given time, at least one of them was available to help me out of a bind. For instance, this acknowledgement text was retrieved by Eric and Dave late on a Sunday night, after a save error wiped out a file the day before I was to deliver the final version of this dissertation.

I would not have made it very far through this program without the Administrative Staff, including Patricia Naughton, Kathy Hay, Julie Douville, Joanna Mancusi, and Patricia Razcka. Their ability to cut through complexities at critical times turned the task of graduating from an NP-hard problem to one solvable in linear time. I thank the Cognitive Science Department *in toto*, for being my surrogate department at various times. Special thanks go to Shinichi Sakata, whose beautiful contributions to Chapter VI make the proofs more elegant yet rigorous, significantly relaxing the conditions necessary for application of these theorems.

I thank David Wolpert, Jan Larsen, Jeff Racine, Axel Roebel, Andres Dale, Duane Desieno, and Brian Ladner, for valuable discussions. Thanks to to Dave Devers and David Noelle, for helping me clarify my presentations, as well as the remainder of GURU (Gary's

Unbelievable Research Unit) including Steve Biafore, Mai Nguyen, Dan Clouse, Margaret Myers, and Curt Padgett. They sat through several iterations of my presentations, attentively and critically, contributing constuctive comments each and every time. Tim Ash generously provided me with simulator code early on in my graduate studies, and more importantly, always a good laugh.

For moral support, I thank Steve Watkins first and foremost, for helping me learn to recognize what it was that I had already accomplished, as well as in helping me view my remaining goals more clearly. I also thank Seana Coulson, Maria Van Liew, Rick Anthony, Alex Young, and Denise Fenton for invaluable friendship. Thanks to my family for keeping me down to earth (meaning that they can humble me at will).

Thanks to Micha Alicia Espinosa for making finishing worthwhile.

VITA

August 29, 1961	Born, Grand Forks, North Dakota
1983	B.S. University of North Dakota
1983–1987	Member of Technical Staff, Rockwell International
1986	M.S. University of Southern California
1987–1988	Teaching Assistant, Department of Computer Science and Engineering University of California, San Diego
1988	Research Assistant, Department of Computer Science and Engineering University of California, San Diego
1988–1991	Staff Scientist, Hecht-Nielsen Neurocomputers Inc, San Diego
1992-1994	Research Assistant, Department of Computer Science and Engineering University of California, San Diego
1994	Doctor of Philosophy University of California, San Diego

PUBLICATIONS

Plutowski, Mark E., Shinichi Sakata, and Hal White. (1994) "Cross-validation Estimates Integrated Mean Squared Error." Cowan, J.D., Tesauro, G., and Alspector, J. (eds.), *Advances in neural information processing systems 6,* San Mateo, CA: Morgan Kaufmann Publishers.

Watkins, Steven S., Paul M. Chau, and Mark Plutowski, Raoul Tawel and Bjorn Lambrigsten. (1994) "A Hybrid Radial Basis Function Neurocomputer." Cowan, J.D., Tesauro, G., and Alspector, J. (eds.), *Advances in neural information processing systems 6,* San Mateo, CA: Morgan Kaufmann Publishers.

Plutowski, M.E., Cottrell, G.W. and White, H. (1993) Experience with selecting exemplars from clean data. Submitted, pending review.

Plutowski, M.E., Cottrell, G.W. and White, H. (1993) Learning Mackey Glass from 25 examples, plus or minus 2. Presented at NIPS 5, December 1992. Published in NIPS 6: *Advances in neural information processing systems 6*, 1994. Cowan, J.D., Tesauro, G., and Alspector, J. (eds.) San Mateo, CA: Morgan Kaufmann Publishers.

Plutowski, Mark, and Halbert White. (1993). Selecting exemplars for training feedforward networks from clean data. *IEEE Transactions on Neural Networks.* 4, 3, pp.305-318.

Plutowski, M.E., Cottrell, G.W., and White, H. (1992) “Behavior of the active selection algorithm on a time series learning task.” Proceedings of the INC (Institute for Neural Computation) Annual Research Symposium. June 12, 1992.

Plutowski, Mark, and Halbert White. (1991). Active selection of training examples for network learning in noiseless environments. *Technical Report No. CS91-180, Computer Science and Engineering Department, University of California, San Diego.*

Plutowski, Mark. (1988). Backpropagation with higher order units. *Poster, IEEE International Conference on Neural Networks,* San Diego, California, June, 1988.

Plutowski, Mark. (1986). VOSS - A VHSIC Hosted Expert System. *Proceedings of IEEE NAECON'86*, Dayton, Ohio.

FIELDS OF STUDY

Major Field: Computer Science and Engineering

Studies in Applied Neural Networks.
Professor Garrison Cottrell

Studies in Statistical Neural Network Theory.
Professor Halbert White

ABSTRACT OF THE DISSERTATION

Selecting Training Exemplars
For Neural Network Learning

by

Mark Plutowski
Doctor of Philosophy in Computer Science and Engineering
University of California, San Diego, 1994
Professor Halbert White, Co-Chair
Professor Garrison Cottrell, Co-Chair

We examine methods for selecting training exemplars for the purpose of training neural networks. We use statistical methods to obtain techniques based upon an objective criterion, obtaining techniques for both clean and noisy data. The techniques efficiently select concise sets of exemplars. The methods we develop are not restricted to use on neural networks, being applicable to nonlinear regression in general. These techniques are nonparametric in the sense that they automatically adjust the complexity of the network model as appropriate for the learning task at hand. Empirical evidence with the method for clean data indicates that this technique can be extremely beneficial to making learning more efficient and reliable. The method also selects concise training sets, substantiating recent theoretical advancements in understanding the relationship between network complexity and the number of training examples. The algorithm we developed for this method has been rigorously tested, and proves to be highly autonomous and robust. We also prove that a well known estimate of network learning performance is statistically accurate and precise. This result has wide scope, being applicable to nonlinear regression in general. Finally, preliminary results with a method for selecting noisy exemplars indicate that this method has the potential for making classical optimization algorithms more robust and reliable in the presence of noise.

Chapter I

Introduction

I.A Scope

Neural network learning procedures are popular and effective techniques for fitting a nonlinear model to an unknown mapping by learning from a set of "training examples." This dissertation studies techniques for selecting training examples from a given set of available examples. These procedures result in the selection of a small subset of the available examples which, in a precise sense, contain most of the information available in the entire set.

It has been shown that neural network learning procedures are inherently statistical techniques. In particular, they are a special case of nonlinear nonparametric regression techniques [110], using least-squares as the optimization criterion. This dissertation employs and extends statistical models of learning. Therefore, we hope that it will be of interest to learning theoreticians, including statisticians. We follow through on our theoretical development with empirical implementation and testing. The resulting methods are intuitively pleasing and straightforward to compute. Practical implementation issues are addressed in detail, and the techniques are demonstrated on example learning tasks. Therefore, this dissertation should also be of interest to the practitioner interested in applying our methods.

I.B Organization of this Dissertation

This chapter introduces and motivates the thesis topic, then defines it, classifying the general topic into two distinct but complementary methodologies, one of which is the focus of the remainder of this dissertation. Chapter II examines related work, providing encapsulated summaries of previous results and providing the context for the results of this dissertation. Chapter III collects together the formal definitions and assumptions used in this dissertation. Chapter IV presents the formal development of a method for selecting training examples from "clean" (i.e., "noiseless") data. Chapter V presents the results of an experimental analysis of the algorithm developed in Chapter IV. Chapter VI presents formal results that, in addition being relevant to statistical nonlinear regression in general provide the theoretical foundation for the study of methods for selecting training examples from noisy data. Chapter VII presents preliminary experimental results from a method developed for selecting training examples from noisy data, based upon the experimental and theoretical results of the previous chapters. Proofs and detailed mathematical development are relegated to the appendices.

I.C Introduction

In practice, most current approaches to neural network learning simply present the network with all of the training examples. This can be very costly for large training sets, and can even cause the network to fail to converge to an acceptable minimum error [105, 34]. Training sets may include many redundant or noisy examples that impede learning. Thus, it is of interest whether a minimal set of training examples can be selected from the training set that is in some sense optimal for the network trying to fit the data. This could be beneficial for training time as well as providing a better fit to the data.

We refer to examples selected in this way as "exemplars." Our goal is to minimize the number of exemplars that must be used for training, given a set of many candidates to choose from. We will use the term "exemplar selection" to refer to all techniques for selecting exemplars. There exist methods for selecting exemplars that do not require using the network in the selection process. In contrast, the methods we develop for selecting training examples from a given data set are known as as *active selection* of training examples. Rather than passively accepting examples generated by some unknown data generating process, knowledge accumulated by the network from previous training examples can be exploited to judiciously select subsequent exemplars. Therefore, the network can be viewed as an active participant in its training.

If this approach turns out to be generally applicable, several benefits are immediately apparent for a wide range of situations. First, if a set of candidate examples is too large to contain in working memory, our method provides a useful method of selecting a small, yet still sufficiently informative training set. Second, the development of our techniques do not depend upon using a particulator estimator (neural network or otherwise); thus they may be used by any statistical estimator. Third, the method could be used in data-driven learning techniques where the model complexity is directly related to the number of exemplars. For example, in [85], the basis functions are allocated at a rate related to the number of exemplars, which are chosen randomly from the set of candidates. Our principled way of choosing the exemplars could make such algorithms more efficient by maximizing the information content of each exemplar, resulting in a smaller network. Fourth, learning algorithms which require evaluation of the entire training set at each iteration may be made more efficient. In particular, parameter update algorithms involving the Hessian matrix (or its approximation) and conjugate gradient techniques are likely candidates. Currently, heuristics are employed with conjugate gradient algorithms for the purpose of reducing the number of examples that need to be considered at each iteration [78, 79].[1]

In general, maximizing the information density of the training set may have the side benefit of making training more reliable and efficient. Iterative learning algorithms that process each example a large number of times over the course of training could benefit a great deal from a minimal training set by a reduction in the overall cost of training. We have confirmed this conjecture on small problems. Our evidence indicates that the benefit of these techniques outweigh the overhead of the selection process, that the overhead of selection is typically a small percentage of the total cost of training, and that generalization can be improved over using all of the available examples.

[1] Methods for making each iteration more efficient are developed in [77, 79].

I.D Motivation

The run-time complexity of learning algorithms clearly depends upon the number of examples used in training. This is especially evident in iterative learning algorithms that process each example a large number of times over the course of training. Empirical experience clearly indicates that the accuracy and reliability of the network training depends greatly upon the amount of data available for training. In practice, we have seen applications with huge data sets, in which the amount of processing time required for learning is driven by the amount of time necessary for the learning algorithm to iterate through the data set. Similar accounts of this effect abound in the learning literature.

Formal considerations provide considerable justification for this empirical experience. Much work has been done examining generalization ability as a function of the number of training examples, in a variety of contexts (cf., [64, 9, 10, 44, 45, 8]). These sample bounds estimate the number of examples required for proper coverage of "example space" (the space of all examples). They provide insight to the data requirement of learning under conventional sampling methods.

Most of these results indicate that for examples passively sampled according to the environmental distribution, the number of examples sufficient for proper training grows with the complexity of the network model. Measures of network complexity include the number of learnable parameters ("weights"), nodes, or layers, a Lipshitz or other smoothness condition on the network function, or the number of bits of precision to code the network. A common result is that the number of examples sufficient to ensure a small generalization error of ϵ with probability $1 - \delta$ grows at least as fast as the number of weights, p (growing also with decreasing ϵ or δ). These bounds are highly application dependent, depending greatly upon the particular method of estimating the capacity of the class of functions representable by the model being considered [44].

We provide a concrete example of such a bound relevant to our situation in Appendix A, derived using results from [111]. There we demonstrate that for a particular common neural network architecture, with reasonable choices of ϵ and δ, the sufficient number of examples is approximately $O\left(\Delta^4 p \, ln \Delta\right)$, where Δ is a bound on the maximum absolute magnitude of the weights. See Equation (A.6) for the precise statement of this result. This gives a typical example of how the number of examples "sufficient" to obtain a desired level of generalization (here, in terms of a δ,ϵ bound on the error) depends upon the number of network weights. Under this particular analysis, the bound is even more sensitive to Δ, because with all other parameters being fixed, the bound grows as $\Delta^4 \ln \Delta$. It also illustrates the approximations that are typically made in order to obtain such bounds. Because of these approximations, the bounds are necessarily inexact, and are suspected by practitioners of being quite loose, as far fewer exemplars are typically required in practical applications. This is to be expected, due to the approximations that are employed to arrive at these results, as well as to the fact that these bounds are typically obtained via worst-case analysis (although some work exists using average case analysis [45]). However, such bounds are useful in indicating the rate at which the data requirement grows as network complexity increases.

We can turn such a bound into a statement about the benefit that a single random example provides, as measured by the amount it helps reduce generalization error (where we measure generalization error in the same δ,ϵ sense as above, as the ability to obtain accuracy within ϵ with probability $1-\delta$). We do this in Appendix B, using the sample bound derived in Appendix A. In particular, we derive a bound on the rate of decrease of a measure of the benefit due to a single random (more precisely, i.i.d.[2]) example. We are able to show that the benefit of appending a randomly selected example to the training set decreases exponentially as the size of the training set increases. We hope to counteract this effect by judicious selection of exemplars by exploiting knowledge gained by learning on previously selected examples.

Another related issue is the number of examples that can always be fit by a given network. It turns out that the VC dimension corresponds to this number. In [71] there is a constructive proof that feedforward neural networks with two hidden layers of linear threshold gates have VC dimension $\Omega(p \log p)$, where p is the number of weights in the network.

We surmise that such results may be closely related to the lower bound on the number of examples *necessary* to train a network of given size. This is due to correspondence between the VC dimension and the number of arbitrary examples that the network is capable of fitting exactly. If a network trained on $m-1$ examples is capable of fitting m arbitrary examples, it is possible, in general, for it to generalize poorly away from the current training set, since it is flexible enough to fit another arbitrary example. For a number of examples greater than the VC dimension, there exists some training set for which the network must resort to interpolation in order to achieve the least-squares fit. Therefore, it is plausible that the VC dimension gives a lower bound on the number of examples necessary to ensure generalization, in general.[3]

Our empirical results shed some light on this issue, because the method developed here for clean data selects a number of exemplars equal to the number of network weights times a small constant just greater than 1.[4]

I.E Overview

This section provides an overview of the remainder of this dissertation.

There are two distinct techniques for choosing training examples given information derived from a network partially trained on a set of previously acquired examples, namely, "active sampling" (the task of adding new examples to the set of available examples) and "active selection" (the task of selecting training *exemplars* from the set of available exam-

[2] "independent, identically distributed"

[3] It is also possible for the learning algorithm itself to effectively lower the data requirement. This can occur if the learning algorithm imposes a smoothness constraint on the network function. We have witnessed this phenomenon in our experience with training networks on exemplar sets that are smaller than the VC dimension of the network. When presented with a number of exemplars smaller than the VC dimension of the network, learning would typically result in a smooth fit to the data, in spite of the fact that the network is capable of fitting the same data with a very large set of functions containing many that are much less smooth.

[4] This result does not confirm the $p \log p$ bound obtained by Maass. However, our network architecture is not of the construction used in [71], therefore it is possible that its VC dimension grows more slowly than $\Omega(p \log p)$. Indeed, we conjecture that one may combine Equation (A.3) (which gives an expression for the capacity of the class of neural networks we consider here, in terms of its metric entropy) with results from [44] (which relates metric entropy with other measures of model capacity closely related to the VC dimension) to obtain an expression for the VC dimension of the class of neural networks we consider here that more closely resembles the sample bound given by Equation (A.6).

ples). In Chapter II we review the literature on both approaches, since they are related in various ways.

In Chapter III, we formalize all definitions, notation, and assumptions used in this dissertation, along with brief explanations of each in plain English. Intuitively, sampling techniques seek training examples representative, in some sense, of the space of all possible training examples. To evaluate training examples we have chosen an objective sampling criterion based on the Integrated Mean Squared Error (IMSE), which is appropriate for use in conjunction with the least-squares error criterion used for training. This criterion objectively quantifies the tradeoff between model bias (error due the effect of the model being incorrectly specified) and model variance (error due to the effects of random variation in the training data). Chapter III culminates in a demonstration of this well-known decomposition of IMSE into bias and variance. This decomposition yields two useful special cases of the IMSE criterion. One, the Integrated Model Variance (IMV) is applicable when the model is correctly specified, in which case the bias term disappears. The other, Integrated Squared Bias (ISB) is applicable when variance due to noise is negligible.[5]

The approaches we develop are motivated by previous work, but differentiated from it in significant ways. Much previous work addresses a special case of the IMSE criterion that arises by making the strong assumption that the model is correctly specified. In this case, exemplars are selected to minimize model variance. Methods derived from this criterion often make an approximation that allows target (i.e., desired output, or "teacher") information to be ignored.[6] While this approximation is valid asymptotically as the amount of training data grows large, it can be quite poor for small sample sizes. However, when enough data is available to ensure validity of the approximations, the resulting techniques can be used to evaluate how beneficial it would be to sample the environment at particular points in the input space. Typically, these inputs are then presented to an "oracle" or "teacher," which then provides the (possibly noisy) target information necessary to obtain examples for supervised training. Such methods are known as "query-based sampling," or simply as "query" methods. These are methods for obtaining more training examples, or for pruning relatively uninformative exemplars from the training set. They cannot be used for *efficient* exemplar selection, in general, because they typically require "priming" the technique by training on a large amount of data.

While querying techniques address a much different problem than exemplar selection, they are in general complementary, and can be used in conjunction with exemplar selection. We provide an extensive review of this previous work in Chapter II, in order to place our work in the proper context. Besides providing the proper context for our study, previous work on active sampling also guides our choice of a general sampling criterion for evaluating training sets (namely, the IMSE criterion. defined in Chapter III, Section III.D.1).

We develop a method for selecting examples from clean data in Chapter IV. This

[5] By "noise," we mean uncertainty superimposed upon the target (i.e., "teacher") information. This may be either inherent (stochastic) uncertainty, or due to measurement error. These terms are made precise in Chapter III.

[6] Exact calculation of the posterior model variance requires target information. This is because target information is involved in the estimate of the curvature matrix of second derivatives of the error criterion (a.k.a. the "Hessian"). When the curvature matrix is well estimated by a matrix obtained solely from first derivatives, an approximation can be made such that the equation no longer depends upon the target information (cf., [37, 96, 73, 72]).

development employs another special case of the IMSE criterion that arises from assuming that the variance is negligible. This results in the Integrated Squared Bias criterion (ISB) which is appropriate when the training examples are clean (that is, essentially deterministic or noise-free). From this objective sampling criterion, we derive a selection criterion for a class of estimators, called ΔISB. The basic idea is to choose the training example that gives the biggest estimated decrement in ISB. The procedure starts by selecting the data point from the sample that minimizes the ISB (which turns out to be a straightforward calculation), fitting a model, and then using this model to find the next best exemplar. In this way, the model allows us to estimate the change in ISB for each candidate example. We simply pick the example that maximizes this change. We then fit a new model to the new set of exemplars, and repeat until we achieve an acceptable error level.

In Chapter V we present the results of benchmark studies with the ΔISB technique. We have found that the resource demands of the ΔISB method are much less than that of traditional methods (e.g., random sampling) on the example problems we have tried. We also found that the number of exemplars necessary to learn these example problems was a small portion of the total number of available training examples. We also found a direct relation between the number of exemplars and the number of weights in the network model necessary to learn the desired mapping, such that the number of exemplars required was equal to a small constant times the number of network weights. This result corroborates the theoretical results mentioned above that tie the number of examples necessary to learn a mapping to the complexity of the network necessary to obtain the desired fit [64, 9, 10, 44, 45, 71]. Our benchmark studies also demonstrate how a significant class of learning tasks can be cast into a form in which this technique applies. This approach exploits the well-known method of learning a single-valued time-series by embedding into a multidimensional space. A successful application of the embedding procedure results in training examples that are essentially noise-free [58]. Note also that learning tasks which use an autoassociator network (that is, learning an identity mapping) train using clean training examples (cf. [30], where a bottlenecked identity mapping is used for image compression, and [32], where it is used for dimensionality reduction). Therefore, while developed for learning deterministic mappings, the ΔISB selection technique is applicable to some important practical problems.

In Chapter VI, we consider applying exemplar selection to learning tasks in which the model bias and variance are to be minimized simultaneously. This is the typical situation when the training examples are noisy and the correct model is unknown. To lay the foundation for developing an IMSE-based method for this purpose, we prove that two commonly used estimates of generalization[7] deliver estimates of IMSE that are accurate and precise, in a statistical sense. Given reasonable conditions, this result is applicable to nonlinear regression in general. While there are several previous results which obtain similar results for particular types of models or learning tasks, this is the first appearance of a general result of this form in the statistical and econometric literature that we are aware of.

Finally, in Chapter VII we present our results to date with an IMSE-based method for pruning, as well as selecting, exemplars from noisy data. Although the method is still evolving, preliminary results are promising. Currently, this work has developed to the

[7]These estimates are more commonly referred to as "cross-validation" or "cross-validatory" measures. In particular, we consider the "hold-out set" and "delete-one" cross-validation measures.

point where we can illustrate its behavior on a simple learning task, and demonstrate that it displays satisfying behavior on this toy problem. In brief, we found that active exemplar selection can be adversely affected by outliers in small samples, and were able to counteract this by iteratively pruning away outlying exemplars. The results clearly indicate that it is possible to improve the accuracy and reliability of neural network training on noisy data by judicious selection (and pruning) of training examples. We leave it to future work to develop the method more fully and evaluate it on practical problems.

Chapter II

Review

This chapter provides a review of subject matter addressed in this dissertation, with pointers provided for the reader interested in more comprehensive treatment.

We begin this chapter with a review of work done relating the number of examples with generalization. Next, we review much of the work done on active sampling. We then provide a comprehensive review of the work done on active selection to date. Some of our results have scope extending into more general issues in statistics, in particular, estimates of generalization, and robust statistics. Therefore, we provide pointers to the large amount of literature regarding cross-validatory measures, the Integrated Mean Squared Error measure, and outlier detection. Finally, we compare our results to the current state of the art, to place them in their proper context.

II.A Sample Size Bounds

There are several theoretical results relating the rate at which generalization error of a fixed size network decreases as the number of examples grows large. These rates tell us a great deal about the data requirement of neural network training. In particular, these rates can be used to obtain bounds on the number of examples sufficient to attain a particular level of generalization in a certain sense. We can also say something about the minimum number of examples necessary for generalization to occur.

In [64] theory is provided demonstrating the statistical properties of the information-theoretic technique known as "Minimum Description Length." This work evaluates the ensemble behavior of networks of a fixed architecture to derive asymptotic learning curves, demonstrating how generalization depends upon the number of examples. Most pertinent to our discussion is the theoretical demonstration that the benefit of an additional example falls off exponentially as the sample size increases.

For a network implementing a given class of functions, trained from examples obtained independently from identical distributions (i.i.d.), and for a desired level of accuracy ϵ to be obtained with probability $1-\delta$, we can give an upper bound on the sufficient number of examples required for training [9, 10]. While most of these bounds are obtained by worst-case analysis, [45] does a worst-case analysis using Vapnik-Chervonenkis (VC) dimension, as well as an average-case analysis using Bayesian statistics and information theory. It also provides another theoretical demonstration that the "average instantaneous information gain" from a single additional example is decreasing.

For concreteness and convenience to the interested reader, we derive analogous results directly applicable to our situation. See Appendix A for derivation of a bound on the sufficient sample size, and Appendix B for a derivation demonstrating the exponentially decreasing benefit of a single additional example, using results from [111]. (See also the related discussion in Section I.D.)

The most pertinent results regarding the benefit of active querying can be found in [76, 98]. These results address the issue more directly, by evaluating how the estimation error decreases with each additional example. The results in [76] provide convergence rates for single hidden layer feedforward networks, applied to learning smooth functions. They demonstrate that in this case the estimation error approaches $O_p(n^{-1/2})$. The results in [98] are obtained from a different perspective, however, they corroborate this algebraic decrease in the estimation error for randomly selected examples, and demonstrate that judicious selection of examples can result in an exponential decrease in estimation error. We discuss the work of [98] in more detail below.

II.B Active Sampling

Active sampling is essentially concerned with determining the distribution of training examples, which we refer to as the "sampling distribution." The modified sampling distribution determined by active sampling may differ from the "environmental distribution" according to which examples occur naturally in the learning environment.

Judicious sampling must determine the sampling distribution by utilizing prior knowledge about at least one of three objects: the model (i.e., the neural network architecture), the learning algorithm (largely determined by the error criterion used for training, and the learning rule by which this error is minimized), and the learning task (which includes the environmental distribution and any prior knowledge of the target function). Given enough knowledge about the learning task and the network model, it is possible to determine the optimal sampling distribution in advance, without need for active sampling [6, 37].

Active sampling provides the means for querying the environment for new information, for instance, to fill in the gaps in a finite sample due to random deviation in the location of examples, or, to verify critical information. Active sampling approaches exploit knowledge gleaned from examples given so far to determine *regions of uncertainty.* Active sampling determines the sampling distribution according to the result of learning upon previous examples, and so in this sense, the network is an "active" participant in its training.

Typically, the task is to minimize the number of exemplars one has to gather, or to corroborate information crucial to the estimate. This is desirable when data measurements are costly, or when there are critical gaps in the data due to sampling variation. (The techniques used to this end may also be used to prune examples from the training set after training is completed; however, this is a form of active selection, and will be discussed below.)

Historically, active sampling has been referred to variously as "active learning," "sequential design," or "query-based learning." In a loose sense, we can think of a concise set of exemplars as analogous to the notion of "extreme points" employed by Cover for classification tasks [31]. Active sampling is often called *query learning* as it "queries" an "oracle" for values of the function in these regions[1] (cf. [50, 26, 27, 113]).

First, we survey some formal results. Secondly, we review a smart stochastic sampling method. Third, we review applications where knowledge of either the learning task or the model (or both) is exploited. Finally, we review methods which minimize model variance.

II.B.1 Formal Results Regarding Generalization

Query-based deduction programs are studied in the computational learning theory literature [107, 40, 46, 61]. Hellerstein and Karpinski study exact learning of a class of boolean tasks. They use a combinatoric characterization of the learning task to prove that learning by queries to an oracle permits feasible time algorithms for this class of problems, whereas learning these tasks from examples drawn at random is not possible in polynomial time[2][46]. In some formal situations, query-based learning is provably more powerful (in computation theoretic terms) than passive inductive inference (see for example, [40], which demonstrates that learning from responses to queries provided by an oracle extends the set of functions which can be inferred by inductive inference).

This previous work provides theoretical justification for the use of active sampling for reducing the computation required by learning, or for extending the class of problems which can be learned. However, none of the results provide useful algorithms. Baum does propose a query-based learning algorithm for iteratively seeking out classification boundaries for which a convergence proof is supplied [8]. However, all of these results apply to boolean tasks, whereas we are interested in the more general task of learning arbitrary mappings.

II.B.2 Formal Results for Bayesian Querying

Sollich provides theoretical results indicating that active querying can give a significant benefit [98]. A Bayesian analysis is used, thereby assuming prior knowledge of the learning task is available. Prior knowledge is required of the prior distribution of the target functions, and as well, for some of the results, of the correct model complexity required to parameterize the target function. This allows analysis of the benefit of a *particular* set of exemplars, as the formulation allows one to formally state the probability of being provided with particular set of exemplars by the "teacher."

Results are obtained for three particular applications, the first being a simple (although nonlinear) classification task, the second being the task of training a perceptron, and the third, a task where model complexity is not known in advance.

First, we discuss the results for the linear perceptron. In the limit of an infinite number of training examples, querying does no better than random sampling at minimizing generalization error. However, querying does provide a significant benefit when training on finite size training sets. Moreover, the improvement in generalization due to querying depends greatly upon the ratio of exemplars to the number of weights. The improvement due to querying increased dramatically as this ratio rose from 0 to 1, peaking shortly thereafter, and then more slowly decreasing as the ratio increased. (This gives some indication of the

[1] An *oracle* answers with a (possibly noisy) target in response to being "questioned" with a particular input.

[2] This holds if $RP \neq NP$, which is a conjecture that many computational theorists believe holds true. Here, RP is random polynomial space, and NP is nondeterministic polynomial space.

conciseness of exemplar sets selected by a method developed in this dissertation, which resulted a ratio of exemplars to weights of just over 1.0 for a learning task involving a nonlinear model.) The improvement due to querying in Sollich's experiments was greatest when the noise was lowest (that is, the signal to noise ratio was greatest) indicating that active selection becomes a more difficult task as the level of noise increases.

Now we turn to the results for the simple nonlinear task, where parameterization of the target is known in advance. Sollich points out the important fact that querying algorithms can be derived according to two general classes of objective criterion, one maximizing information gain, and the other minimizing generalization error. It is demonstrated that a query algorithm designed to minimize the information-theoretic difference between student and teacher will not, in general, perform as well (in terms of reducing generalization error) as an algorithm designed to minimize generalization error. (See also [112].) However, both approaches cause generalization error to decrease exponentially (whereas it decays algebraically otherwise). (See also [76] for results more applicable to our situation demonstrating an algebraic decrease of generalization error with increasing training set size.)

Finally, Sollich considers the case where model complexity is not optimal. In this case, it is possible for a querying algorithm to be detrimental. This can occur when training examples are very noisy, and the number of exemplars is small. In these cases, the querying algorithm can be *overconfident* in its selection, which can ultimately lead learning astray in presence of noise. (Compare this with our similar findings in Chapter VII.) However, in all other cases, querying algorithms will give better generalization in the limit (as the number of exemplars grows large), as well as for finite size training sets, in which case the relative improvement in generalization due to a single judicious query is, in general, greater than that due to a single additional random example. Again, the improvement due to querying was greatest when the ratio of exemplars to weights was near 1.0 (interestingly, with smaller peaks at 2.0, 3.0, and 4.0, in the experiments with low noise). These results give strong support for the development of active sampling methods, although they also warn that it can become a much more challenging task in the presence of noise.

II.B.3 Stochastic Sampling

One approach to sampling is to sample randomly at first, until either the environmental distribution of examples is well-estimated, or the form of the desired function is becoming apparent, and sample actively thereafter. In this approach, the sampling distribution is dynamically determined from information acquired through prior training (i.e., is "data-adaptive.")

One example of this is the stochastic sampling approach [113]. (Somewhat related is [50], where knowledge of the model is exploited after likewise training on an initial set of examples obtained "passively.") Previous work commonly compares this approach to stochastic sampling with the task of winning at a game of slot machines involving a possibly infinite number of machines. The task is to determine a sampling distribution maximizing payoff. The heuristic is to revisit previously pulled "slot machines" according to their success, and intermittently visit new machines nearby previous successes. They derive a sequential procedure for seeking the minimum (precisely, the essential minimum, in probability) of a function on finite-dimensional euclidean space. This technique provides an active sampling technique for querying the environment in the presence of noisy measurements for training a black box model. It does require "priming" by training the model upon a small sample obtained "passively" before commencing with active sampling.

A convergence result is provided. The proof requires an ergodicity assumption, implying that the sampling distribution is asymptotically dense. Therefore, the computational demands of the technique in practice require more investigation before we would recommend this technique for general purpose use.

We provide this reference as an excellent example of an active sampling technique which could address the issue of ensuring sufficient coverage of the example space.

II.B.4 Exploiting Knowledge of the Application

A common theme in query learning is to use a partially trained network to determine *regions of uncertainty* in the environment and then query the oracle for values of the function in these regions. The differences among the methods lie in how "uncertainty" is quantified. Here we several three approaches where task knowledge is exploited to optimize the sampling distribution.

Text Classification

In [65] text classification is defined as the automated grouping of textual entities. In this application, text is cheap, but the process of determining what classes an example of text belongs to is expensive. A large number of "unlabeled" examples (those for which no target information is yet made available) are generated. A subset of these are then presented to an oracle for labeling. The algorithm was able to reduce the amount of data that needs to be classified (which here, is done manually) by as much as 500-fold. This article also provides a comprehensive review of other work done applying querying to this learning task.

Visual learning tasks

Visual learning tasks lend themselves to intuitive heuristics which exploit concepts of visual attention inspired by cognitive modeling. See for instance [3], where the system decides which portions of the image are relevant, and then acquires information from the selected portions. This approach gives very efficient algorithms for solving certain learning tasks involving clustering analysis. They use local receptive fields for rapid adaptation. However, the task performed is essentially a toy problem. The learning task is well-defined, as is the network model. The method of selecting data points exploits regularities in the learning task as well as in the network model to obtain data which is intuitively most informative to the network learning rule. It is not immediately evident how to extend this intuitive heuristic for general purpose learning.

Yamada and Cottrell have developed this approach significantly for the purpose of face recognition [114]. Yamada uses Gabor transformations on image data to produce features which can be localized to particular regions of the image. Then, a heuristic chooses features which are most informative to the recognition algorithm. The result is a face recognition algorithm which operates without needing to process all of the available data, by processing selected subregions of the image. (Because much of the sampling distribution is determined in advance, active sampling plays a minor role in this approach.)

Robotic Task

Consider the problem of learning input/output mappings through exploration, e.g. learning the kinematics or dynamics of a robotic manipulator. If actions are expensive and computation is cheap, then we may explore by selecting a trajectory through the input space which provides the most information in the fewest number of steps. In [28] results from the field of optimal experiment design are used to guide such exploration, and demonstrate its use on a simple kinematics problem.

Finding Class Boundaries

One problem common among sampling techniques is that the cost of computing optimality over the input space scales exponentially with the dimension of the input space. An alternative is to selectively sample regions of the input space. A query-based approach for neural network learning of classification tasks that generates exemplars in the vicinity of classification boundaries is given in [50, 51]. This technique is intended for use on a partially trained "classifier" for learning a discrimination task. (This is often referred to in neural network literature as "classification," although that term is used in statistical parlance to refer to what is known in neural network literature as the unsupervised learning task "clustering.") Hwang, et.al. obtain a useful sampling technique by (implicitly) assuming a uniform distribution over the input space, using a heuristic that determines the classification boundary of the network by inverting the network function. It is assumed that boundaries are regions where additional information would most directly test the network fit. Let f represent the network function, and let $f : \Re^r \mapsto [0, 1], r$ integer. Then, the classification boundary corresponds to those inputs for which the network output is 0.5. Examples are selected from the set of inputs in $f^{-1}(0.5)$. Intuitively, these examples correspond to regions of maximum classification ambiguity. After the network is well trained, this technique can be fine tuned by considering the sensitivity of the output with respect to each input. Then, the technique chooses input examples for which a slight perturbation of the example causes a large perturbation in the network output.

Although intuitively pleasing, this heuristic has drawbacks. It is not clear how well it would perform with an arbitrary distribution of examples in input space. The heuristic relies heavily upon the structure of the learning task. Examples selected accordingly are not necessarily informative, because while it seems plausible that the method would excel at refining existing model boundaries, it is not clear how well it would perform at locating new class boundaries. While appealing for use on discrimination tasks, this intuitive heuristic does not admit an immediate generalization to learning arbitrary mappings, because no explicit objective function is optimized.

General-Purpose Boolean Discrimination

For a principled approach to selecting examples informative for learning a discrimination task, see [72, 73]. There, bayesian statistics are used to obtain an information-based querying strategy. The performance of the resulting technique corroborates the intuitive strategy of concentrating queries near classification boundaries. The major drawback of the resulting techniques is that it is necessary to assume that the model is the correct one for the task, which is translated into our situation as meaning that the network must have the appropriate complexity [73]. These methods have direct analogs with methods developed previously (using a sampling theoretic approach[3]) in the theory of *response surface methodology*, *optimal experiments* and *sequential design* [37, 21, 59], which we review in the next section. The same capabilities and limitations are inherent to both sets of techniques, and are discussed in the following section.

II.B.5 Minimal Model Variance Techniques

Using queries for the general task of estimating an arbitrary, unknown mapping has been addressed earliest and most extensively in the statistics and econometrics literature, in the theory of *response surface methodology*, *optimal experiments* and *sequential design* [37, 81, 35, 21, 59]. These approaches typically utilize the IMSE sampling criterion (defined in Section III.D.1) for the purpose of sampling the environment for new training examples [6, 29, 81, 83, 96]. Fedorov gives the most complete treatment, and obtains several querying methods for performing interpolation and extrapolation from noisy real valued data, including a method for obtaining new data which is maximally informative with respect to comparing competing models. These approaches require a correctly specified model, which is a rather strong assumption. Given sufficient training, this assumption allows the model bias (approximation error) to be ignored, resulting in techniques primarily devoted to minimizing some measure of model variance (estimation error).

These results have been extended to consider nonlinear models, but the extensions are subject to stringent conditions (typically imposed to assure that a linearized version of the model can be obtained to allow the application of the techniques, which are developed for linear models). These conditions translate into our situation as meaning that the network must have (likewise) the appropriate complexity for the task, and also must be trained to within a stringent tolerance to assure that linearization of the model yields a good approximation.

These minimum-variance techniques can also be used to prune away examples after training is completed. In our terminology, this is a form of active selection, and will be discussed in the next section.

II.C Active Selection

Active selection starts with a given set of examples, and selects a subset to use as training "exemplars," exploiting information obtained from the model due to learning from previous exemplars. It relies upon the environmental distribution to generate a set of candidate examples large enough to ensure proper coverage of the example space. Active selection is used to deal with the redundancy of an oversampled set of examples to reduce the size of the training set. Previous work on active selection is limited; however, the work that has been done corroborates our experience that active selection techniques are feasible and beneficial.

[3] As opposed to bayesian statistics, the sampling theoretic approach strictly defines probability as the limit of relative frequencies determined by a sequence of samples.

II.C.1 Stochastic Sampling

Here we are interested in selecting an informative subset of the available examples. Another approach, appropriate for use with stochastic approximation[4] (where weight updates are performed after presentation of each example, rather than averaging weight updates over the entire set) is given in [82, 24]. In particular, it gives a way of ordering the presentation of the training examples during learning. The method given in [24] appears to lie somewhere between the typical method (of cycling through the entire training set at each iteration, permuting the order at each iteration) and the method given in [82] (of training only on examples for which network error exceeds a preset threshold). Both methods adapt the frequency of presentation according to the state of the network, and therefore, can be viewed as active selection strategies.

II.C.2 Version Space Technique

Cohn examines a general framework for learning boolean mappings [26, 27]. The task is to correctly "classify" novel points drawn from a space X according to a distribution μ (that is, to learn to discriminate among different classes). When learning from examples drawn according to input distribution μ, the probability that a new example allows a reduction in error is exactly the probability that the current hypothesis will misclassify the example. We would clearly prefer to select points which can provide useful information with higher probability. Specifically, we may choose examples lying within a known region of uncertainty. Such examples are those for which there exist two different hypotheses (in the current hypothesis space) which are consistent on all previous examples, but which differ on the new example. (In network terms, this is equivalent to having two networks which classify all previously encountered examples correctly, but which differ on a new example.) An example drawn according to μ would be presented to the oracle for classification only if it has this property.

This is referred to as a *version space* technique. Intuitively, it is analogous to playing "twenty questions" because the version space of models that are consistent with the given set of data is reduced with each new example. This approach assumes that the acquisition of an example occurs in two stages. An input location is obtained by random draw on X according to μ. Then, the location is presented to an oracle for classification. The algorithm estimates the worth of a particular example relative to other available examples, therefore, it is a way of selecting exemplars from a set of available examples. Furthermore, the method does not modify the sampling distribution (since the oracle accepts inputs according to the environmental distribution, μ). Therefore, this can be viewed as an exemplar selection technique.

While this work provides useful algorithms, they are, in general, computationally intensive, and they are intended for boolean mappings, whereas we are interested in the general case. However, the results do demonstrate that sampling techniques can provide useful algorithms for network learning.

[4] This is often referred to in neural network parlance as "online backpropagation"

II.C.3 Minimum Variance Pruning

Note that the methods mentioned above for pruning away examples after training are essentially active selection methods, since they are a way to select a subset from a set of available examples [37, 83, 96, 72, 73]. This technique can be used to evaluate the importance of exemplars after training is completed. A certain subset of the training examples may be critical to the form of the final estimate obtained from the initial set, in that a small amount of noise applied to these examples can have significant impact on the network estimate [37, 83, 96]. In this sense, the approach is concerned with determining the sensitivity of the network estimate to particular training examples. Given knowledge of this sensitivity, we may either resample particular examples, or, prune certain examples from the training set.

II.C.4 Nonparametric Application

While much of the work above has been applied to nonlinear models, is developed largely for linear models, or to discrimination learning tasks. If it is known that the function to be learned is reasonably smooth, a criterion exists for choosing examples for a nonparametric, nonlinear, kernel-based smoothing approach (see [35], where Faraway uses the IMSE criterion for sequential selection of training examples). However, a (possibly large) amount of data is necessary to "prime" the technique in order to regulate model complexity.

Faraway uses a kernel-based method to achieve nonparametric estimation of continuous mappings. (This is a localized approximation, where each kernel gives essentially local contribution to the global mapping. Although applicable to a wide class of kernel types, much of this work uses models composed of gaussian basis functions.) Previous work [81], indicates that if the kernels all use the same global "bandwidth" (intuitively, the kernel "size", which for a gaussian basis function corresponds to the width of the bell-shaped curve) then asymptotically the optimal distribution of examples is evenly spaced. Furthermore, if some degree of local smoothing is applied to the data, then the optimal distribution of examples (in input space) is a function of the roughness of the target function g.

Faraway investigates the possible benefit of actively sampling training examples for the purpose of training a kernel estimator using locally variable kernel bandwidths, that is, where the size of an individual basis function can vary. Faraway concludes that local smoothing must be used to allow the actively selected data to outperform the evenly spaced distribution when globally constant bandwidth kernels are used. Hence, Faraway focuses on deriving an active selection technique for training a kernel regression model with variable kernel bandwidths, using a data-adaptive method for selecting the local kernel bandwidths,

A crucial element of Faraway's approach is the selection of the global bandwidth, as a starting point for selecting good local bandwidths. In our context, selection of the global bandwidth is analogous to selecting the network complexity. First, an initial sample of size n is used to select a good global kernel bandwidth. This is done by choosing the global bandwidth minimizing IMSE over the initial sample. Then, the local bandwidths for each kernel are obtained by using an estimate of the second derivative of the target function g. This calculation utilizes an asymptotic expansion of IMSE.

The optimal sampling distribution is known to be proportional to $(g^{(2)})^{2/9}$ [81]. By comparing the active sampling distribution with the optimal asymptotic distribution, Faraway concludes that variable kernel bandwidths must be used for this technique to have an advantage over an evenly spaced distribution of examples. By demonstration upon simple examples, Faraway illustrates that the sampling method does approach the optimal distribution in the limit.

We note that the estimate of the local bandwidth requires estimation of the second derivative of g, therefore, g must be twice continuously differentiable. Note that the task of correctly specifying the network is still present, although not as a prior assumption, but as a data-driven aspect of the technique. This requires priming the technique by training over an initial i.i.d. sample. Thereafter, the training set examples can be selected according to the result of training over previous examples. Once $g^{(2)}$ is estimated, to calculate the local bandwidth requires evaluation of the estimate of $g^{(2)}$ over all available examples. Also, the new example must be selected from a set of available candidates, and so, in our terminology, this is an active selection technique.

II.C.5 Exemplar Selection for Clean Data

In [93] an algorithm is developed for actively selecting exemplars from clean data, using a maximum-error criterion. This approach is the result of analyzing the relation between approximation and generalization. Cross validation error over a hold-out set is used to estimate generalization. Selection occurs when the cross validation measure exceeds the network error over the current set of exemplars. At this time, an exemplar is selected having maximal error.

The method is applied to two time series prediction tasks. The total computational cost of training is compared with the cost due to using online backpropagation (that is, the stochastic approximation version of steepest descent where weights are updated after presentation of each exemplar). Röebel's method required only 30 percent of the computational cost required by training with online backpropagation.[5]

As might be expected, this algorithm is adversely affected by noisy outliers in small samples, although it reportedly continues to work well when the noise is small relative to the training error [94]. No quantitative results are provided in this regard. However, the experiments on noisy training data were conducted using complex musical signals, validating the reliability of this active selection method on a challenging practical application.

II.D Cross-Validation and IMSE

To lay the foundation for IMSE-based selection of noisy exemplars, we prove that two common versions of cross-validation (typically used by practitioners to estimate generalization - i.e., network error on novel examples) are, in a statistical sense, accurate and precise estimates of IMSE (see Chapter VI). We now provide references to related results in the statistical and econometric literature.

Cross-validatory measures have been used successfully to assess the generalization performance of a wide range of estimators [99, 102, 19, 20, 75, 108]. Cross-validatory measures have been derived for various performance criteria, including the Kullback-Liebler Information Criterion (KLIC) and the Integrated Squared Error (ISE, asymptotically equivalent to IMSE) [43]. Although provably inappropriate in certain applications [95, 25], op-

[5]The online backpropagation experiment was conducted by an independent researcher in a competitive situation [94].

timality and consistency results for the cross-validatory measures have been obtained for several estimators, including linear regression, orthogonal series, splines, histograms, and kernel density estimators [43, 18, 103, 66, 67, 106, 5]. The author is not aware of similar results applicable to neural networks, although two more general, but weaker results do apply [101]. A general result applicable to neural networks shows asymptotic equivalence between cross-validation and Akaike's Criterion for network selection [100, 68], as well as between cross-validation and Moody's Criterion [80, 68].

Although our results prove that two cross-validation measures provide estimates of IMSE that are asymptotically accurate and precise, cross-validatory estimates can exhibit high variance in small samples. In this situation, one might be better served by resampling methods which, although computationally intensive, may exhibit lower variance in small samples. The statistical properties of these methods, as they relate to neural networks, are studied in [91].

II.E Outliers and Pruning

We brush up against the topic of "outliers" in our preliminary work on developing a method for selecting exemplars from noisy data. The study of "outliers" is outside the scope of this dissertation. We consider this term to have many different connotations and interpretations. The reader interested in this topic may begin with the study of *robust statistics* given in [49], as well the references provided in [56, 22, 69]. As a small peek inside this actively studied topic, we provide two distinct views on the matter. The following quote depicts outliers as the result of finiteness of the sample:

> The term outliers does not mean that they are not part of the joint data probability distribution or that they contain no information for estimating the regression surface; it means rather that outliers are too small a fraction of the observations to be allowed to dominate the small-sample behaviour of the statistics to be calculated. With parametric regression modelling techniques it is easy to quantify this effect by simply computing the effect that each data point has on the regression surface. This is not a trivial problem in non-parametric modelling but the statistics literature is full of methods to deal with it [92].

The following perspective views outliers as a result of the data generating process:

> Outliers are generated from a distribution that is a perturbation to the underlying distribution, for example, a small amount of noise with ever changing distribution in the background [69].

Yet another view is provided in [56], which uses a maximum-likelihood definition of "outliers." In our work, although we refer to pruned exemplars as "outliers," we simply excise exemplars that are misleading, according to a estimate of how much an estimate of generalization is improved by deleting them. We do not claim our current method of pruning noisy exemplars to be appropriate for excising outliers as defined in either of the preceding two quotes or in any of the accompanying references. However, our active pruning method is an IMSE-approximate method, a property that provides justification for its use.

II.F Relation to the State of the Art

Our approach is similar to many previous approaches, in that examples are chosen according to information derived from a partially trained network. We consider a general learning task, namely, estimating an unknown functional mapping. In this regard, our methods are closest to Faraway's technique. Using a formulation applicable to general purpose nonlinear regression, we adapt techniques for the purpose of training multilayer feedforward networks.

We have automated this for the case of noiseless data, inspired by minimum description length principles and Bayesian formulations. However, our approach is distinguished from the Bayesian formulations, which are typically sensitive to misspecification of the model (that is, they rely strongly upon selection of the optimal model complexity) and require assumptions on the prior distribution of the target function. The latter can be a useful feature when prior knowledge of the target function is available, however, our approach is nonparametric in the sense that the complexity of the network need not be pre-specified. Parameterization of the network function is necessary to evaluate training examples, but modifying the number of network parameters (weights) between successive training examples is allowed.

We extended these methods for the case of noisy training data by minimizing cross-validatory estimates of IMSE. We are motivated by the theory of response surface methodology, optimal experiments and sequential design in working with the IMSE sampling criterion [37, 6, 21, 59, 29, 83, 96]. We were initially inspired by the work on selective attention (cf. [3]), in that focusing the computation upon a set of judiciously chosen data we may reduce computational demands of processing the data. While the success of previous application-specific work is encouraging, our objectives are aimed at general purpose learning, rather than a particular task.

Our objective is to reduce the data requirement of learning towards reducing the computational demands of learning. We have chosen to study the task of selecting a subset of available candidate examples to be used for training. This approach does not address the issue of resampling the environment, and so it assumes that we have a sufficient amount of data. Any of the active sampling methods discussed above could be used in complementary fashion to ensure this.

While Röebel's method (see [93]) may be less expensive computationally in certain circumstances (because it does not require computation of derivatives of the network function) the cost of the method may become significantly more expensive as the number of available examples increases (this is due to the *online* validation of generalization error over the entire set of available examples during training, whereas our method accesses the set of available examples only when a new exemplar is being selected). Furthermore, the ΔISB method provides the ability to automatically adjust network complexity during training, and yields more concise exemplar sets on the Mackey-Glass time series learning task. Our results also indicate that a maximum-error based method is adversely affected by a more challenging (albeit, still noiseless) learning task, whereas ΔISB continues to work reliably.[6] When trained to predict a difficult chaotic time series, a maximum-error based method was

[6] In particular, maximum-error based methods will select an exemplar maximizing the training error measure even if the example is representative of only a small portion of the entire space of training examples. Consider, for instance, a function with a small spike, or with a region of radical variation falling in a region of space where the density of the inputs is very low relative to the rest of the input space.

prone to failing to converge to the desired accuracy of fit. These results are presented in Chapter V.

The benchmark comparison of Röebel's method with online backpropagation provides an important data point indicating that active selection can be less costly than online backpropagation, which is considered by many practitioners to be more efficient on highly redundant training sets than the "classical" optimization methods[7] considered in this dissertation (cf. [78, 79]).

The results of [98] give some indication of the conciseness of exemplar sets obtained experimentally using our method, because Sollich's results demonstrated that the benefit of querying, in terms of the ratio of the generalization provided by a certain number of exemplars obtained by querying to the generalization obtained by the same number of exemplars obtained randomly (according to the environmental distribution) was maximized when the ratio of the number of exemplars to the number of network weights was just over 1. Our experiments demonstrate that learning by active selection resulted in a ratio of exemplars to weights of just over 1 (see Chapter V). Also, Sollich's results demonstrate that, in case of noisy examples, inappropriate application of querying can be detrimental, in which case the querying technique can be viewed as being "overconfident" in its ability to estimate the benefit of a candidate exemplar. This sheds light on the behavior of our method as applied to noisy examples (see Chapter VII), which we found to exhibit a similar phenomenon.

Our approach is most similar to [35], in the definition of the learning task being addressed, as well as in the use of the IMSE criterion for the evaluation of training examples. See Chapter III for the definition of the learning task we consider, which is similar to the learning task addressed by Faraway, and also for the definition of the IMSE criterion.

Along the way, our work touches on the issue of generalization, justifying the use of cross-validation for estimating a particular measure of generalization, and relating this measure to other measures of generalization more commonly referred to in the neural network literature. Finally, our preliminary work in obtaining a method for selecting exemplars from noisy data raises the question of what it means for an exemplar to be an "outlier" in a small sample, a concept addressed in more depth in the study of *robust statistics*. More work is required to evaluate the strength of the connection between our work and the results of robust statistics. Work also remains in refining and evaluating the performance of our active selection method for noisy data.

[7]These are known in the neural network literature as "batching" techniques, since weight updates are obtained by averaging the gradient of the squared error (with respect to network weights) over the entire training set. Examples include Newton-Raphson, Gauss-Newton, and conjugate gradient descent.

Chapter III

Definitions

III.A Learning Task

We consider the learning task of determining the relationship between a random vector X and a random scalar Y, where X takes values in a subset $\mathbf{X}$ of $\Re^r$, and Y takes values in a subset $\mathbf{Y}$ of $\Re$. (e.g. $\mathbf{X} = \Re^r$, for r a positive integer and $\mathbf{Y} = \Re$). We refer to $\mathbf{X}$ as the *input space*. We also refer to $\mathbf{Y}$ as either the *output* or *target* space, because both the training targets and the network outputs lie in this space. The learning task is thus one of training a neural network with r inputs and one output. In what follows, we will use the prime "$'$" to denote transpose. It is straightforward to extend the following analysis to networks with multiple outputs (and hence, trained with multiple targets).

The following assumption is on the observations to be used in the training of the networks.[1]

Assumption 1 $\mathbf{X}$ *is a Borel subset of* $\Re^r$ *and* $\mathbf{Y}$ *is a Borel subset of* $\Re$. *Let* $\mathbf{Z} \equiv \mathbf{X} \times \mathbf{Y}$ *and* $\Omega \equiv \mathbf{Z}^\infty \equiv \times_{i=1}^{\infty} \mathbf{Z}$. *Let* $(\Omega, \mathcal{F}, \mathcal{P})$ *be a probability space with* $\mathcal{F} = \mathcal{B}(\Omega)$, *the Borel* σ*-field generated by the measurable finite dimensional product cylinders. The observations on* $Z \equiv (X', Y)'$ *to be used in the training of the network are a realization of an independent identically distributed (i.i.d.) stochastic process* $\{Z_i \equiv (X_i', Y_i)' : \Omega \to \mathbf{X} \times \mathbf{Y}\}$.

When $\omega \in \Omega$ is fixed, we write $z_i \equiv Z_i(\omega)$ for each $i = 1, 2, \ldots$. Also write $Z^N \equiv (Z_1, \ldots, Z_N)$ and $z^n \equiv (z_1, \ldots, z_n)$. [2]

We denote a particular *sample*, or, *training set* of size n as $z^n = (z_1, \ldots, z_n)$. We suppose that the measurement process could be continued indefinitely, to obtain a sequence of observations $\{z_i\} = (z_i,\ i = 1,2,\ldots)$, as a realization of the random sequence $\{Z_i = (X_i', Y_i)'\}$, where X_i and Y_i have the distributions of X and Y respectively for all $i = 1,2,\ldots$. We denote a *random* training set of size n as Z^n.

Assumption 1 allows uncertainty caused by measurement errors of observations as well as a *probabilistic*[3] relationship between X and Y. It, however, does not prevent

[1] It stipulates that the training examples must be obtained independently from the same distribution, and that the probability spaces used to model the learning task are well defined.

[2] We may associate ω with a sequence of particular realizations of the random variables, so that $\omega = \{z_i\} = \{Z_i(\omega)\}$.

[3] i.e., "noisy"

a deterministic[4] relationship between X and Y such that $Y = g(X)$ for some measurable mapping $g : \Re^r \rightarrow \Re$. [5]

In our simulations we will have complete control in determining a particular point in input space, x_i. We define a (fixed) "environmental" probability measure μ that describes the relative frequencies with which different values for x_i are set; $\mu : \mathcal{B}(\Re^r) \mapsto [0,1]$. $\mathcal{B}(\Re^r)$ is the Borel σ-algebra generated by the open sets of $\Re^r$. We will also refer to a probability measure ν, that describes the relative frequencies with which different values of z_i are set, and a conditional probability measure γ that describes the conditional relationship between X_i and Z_i, such that γ describes the distribution of the targets Y_i given X_i.

We suppose interest attaches to the conditional expectation of Y given X, written $g(X) = E(Y|X)$.[6] The next assumption guarantees the existence of $E(Y_i|X_i)$ and $E(\varepsilon_i|X_i)$, $\varepsilon_i \equiv Y_i - E(Y_i|X_i)$, as well as the existence of the variance of ε_i.[7]

Assumption 2 *$E(Y^2) < \infty$.*

For convenience, we next assume homoskedasticity of the conditional variance of Y_i given X_i. [8]

Assumption 3 *$E(\varepsilon_1^2|X_1) = \sigma^2$, where σ^2 is a strictly positive constant.*

III.B Network Model

Let $f^p(\cdot,\cdot) : \mathbf{X} \times \mathbf{W}^p \rightarrow \mathbf{Y}$ be a network function with the "weight space" $\mathbf{W}^p$, where p denotes the dimension of the "weight space" (the number of weights.) We impose some mild conditions on the network architecture.[9]

Assumption 4 *For each $p \in \{1, 2, \ldots, \bar{p}\}$, $\bar{p} \in \mathbf{N}$, $\mathbf{W}^p$ is a compact subset of $\Re^p$, and $f^p : \mathbf{X} \times \mathbf{W}^p \rightarrow \Re$ satisfies the following conditions:*

1. *$f^p(\cdot, w) : \mathbf{X} \rightarrow \mathbf{Y}$ is measurable for each $w \in \mathbf{W}^p$;*
2. *$f^p(x, \cdot) : \mathbf{W}^p \rightarrow \mathbf{Y}$ is continuous for all $x \in \mathbf{X}$.*

We further make a joint assumption on the underlying data generating process and the network architecture to assure that the training dataset and the network behave appropriately.[10]

Assumption 5 *There exists a function $D : \mathbf{X} \rightarrow \Re_+ \equiv [0, \infty)$ such that for each $x \in \mathbf{X}$ and $w \in \mathbf{W}^p$, $|f^p(x, w)| \leq D(x)$, and $E\left[(D(X))^2\right] < \infty$.*

---

[4] i.e., "clean" or "noiseless"

[5] Measurability of g and other variables is required by the mathematics we use to ensure that the objects of interest to us are well defined random variables.

[6] In this case g represents the average relationship, the conditional expectation of Y given X.

[7] ϵ_i represents the "noise" or "uncertainty" inherent to observation i.

[8] Roughly, this means that the variance of the noise is constant over the input space. Note that this does not require that the distribution of the noise process be identical everywhere in input space.

[9] Note that this would require a continuous activation or "squashing" function for a standard feedforward net.

[10] There exists some function that is square-integrable, that dominates the network function. This allows the application of the dominated convergence theorem to establish subsequent useful results in probability and real analysis.

Hence, f^p is square integrable for each $w \in \mathbf{W}^p$. We will measure network performance using mean squared error, which for weights w is given by[11]

$$\lambda(w;p) \equiv E\left[(Y - f^p(X,w))^2\right]. \tag{III.1}$$

The optimal weights are the weights that minimize $\lambda(w)$. The set of all optimal weights are given by $\mathbf{W}^{p*} \equiv \{w^* \in \mathbf{W}^p : \lambda(w^*;p) \leq \lambda(w;p) \text{ for any } w \in \mathbf{W}^p\}$. Equivalently, this set is given by $w^* \in \arg\min_w \in \mathbf{W}^{\mathrm{p}} \int (g(x) - f^p(x,w))^2 \mu(dx)$. The index of the best network is p^*, given by the smallest p minimizing $\min_{w \in \mathrm{W}^p} \lambda(w;p)$, $p \in \{1, 2, \ldots, \bar{p}\}$.

III.C Least-Squares Estimator

When assumptions 1 and 4 hold, the nonlinear least-squares estimator exists. Formally, we have [12]

Lemma 1 *Suppose that Assumptions 1 and 4 hold. Then*

1. *For each $N \in \mathbf{N}$, there exists a measurable function $l_N(\cdot;p) : \mathbf{Z}^N \to \mathbf{W}^p$ such that $l_N(Z^N;p)$ solves the following problem with probability one:*

$$\min_{w \in \mathbf{W}^p} N^{-1} \sum_{i=1}^{N} (Y_i - f^p(X_i,w))^2 .$$

Suppose in addition that Assumptions 2 and 5 hold. Then

2. *$\lambda(\cdot;p) : \mathbf{W}^p \to \Re$ is continuous on $\mathbf{W}^p$, and $\mathbf{W}^{p*}$ is not empty.*

For convenience, we also define $\hat{w}_N^p : \Omega \to \mathbf{W}^p$ by

$$\hat{w}_N^p(\omega) \equiv l_N(Z^N(\omega);p)$$

for each $\omega \in \Omega$.[13]

Next let $i_1, i_2, \ldots, i_N$ be distinct natural numbers and let $\breve{Z}^N = (Z_{i_1}, \ldots, Z_{i_N})'$. Then $l_N(\breve{Z}^N;p)$ given above delivers weights w minimizing

$$\frac{1}{N} \sum_{j=1}^{N} (Y_{i_j} - f^p(X_{i_j},w))^2$$

with probability one. In particular, we will consider the estimate using the dataset z_{-i}^N made by deleting the ith observation from z^N. Let Z_{-i}^N be a random matrix made by deleting the ith row from Z^N. Thus, $l_{N-1}(Z_{-i}^N;p)$ is a measurable least squares estimator and we can consider its probabilistic behavior.

[11] This gives the expected squared error for a network function evaluated over all possible target-input pairs.

[12] This lemma demonstrates that there is a set of weights that minimize the least squares learning criterion, that the learning criterion is continuous over weight space, and that the set of learned weights are themselves random variables and hence subject to the laws of probability.

[13] Therefore, we can relate a *random* weight vector with the *random* training set from which it is obtained by learning.

III.D Measures of Generalization

III.D.1 Integrated Mean Squared Error

Integrated Mean Squared Error (IMSE) has been used to regulate network complexity [41]. Another (conditional) version of IMSE is used as a criterion for evaluating training examples [37, 21, 59, 35]. The first version depends only on the sample size, not the particular sample. The second (conditional) version depends additionally upon the observed location of the examples in the input space. The (unconditional) mean squared error of the network output at a particular input value x is

$$M_N(x;p) = E\left[\{g(x) - f^p(x, l_N(Z^N;p))\}^2\right]. \tag{III.2}$$

This is referred to as the "MSE." Evaluating the expectation over the input space gives the unconditional IMSE:

$$IMSE_N(p) \quad = \quad E\left[M_N(X;p)\right], \tag{III.3}$$

To evaluate exemplars obtained at inputs x^N, we modify Equation (III.2) by conditioning on x^N, giving

$$M_N(x|x^N;p) \quad = \quad E\left[\{g(x) - f^p(x, l_N(Z^N;p))\}^2 | X^N = x^N\right],$$

The conditional IMSE (given inputs x^N) is then

$$IMSE_N(x^N;p) \quad = \quad E\left[M_N(X|x^N;p)\right]. \tag{III.4}$$

Bias-Variance Decomposition of IMSE

IMSE embodies the effects of model bias (approximation error due to misspecification of the network model) and model variance (variation of the network function due to estimation error in the presence of noise). By partitioning IMSE, we motivate this interpretation of the IMSE. Given x^n, take $E\left[f^p(x, \hat{w}_n^p)|\, x^n\right]$ as the expected output of the network at input x:

$$E\left[f^p(x, \hat{w}_n^p)|\, x^n\right] \;=\; \int f^p(x,\, l_n(x^n, y^n; p))\, \gamma^n(dy^n|x^n)$$

We can partition MSE as follows:

$$\begin{aligned}
M_n(x|x^n;p) &= E\left[(g(x) - f^p(x, \hat{w}_n^p))^2 |\, x^n\right] \\
&= E\left[(g(x) - E[f^p(x, \hat{w}_n^p)|\, x^n] + E[f^p(x, \hat{w}_n^p)|\, x^n] - f^p(x, \hat{w}_n^p))^2 |\, x^n\right] \\
&= (g(x) - E[f^p(x, \hat{w}_n^p)|\, x^n])^2 \\
&\quad + E\left[(E[f^p(x, \hat{w}_n^p)|\, x^n] - f^p(x, \hat{w}_n^p))^2 |\, x^n\right] \\
&\quad +2\, E\left[(g(x) - E[f^p(x, \hat{w}_n^p)|\, x^n])\,(E[f^p(x, \hat{w}_n^p)|\, x^n] - f^p(x, \hat{w}_n^p))\,|\, x^n\right] \\
&= (g(x) - E[f^p(x, \hat{w}_n^p)|\, x^n])^2 + E\left[(E[f^p(x, \hat{w}_n^p)|\, x^n] - f^p(x, \hat{w}_n^p))^2 |\, x^n\right] \\
&= B(x|x^n;p)^2 + V(x|x^n;p).
\end{aligned} \tag{III.5}$$

The term $B(x|x^n;p) = g(x) - E[f^p(x, \hat{w}_n^p)|\, x^n]$ quantifies the *bias* of the estimator $\hat{w}_n$. $E[f^p(x, \hat{w}_n^p)|\, x^n] - f^p(x, \hat{w}_n^p)$ is the *model variation*. We have

$$B(x|x^n;p)^2 \;=\; (g(x) - E[f^p(x, \hat{w}_n^p)|\, x^n])^2,$$

the *squared bias*, and

$$V(x|x^n;p) \;=\; E\left[(E[f^p(x, \hat{w}_n^p)|\, x^n] \;-\; f^p(x, \hat{w}_n^p))^2|\, x^n\right],$$

the *model variance*. Note that the covariance term vanishes because $B(x|x^n;p)$ is non-stochastic. Hence

$$\begin{aligned} IMSE_N(x^n) &= \int B(x|x^n;p)^2 + V(x|x^n;p)\,\mu(dx). \\ &= ISB(x^n;p) + IMV(x^n;p), \end{aligned} \tag{III.6}$$

where we call ISB the Integrated Squared Bias, and IMV the Integrated Model Variance.

Integrated Squared Bias

In the case where we are able to obtain clean training examples, y^n is completely determined by x^n, as is $\hat{w}_n$. Therefore, for any x^n the training sample becomes $z^n = (x^n, g(x^n))$, where we abuse notation by writing $g(x^n) = (g(x_1), g(x_2), \ldots, g(x_n))'$. Integrating the squared error over the input space gives the Integrated Squared Bias:

$$ISB(x^n;p) \;=\; \int (g(x) \;-\; f^p(x, l_n(x^n, g(x^n);p)))^2 \mu(dx), \tag{III.7}$$

where μ is an appropriately chosen measure on $(\mathbf{X}, \mathcal{B}(\mathbf{X}))$.

III.D.2 Expected Network Error

Given our assumptions, we can relate cross-validation to IMSE. For clarity and notational convenience, we first introduce another measure of generalization closely related to IMSE.

For each weight $w \in \mathbf{W}^p$, we have defined the mean squared error $\lambda(w;p)$ in Equation (III.1). We define Q_N to map each dataset to the mean squared error of the estimated network

$$Q_N(z^N;p) \equiv \lambda(l_N(z^N;p);p).$$

When Assumption 3 holds, we have

$$\begin{aligned} \lambda(w;p) &= E\left[(g(X) - f^p(X,w))^2\right] + \sigma^2 \\ &= E\left[(g(X_{N+1}) - f^p(X_{N+1},w))^2\right] + \sigma^2 \end{aligned}$$

as is easily verified. We therefore have by independence that

$$Q_N(z^N;p) \;=\; E\left[(g(X_{N+1}) - f^p(X_{N+1}, l_N(Z^N;p)))^2|Z^N = z^N\right] + \sigma^2.$$

Thus, by using the law of iterated expectations, we have

$$\begin{aligned}
E\left[Q_N(Z^N;p)\right] &= E\left[E\left[Q_N(Z^N;p)|Z^N\right]\right] \\
&= E\left[E\left[(g(X_{N+1}) - f^p(X_{N+1}, l_N(Z^N;p)))^2|Z^N\right] + \sigma^2\right] \\
&= E\left[(g(X_{N+1}) - f^p(X_{N+1}, l_N(Z^N;p)))^2\right] + \sigma^2 \\
&= E\left[E\left[(g(X_{N+1}) - f^p(X_{N+1}, l_N(Z^N;p)))^2|Z_{N+1}\right]\right] + \sigma^2 \\
&= E[M_N(X_{N+1};p)] + \sigma^2 \\
&= E[M_N(X;p)] + \sigma^2 \\
&= IMSE_N(p) + \sigma^2.
\end{aligned}$$

Likewise, given $x^N \in \mathbf{X}^N$,

$$E[Q_N(Z^N;\, p)|X^N \;=\; x^N] = IMSE_N(x^N;p) + \sigma^2. \tag{III.8}$$

III.D.3 Statistical Risk

Here, we define what is commonly referred to as statistical "risk" or, "regret," in order to demonstrate the relation between the IMSE and this well-known statistical measure of generalization. The reader familiar with functional analysis will recognize it as the metric distance between the network and the target function obtained using the L_2 norm:

$$R_g(f^p(\cdot, w)) = \|f^p(\cdot, w) - g\|_2,$$

where

$$\begin{aligned}
\|f^p(\cdot, w), g\| &\equiv \{\int |f^p(x,w) - g(x)|^2 \mu(dx)\}^{\frac{1}{2}} \\
&= E[|f^p(X,w) - g(X)|^2]^{\frac{1}{2}}.
\end{aligned}$$

For the case that the sampling distribution is i.i.d., it is straightforward to show that the expected squared regret (over all i.i.d. training sets of size n) is equal to the IMSE,

$$E\left[||f^p(\cdot, l_n(Z^n;p)) - g||_2^2\right] = IMSE_N(p).$$

Conversely, the expected squared regret conditioned on a particular set of inputs x^n is equal to the IMSE for that set of inputs,

$$E\left[||f^p(\cdot, l_n(Z^n;p)) - g||_2^2 \mid x^n\right] = IMSE_N(x^n;p).$$

See Appendix C for details. As an aside, we note that with this relationship convergence results for IMSE are readily obtained by incorporating a complexity regularization penalty into the learning criterion and into the definition of the IMSE. The results of Appendix C then allow us to apply convergence rates obtained for the complexity regularized estimator to the IMSE criterion [11, 12, 13]. Note also that Assumptions 1-3 imply

$$R_g(f^p(\cdot, l_n(Z^n;p))) = \lambda(l_n(Z^n;p);p) - \sigma^2.$$

III.E Estimates of Generalization

In practice, we must (typically) estimate the error of network p over novel data ("generalization") from a finite set of examples. We define two commonly used versions of cross-validatory measures, which estimate the generalization performance of a network by evaluating the error over examples not used in training. The first is appropriate when data is abundant; the second is appropriate when data is in short supply.

III.E.1 Hold-Out Cross-Validation

The following version of cross-validation is commonly used for evaluating "generalization" when an abundant supply of novel data is available for use as a "hold-out" set:

$$G_{N,M}(z^N, \tilde{z}^M; p) = \frac{1}{M} \sum_{i=1}^{M} \left(\tilde{y}_i - f^p(\tilde{x}_i, l_N(z^N; p)) \right)^2, \tag{III.9}$$

where $\tilde{z}^M = (z_{N+1}, \ldots, z_{N+M})$.

III.E.2 Delete-One Cross-Validation

Delete-one cross-validation measure maximizes the number of examples used in training while still using every available example to test generalization:

$$C_N(z^N; p) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - f^p(x_i, l_{N-1}(z^N_{-i}; p)) \right)^2 \tag{III.10}$$

where z^N_{-i} denotes the training set obtained by deleting the ith example. Using z^N_{-i} instead of z^N avoids a downward bias due to testing upon examples used in training, as we show below (Theorem 3 in Chapter VI).

Chapter IV

Exemplar Selection for Clean Data

In this chapter we derive a method for selecting exemplars for training a multilayer feedforward network architecture to estimate an unknown deterministic mapping. Therefore, the training data is "clean," i.e., measured either without error or with negligible error. This method incrementally grows the training set as necessary to achieve the desired level of accuracy. Our selection criterion does not depend on using a neural network estimator, thus it may be used for general purpose nonlinear regression using any statistical estimator.

The objective is to minimize the data requirement of learning. In a particular sense, we are performing a kind of data compression, by selecting exemplars representative of the set of all available examples. Towards this end, we choose a criterion for selecting training examples that works well in conjunction with the criterion used for learning, here, least squares. We proceed sequentially, selecting an example that, when added to the previous set of training examples and learned, maximizes the decrement of network squared error over the input space. When dealing with clean data and deterministic relationships, we desire concise training sets that minimize the *Integrated Squared Bias* (ISB). We use the ISB to derive a selection criterion for evaluating individual training examples, the ΔISB, that we maximize to select new exemplars.

We conclude with graphical illustrations of the method, and demonstrate its use during network training. Several benefits are apparent for practical use in a variety of applications. Experimental results indicate that training upon exemplars selected in this fashion can save computation in general purpose use as well.

We demonstrate that our method selects concise training sets, and that it can be used to reduce the cost of network training. The method works well on networks with multiple inputs and outputs. Theoretically, it should work on problems of any dimension. However, experience in high dimensional domains is required to determine whether the benefits of our technique outweigh the overhead of the selection process. We have confirmed this conjecture on small problems, showing a reduction in the number of floating point operations required to train a network on several learning tasks. In fact, we found that once a concise set of exemplars is found, the cost of retraining another network to fit the entire set of candidates can be done at a fraction of the cost of training over all the examples. We can also grow the network while simultaneously growing the training set, again resulting in a reduction of floating point operations on the problems we tried.

IV.A Derivation of the Method

The approach developed here applies to the noiseless learning task exclusively. We choose a criterion for evaluating sets of training examples appropriate for use with the squared error performance criterion. It is intractable to select a set of exemplars that minimize this criterion directly, but it is feasible to maximize its decrement while growing the training set. Therefore, we sequentially add exemplars to the training set, training the network in between selection of successive exemplars. We derive a criterion for evaluating individual examples that allows us to select exemplars maximizing the decrement in squared error that would result from adding the new exemplar to the training set and training upon the resulting training set.

For a sample of size n, the optimal training set satisfies $x^n_* = \arg\min_{x^n} ISB(x^n)$. Finding such a training set is computationally impractical. Here we pursue a more modest goal. Given a training set x^n, we shall seek x_{n+1} that maximizes the decrement in ISB, $ISB(x^n) - ISB(x^{n+1}), x^{n+1} = (x^n, x_{n+1})$. Note that this property for x_{n+1} is a necessary condition for solving the computationally impractical global problem, but will not necessarily deliver the globally optimal solution. Nevertheless, this approach permits a computationally feasible and attractive method for sequential selection of exemplars.

Let $\check{x}_{n+1}$ be a candidate input, and $\check{x}^{n+1} = \{x^n, \check{x}_{n+1}\}$. Let $L_n(x^n) = l_n(x^n, g(x^n))$, (where $g(x^n)$ is as defined in the Chapter III), so that $w_n = L_n(x^n)$, and put $\check{w}_{n+1} = L_{n+1}(\check{x}^{n+1})$. Thus we seek to maximize

$$\begin{aligned} ISB(x^n) - ISB(\check{x}^{n+1}) &= \beta^2(L_n(x^n)) - \beta^2(L_{n+1}(\check{x}^{n+1})) \\ &= \beta^2(w_n) - \beta^2(\check{w}_{n+1}), \end{aligned} \tag{IV.1}$$

where $\beta^2(w) = \int [g(x) - f(x,w)]^2 \mu(dx)$. (The variable L is introduced to make explicit the role of x^n in the noiseless environment; the variable β^2 is introduced to allow a change of argument emphasizing the role of the network weights.) We could pick x_{n+1} to maximize expression (IV.1) directly, albeit at great expense. Fortunately, a much simpler procedure can be obtained by exploiting certain properties of the learning algorithm.

To proceed, we take a first order Taylor expansion of $\beta^2(\check{w}_{n+1})$ around $\beta^2(w_n)$, yielding

$$\beta^2(w_n) - \beta^2(\check{w}_{n+1}) \approx (w_n - \check{w}_{n+1})' \nabla_w \int (g(x) - f(x, w_n))^2 \mu(dx).$$

By Bartle, (Corollary 5.9, [15]) the gradient and integral can be interchanged to yield

$$\nabla_w \int (g(x) - f(x,w_n))^2 \mu(dx) = -2 \int \nabla_w f(x, w_n)\, (g(x) - f(x, w_n)) \mu(dx), \tag{IV.2}$$

provided in addition to the compactness of W that $\partial f / \partial w$ exists on $X \times W$, and that $f(x, \cdot)$ and the elements of $\nabla_w f(x, \cdot)$ are dominated by square integrable functions of x only. We have already assumed implicitly that g is square integrable. Consequently,

$$\beta^2(w_n) - \beta^2(\check{w}_{n+1}) \approx -2(w_n - \check{w}_{n+1})' \int \nabla_w f(x, w_n)(g(x) - f(x, w_n)) \mu(dx). \tag{IV.3}$$

Equation (IV.3) provides a computationally simpler estimate of equation (IV.1). For each successive selection of an example from a set of candidates, the integrand of equation (IV.3) need be computed once, and this can be approximated by summing over the set of candidate examples. Then, to evaluate each individual example in the set of candidates, we compute $(w_n - \check{w}_{n+1})$, and take its inner product with the vector quantity obtained from the summation over the set of candidates.

To calculate (IV.3), we need to estimate $(w_n - \check{w}_{n+1})$, given $\check{x}_{n+1}$. When n is large, it is generally true that $\check{w}_{n+1}$ is in a neighborhood of w_n. The particular weight update we assume satisfies this requirement. For example, as described in [96], we may use the Gauss-Newton weight update:

$$w_n - \check{w}_{n+1} \approx$$

$$-\left[\frac{1}{n+1}H(\check{x}^{n+1}, w_n)\right]^{-1} \frac{1}{n+1}\nabla_w f(\check{x}^{n+1}, w_n)(g(\check{x}^{n+1}) - f(\check{x}^{n+1}, w_n)). \qquad \text{(IV.4)}$$

Here,

- $f(\check{x}^{n+1}, w_n) = (f(x_1, w_n), f(x_2, w_n), \ldots, f(x_n, w_n), f(\check{x}_{n+1}, w_n))'$.
- $\nabla_w f(\check{x}^{n+1}, w_n) = (\nabla_w f(x_1, w_n), \ldots, \nabla_w f(x_n, w_n), \nabla_w f(\check{x}_{n+1}, w_n))$, a matrix with p rows and $n+1$ columns.
- $H(\check{x}^{n+1}, w_n)^{-1} = (\nabla_w f(\check{x}^{n+1}, w_n)\nabla_w f(\check{x}^{n+1}, w_n)')^{-1}$ acts as an approximation to the so-called Hessian matrix of second derivative information [37, 89, 96].

As an aside, note that the Gauss-Newton weight update we use here may be replaced. For example, we might use a steepest descent update, which would save computation at the cost of reduced accuracy. On the other hand, we could use a Newton weight update, which would provide higher accuracy but would require the Hessian matrix of second derivatives.

The estimate of $w_n - \check{w}_{n+1}$, equation (IV.4), can be simplified by noting that for $n = 1, 2, \ldots$, $\frac{1}{n}\sum_{i=1}^{n} \nabla_w f(x_i, w_n)(g(x_i) - f(x_i, w_n)) = 0$. This is by definition of w_n, which is optimal for x^n. (Note that this remains a valid assumption when w_n is approximated by a locally optimal estimate $\tilde{w}_n$ obtained by gradient descent.) As a result,

$$\nabla_w f(\check{x}^{n+1}, w_n)(g(\check{x}^{n+1}) - f(\check{x}^{n+1}, w_n)) = \nabla_w f(\check{x}_{n+1}, w_n)(g(\check{x}_{n+1}) - f(\check{x}_{n+1}, w_n)).$$

We also employ the approximation

$$\left[\frac{1}{n}H(x^n, w_n)\right]^{-1} \approx \left[\frac{1}{n+1}H(\check{x}^{n+1}, \check{w}_{n+1})\right]^{-1}.$$

Collecting the above approximations together, define our estimate of $w_n - \check{w}_{n+1}$ to be

$$\Delta\check{w}_{n+1} \equiv -\left[\frac{1}{n}H(x^n, w_n)\right]^{-1} \frac{1}{n+1}\nabla_w f(\check{x}_{n+1}, w_n)\,(g(\check{x}_{n+1}) - f(\check{x}_{n+1}, w_n)), \qquad \text{(IV.5)}$$

giving as one implementation of (IV.3),

$$\beta^2(w_n) - \beta^2(\check{w}_{n+1}) \approx 2\Delta\check{w}_{n+1}' \int \nabla_w f(x, w_n)(g(x) - f(x, w_n))\mu(dx). \qquad \text{(IV.6)}$$

For use in comparing candidate examples with each other, note that the constant factors of equation (IV.5) may be eliminated, giving

$$\Delta \breve{w}_{n+1} = [H(x^n, w_n)]^{-1} \nabla_w f(\breve{x}_{n+1}, w_n) \, (g(\breve{x}_{n+1}) \; - \; f(\breve{x}_{n+1}, w_n)). \qquad \text{(IV.7)}$$

Ignoring the constant factor in (IV.6) and combining with (IV.7) leads to the criterion that we propose for selecting the next training example,

$$\Delta ISB(\breve{x}_{n+1}|x^n) = (g(\breve{x}_{n+1}) - f(\breve{x}_{n+1}, w_n))' \nabla_w f(\breve{x}_{n+1}, w_n)' [H(x^n, w_n)]^{-1} \int \nabla_w f(x, w_n)(g(x) - f(x, w_n)) \mu(dx). \qquad \text{(IV.8)}$$

In practice the distribution μ is unknown, but can be well approximated by the empirical distribution of the candidate examples. This means that the integral appearing in (IV.8) is approximated by an average over all the candidate examples. This average need be computed only once. The training example x_{n+1} is selected so that

$$x_{n+1} \in \arg \max_{\breve{x}_{n+1}} \Delta ISB(\breve{x}_{n+1}|x^n). \qquad \text{(IV.9)}$$

Although we have arrived at this criterion by making use of approximations valid for large n, this criterion has an appealing interpretation apart from large sample considerations. Specifically, ΔISB can be interpreted directly as the square of the weighted Euclidean distance between the gradient of squared error with respect to network weights integrated over all candidate examples, and the gradient of squared error with respect to network weights for a single candidate example. (The matrix H^{-1} performs the weighting.) Thus, picking x_{n+1} to maximize ΔISB can be viewed as picking the single example having individual error gradient most highly correlated with error gradient of the entire set of examples. Learning with this example is therefore likely to be especially informative, relative to other available examples. The ΔISB criterion thus possesses heuristic appeal in training sets of any size. As n becomes large, the derivation above establishes that ΔISB approaches $ISB(x^n) - ISB(\breve{x}_{n+1})$, providing further support for its use.

IV.B Implementation

We now consider the practical use of the ΔISB criterion (IV.8) for evaluating examples for training a one-hidden layer feedforward network. We have available a set of N examples from which to select training examples. Call this set x^N, the set of candidate examples. Suppose we have trained the network using a subset of n examples x^n. Given x^n, we evaluate $\Delta ISB(\breve{x}_{n+1}|x^n)$ for all $\breve{x}_{n+1} \in x^N$. For simplicity, we begin by putting $n = 1$ and find x_1 such that $x^1 = \{x_1\} = \arg\min_{x^1} ISB(x^1)$. This is computationally appealing, since fitting a single example by a neural network can be accomplished efficiently. Given a priori knowledge about the class of functions that we can expect the network learning rule to utilize to fit a single example, it is possible to calculate the optimal location of x_1 by employing summary statistics over the set of available examples. A single example is fit perfectly with a constant function (see Figures IV.C.2a and IV.6a.) It is then straightforward to choose the single example minimizing $ISB(x^1)$. Calculate $m = \frac{1}{N}\sum_{i=1}^{N} g(x_i)$, $x_i \in x^N$, and select $x_1 = \arg\min_{x \in x^N}(g(x) - m)^2$.

Given x^1, we proceed to grow the training set by successive optimization of ΔISB. For $n = 1, 2, \ldots$, we then choose

$$x_{n+1} = \arg\max_x \Delta ISB(x|x^n). \qquad \text{(IV.10)}$$

In growing the training set, a number of practical matters must be addressed. Because we are trying to learn a deterministic mapping, it is appropriate to specify a tolerance $\mathcal{E}_N^{\max}$ representing the desired accuracy desired for the fit over all of the candidates as measured by

$$\mathcal{E}_N = \left[\frac{1}{N}\sum_{x\in x^N}(g(x) - f(x, w_n))^2\right]^{\frac{1}{2}}.$$

We desire $\mathcal{E}_N \leq \mathcal{E}_N^{\max}$. Clearly, before selecting a new exemplar, we should always check whether $\mathcal{E}_N \leq \mathcal{E}_N^{\max}$, and quit if it does. We refer to a set of exemplars that give this condition as "sufficient."

If this condition does not hold, we must determine when the current exemplars have been fit well enough to select a new one, as the underlying theory and practical experience suggest that, for a given network complexity, the selection criterion will work best if the fit is optimal for the current set of exemplars. Optimality for the given network complexity is ensured either by the finding that $\mathcal{E}_n$ is sufficiently small, where

$$\mathcal{E}_n = \left[\frac{1}{n}\sum_{x\in x^n}(g(x) - f(x, w_n))^2\right]^{\frac{1}{2}},$$

or by finding that repeated optimization attempts do not give further improvement in $\mathcal{E}_n$. In implementing the requirement that $\mathcal{E}_n$ be sufficiently small, we specify that $\mathcal{E}_n \leq \mathcal{E}_N^{\max}$ when selecting x_{n+1}. This requirement follows from the observation that if the given model cannot adequately fit the training points, then it is unlikely it will adequately fit the entire set of candidates, since $x^n \subset x^N$.

Our experience indicates that fitting x^n to an even closer tolerance can result in a more concise set of exemplars, i.e., a sufficient set of smaller size. When the minimal number of exemplars is desired, $\mathcal{E}_n$ should be stringently minimized.

If a somewhat larger than minimal set of exemplars is acceptable, then it may not be necessary to obtain the globally minimal value for $\mathcal{E}_n$ before selecting a new exemplar. The task of seeking the most concise training set can require more total overhead (which includes training as well as selection cost) than for another sufficient but larger set. Relaxing the tolerance on $\mathcal{E}_n$ allows a new exemplar to be selected sooner. This reduces the cost of training over the exemplars, and can thus reduce total overhead as well even though more exemplars may need to be selected to comprise a sufficient set. We regulate this by using a variable $\mathcal{E}_n^{\max}$ to determine when $\mathcal{E}_n$ is suffiently small to select a new exemplar.

A problem that can arise during this process is that an exemplar may be selected that is close to a previous choice, in the sense that both input and target values are close together. This can indicate that optimization in the prior exemplars was not complete (i.e., $\mathcal{E}_n$ can be reduced further,) but also is observed to occur when network complexity is insufficient to achieve $\mathcal{E}_N \leq \mathcal{E}_N^{\max}$. Such exemplars too close to a previous selection are easy to detect, and in such cases we simply discard the new exemplar and continue training on the current set. To avoid the cost of repetitively selecting a new exemplar only to reject it, we modify $\mathcal{E}_n^{\max}$ adaptively during training by reducing it when a selection is rejected. This also makes the method less sensitive to initial setting of $\mathcal{E}_n^{\max}$, in that we may set it to a larger value initially, say, $\mathcal{E}_n^{\max} = \mathcal{E}_N^{\max}$, and then adaptively reduce it during training as necessary. Practical implementation of exemplar rejection is based on selecting a small number d. In our experiments the input and output were scaled equivalently, so we let $d = O(\mathcal{E}_N^{\max})$. If the input vector of a new exemplar is within Euclidean distance d of a previous exemplar in input space, we compute the distance between their output values, and if their output values are closer than $\mathcal{E}_N^{\max}$, we reject the new exemplar. This permits us to keep exemplars in regions of the input space where the output varies rapidly with the input.

If the learning algorithm reaches a local minimum with $\mathcal{E}_N^{\max} < \mathcal{E}_n$, then either the learning rule needs to continue its search within the current weight space in order to obtain a better optimum, or network complexity is insufficient to fit the current set of exemplars. Therefore, at this point we either restart network training or modify network complexity and continue.

If the learning algorithm reaches a local minimum with $\mathcal{E}_n^{\max} < \mathcal{E}_n \leq \mathcal{E}_N^{\max} < \mathcal{E}_N$, we attempt to add a new exemplar, even though $\mathcal{E}_n \leq \mathcal{E}_n^{\max}$ has not been met, since we have found that the method will often select an informative exemplar anyway (i.e., one not too close to previous choices.) If the new exemplar is rejected, we either restart network training or modify network complexity and continue.

It may be difficult or expensive to determine whether a given network can fit a set of exemplars within tolerance. Although we may not know in general when an estimate is sufficiently close to a globally optimal solution, in particular circumstances we may employ capacity results for neural networks to determine whether this is the case (cf., Vapnik-Chervonenkis dimension [2, 10]). For example, assume that each unit employs the usual sigmoid squashing function. If h is the number of hidden units in the neural network, then, given any set of $n \leq h$ examples, there exist weights w_n such that $\sum_{i=1}^{n}(g(x_i) - f(x_i, w_n))^2 = 0$. (Theorem 2.5, [48].) Thus, in this case we know in advance that a set of weights exists that can fit the training examples exactly.

To summarize, we now present an algorithm for implementing the ΔISB criterion that embodies each of the considerations addressed above. The general algorithm begins with a network of arbitrary complexity, and increases the complexity appropriately as the exemplars become too difficult for the network to fit. In our experiments we always begin with a network consisting of a single hidden unit. We use multistarts to determine when the network is having difficulty fitting the data, by restarting the network a number of times on the data before increasing network complexity. Let R count the number of restarts, and let $R^{\max}$ be the maximum number of restarts we allow before increasing network complexity. Note that we reset R after adding a new exemplar, since each new exemplar results in a new training set, $x^n, n = 1, 2, \ldots$.

1. Begin by selecting x_1 minimizing $ISB(x^1)$, and set $R = 0$.
2. Continue training until one of the following occurs and take the actions listed, in sequence:
   (a) $\mathcal{E}_n \leq \mathcal{E}_n^{\max}$
   - If $\mathcal{E}_N \leq \mathcal{E}_N^{\max}$, quit.
   - Select $x_{n+1} = \arg\max_x \Delta ISB(x|x^n)$.
   - Check whether x_{n+1} is too close to previous exemplars. If x_{n+1} is rejected, reduce $\mathcal{E}_n^{\max}$ and continue training.
   - Otherwise, append x_{n+1} to x^n, set $R = 0$, and continue training.

   (b) $\nabla_w f(x^n, w_n)\,(g(x^n) - f(x^n, w_n)) = 0$ and $\mathcal{E}_n \leq \mathcal{E}_N^{\max}$:
   - If $\mathcal{E}_N \leq \mathcal{E}_N^{\max}$, quit.
   - Otherwise, select $x_{n+1} = \arg\max_x \Delta ISB(x|x^n)$.
   - Check whether x_{n+1} is too close to previous exemplars. If x_{n+1} is not rejected, append it to x^n, set $R = 0$, and continue training.
   - Otherwise, if $R < R^{\max}$, restart the network, increment R, and continue training.
   - Otherwise, increase network complexity, set $R = 0$, and continue training.

   (c) $\nabla_w f(x^n, w_n)\,(g(x^n) - f(x^n, w_n)) = 0$ and $\mathcal{E}_n > \mathcal{E}_N^{\max}$:
   - If $R < R^{\max}$, restart the network, increment R, and continue training.
   - Otherwise, increase network complexity, set $R = 0$, and continue training.

This algorithm could be modified to utilize more sophisticated ways of regulating network complexity. It is useful in its present form for many practical situations, and is sufficient for demonstrating the technique.

IV.C Demonstrating the Technique

IV.C.1 Illustrating the behavior of ΔISB

Here we wish to convey the qualitative behavior of the ΔISB criterion. We illustrate this using the network illustrated in Figure IV.1 to obtain the target function $g(x)$ illustrated in Figure IV.2a.

This is a one dimensional map with upper and lower plateaus at about 0.2 and 0.8, respectively, and a steep dropoff. The dropoff is shifted left of center, so that the lower plateau is more extensive than the upper plateau. This diagram illustrates a feedforward network with 1 input, 1 output, and 1 hidden unit. Each unit employs for its transfer function the usual sigmoided output activation $o(a) = (1 + exp(-a))^{-1}$, where a is the weighted sum of the inputs to the unit. The arcs are labelled by real values, denoting the weights. The values of the weights in this figure give a set of weights for which $f(x, w) = g(x)$. This set of weights is $w^* = (-20.0, 70.0, 1.5, -3.0)$. Note the size of weights w_1 and w_2 relative to w_3 and w_4. This is a manifestation of the region in g of high first derivative,

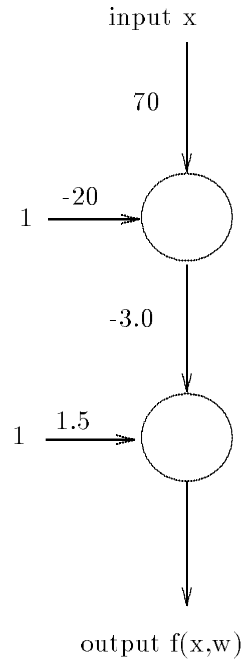


Figure IV.1: A simple one hidden layer neural network.

which the network function fits by employing large magnitude weights on the hidden unit. In what follows, we will illustrate the ΔISB for three different network functions obtained by perturbing the weights of this network slightly. For clarity of illustration, we will assume all inputs are equally likely, so that μ is the uniform distribution.

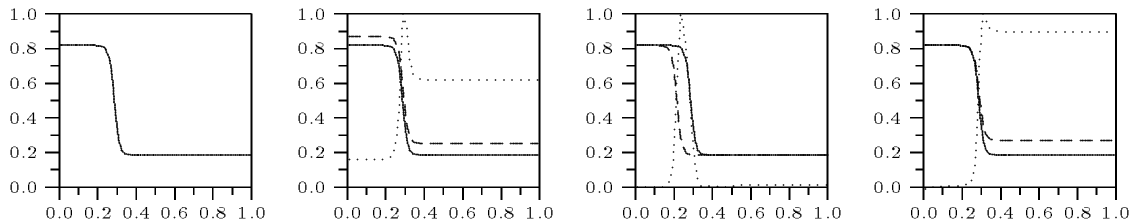
In Figures IV.2b-d, we modify one weight at at time to obtain network functions slightly perturbed from $g(x)$. In each figure, the network function so obtained is given by the dashed line. This has the incidental benefit of illustrating the role each weight plays in the network mapping. For example, in Figure IV.2b, weight w_3 was perturbed, by putting $w_3 = 1.9$. This results in an upward shifting of the entire mapping. The dotted line gives the plot of ΔISB, normalized to allow it to be plotted along with $f(\cdot, w)$ and g. Plotting f and $g(x)$ together allows a comparison of the two functions, and provides a visualization as to how the ΔISB criterion varies as the difference between f and g. Figure IV.2b corroborates the intuition that the sampling criterion indicates the potential of a candidate example for reducing squared error over the input space. Although both plateaus are shifted upwards an equal distance, a training example chosen from the region of input space corresponding to the lower plateau is weighted more heavily than one chosen from the upper plateau. Presumably, that is because an example from the lower plateau might result in bringing the lower plateau of f back in line with g, and this would be better in comparison with choosing an example from the upper plateau, which covers a smaller region of the input space. To see where the next exemplar would be selected by (IV.9), observe on the coordinate axis the location of the maximum value of the Delta ISB curve.

Figures IV.2c and IV.2d illustrate that the criterion gives high value to candidate examples lying in regions where more information is necessary. The criterion displays an intuitively pleasing behavior in these figures, but it is not obvious that the criterion will always act in accordance to our immediate intuition in more general settings. For example, the criterion may place high value on examples in regions where network error is substantially lower than the network error over other regions. This can occur by the effect of μ, which essentially weights each example according to the environmental probability law. It may also occur due to the form of the basis functions. This is because small changes in the network function in one region resulting from the attempt to fit a particular example can have a global impact upon the network function far away from the new exemplar. This is illustrated in the next sections, where we next apply the ΔISB criterion in a sequential manner during network training.

IV.C.2 Training a fixed size network on actively selected exemplars

We investigate what sequence of examples will be chosen by the criterion while learning the mapping of Figure IV.2a. Examples will be chosen from a set of candidates, x^N, generated by the network of Figure IV.1. Here, $N = 200$. Between selection of successive examples, the network function is optimized over the training set, x^n. Subsequent to optimization over x^n, obtaining w_n, $\Delta ISB(x|x^n)$ is calculated for each $x \in x^N$, and x_{n+1} chosen from $\{x \in \max_x \Delta ISB(x|x^n)\}$. The training set is updated, obtaining $x^{n+1} = \{x^n, x_{n+1}\}$. We commence with optimizing $f(x, w)$ over x^{n+1} to obtain w_{n+1}, and so forth.

Discussion of some implementation details is in order. We use the conjugate gradient algorithm to set the network weights, in order to make the computational comparisons more fair. We discuss this in more detail below in Section IV.C.3. As our selection criterion, we use the implementation of equation (IV.9) given by equation (IV.8). In the interest of



(a.) Desired function (b.) **w** = (-20,70,**1.9**,-3) (c.) **w** = (**-15**,70,1.5,-3) (d.) **w** = (-20,70,1.5,**-2.5**)

Figure IV.2: Figure (a.) shows the desired function, $g(x)$, here a scaled and translated sigmoid generated by the one hidden unit network with weights $w = (-20, 70, 1.5, -3.0)$. The next three figures illustrate the ΔISB upon networks obtained by perturbing these weights. The perturbed weight vector is given at the bottom of each figure, with the perturbed weight in bold. The network function $f(x, w)$ is given by the dashed line, $g(x)$ is given by the solid line, and the ΔISB, normalized for plotting purposes, is given by the dotted line.

computational simplicity, we modified the equation for the purpose of demonstrating the technique. The particular gradient descent algorithm we used (conjugate gradient) does not calculate $H(x^n, w_n)$, so we used an approximation to it obtained by calculating only the diagonal elements of $H(x^n, w_n)$. This has the additional benefit of resulting in a matrix that is guaranteed to be positive definite, and inexpensive to invert. When the number of current exemplars was less than the number of parameters in the network, we set $H(x^n, w_n)$ to the identity matrix to avoid the situation where elements of the diagonal matrix become large due to ill-conditioning. Although we recommend using $H(x^n, w_n)$ whenever possible, we note that the performance of the ΔISB criterion did not seem overly sensitive to the approximations we made here.

We followed the algorithm given above in Section IV.B. We began with a training set consisting of a single example. In order to select x_1, we calculated the average value of the outputs over the candidate examples, as discussed above, and selected as the first exemplar the candidate x with $g(x)$ closest to this value. See Figure IV.C.2a. The initial training set is indicated in the figure by a marker superimposed upon plot of the desired function, $g(x)$. We initialized the network by randomizing the weights to values in $[-0.3, 0.3]$, and trained the network over x^1, using a value of $\mathcal{E}_N^{\max} = 0.01$. The resulting network $f(\cdot, w_1)$, gives a constant function passing through x_1. After obtaining w_1 on x^1, we computed $\Delta ISB(x|x^1)$ for $x \in x^N$. ΔISB is normalized by its maximum absolute value, so that it may be plotted along with f and g. Note that $\Delta ISB(x|x^1)$ may take on negative values.

The next exemplar selected by (IV.9) is located by the maximum value of the ΔISB curve. This exemplar is appended to the initial training set x^1 to give the second training set, x^2. Figure IV.C.2b shows the result of training on x^2, the two examples in x^2 being indicated, as before, by markers superimposed upon $g(x)$. Note how training on x^2 resulted in improvement along the entire lower plateau. After obtaining w_2, ΔISB

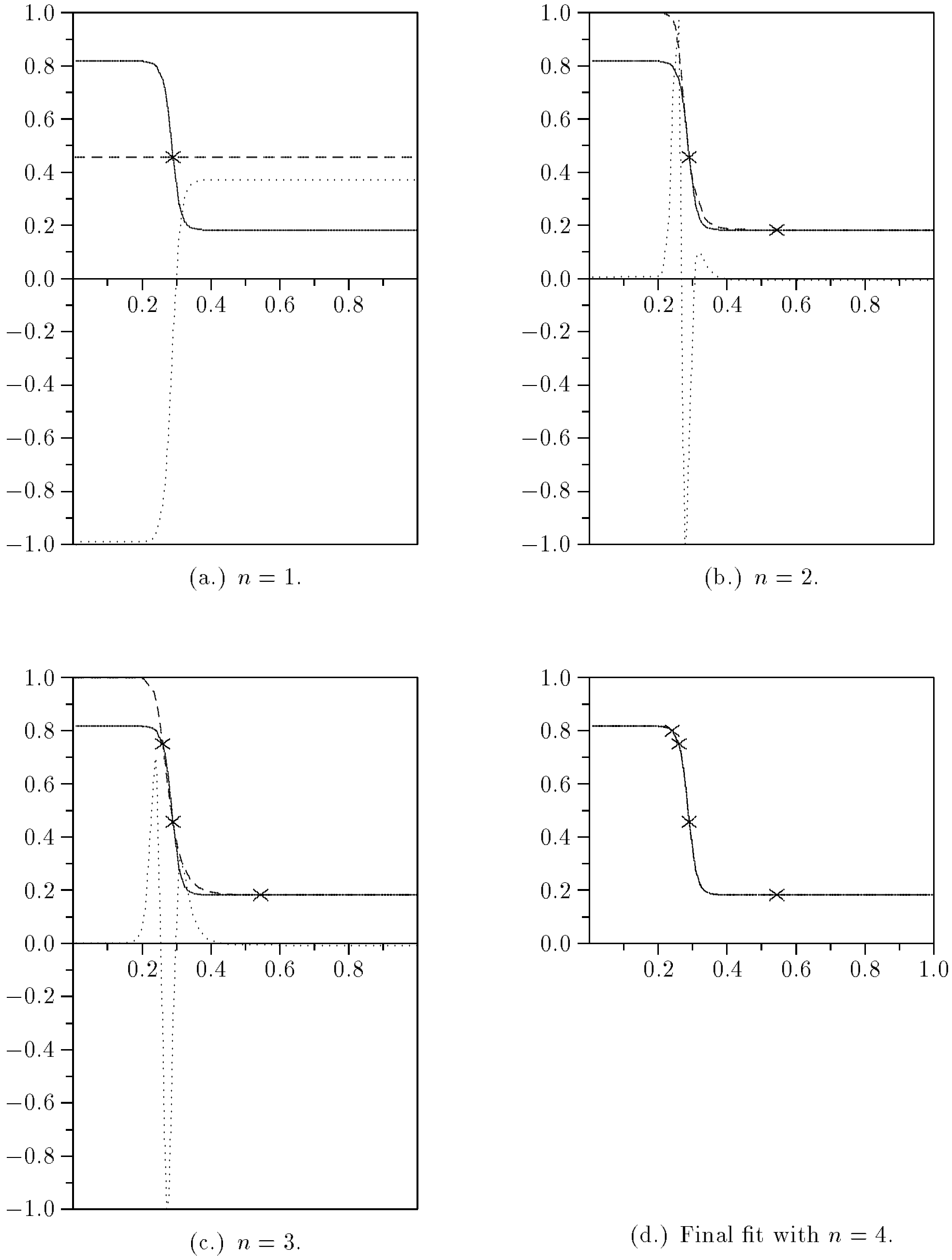


Figure IV.3: These figures illustrate the sequential selection of exemplars during learning. The solid line gives the target $g(x)$. The dashed line gives the network function. The dotted line gives $\Delta ISB(x|x^n)$ for $n = 1, 2, 3$. The exemplars are given by the cross hatch marks on the solid line. For each n, the new exemplar x_{n+1} is given by the lowest valued x maximizing $\Delta ISB(x|x^n)$.

was maximized at a point x_3 near the top of the steep portion of g. Note that this point is not one for which network error was maximal. Hence, ΔISB need not coincide with the heuristic of training on examples according to maximal error. We expect this to be the case for the squared error learning rule we have assumed. After training on x^3, the network function shows only a slight improvement, but this is enough to allow the selection of the final exemplar which provides enough information to complete the learning task. The final exemplar was also selected on the steep portion, near, but not quite on the upper plateau. However, due to the sigmoidal basis functions used in the network, the information provided by x_3 and x_4 is sufficient for the network to learn the upper plateau, even though no exemplars are located in that region. The result is a network function $f(\cdot, w_4)$ within $\mathcal{E}_{run}^{\max}$ of g.

An alternative selection method can be obtained by selecting exemplars uniformly distributed in the inputs space. We chose a sequence of training sets in this fashion and found that this particular learning task required 16 evenly spaced exemplars in order to fit the function within the desired tolerance. The set of exemplars selected in this fashion is displayed along with the resulting trained network function in Figure IV.4. Note the sparsity of exemplars on the steep portion of the curve, and the relative density of exemplars along the plateaus.

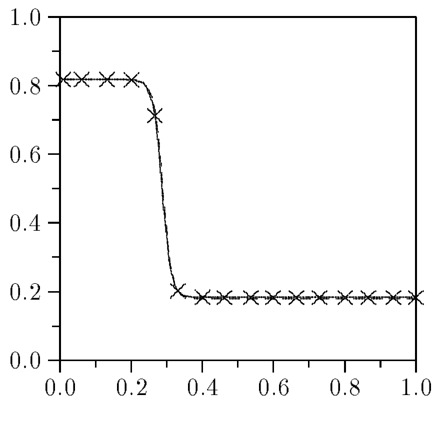


Figure IV.4: Selecting uniformly spaced exemplars required 16 exemplars to fit the sigmoidal function.

IV.C.3 Computational cost comparison

Here we illustrate how active selection may save computation in general purpose use. We compare it with the baseline approach of training over all available examples. Before reporting the results, we first discuss some relevant implementation details.

We measured computational cost in terms of floating point operations. We counted every multiply, divide, and add used during every phase of network training. It suffices to report only on the multiplies here, since the number of adds and divides have been proportional to the multiplies in all of our experiments.

We compare the computational cost of the algorithm in Section IV.B with the cost of using all of the candidate examples. The question is whether or not the overhead due to exemplar selection will be compensated for by a savings due to a lower cost of processing a concise training set. We also keep track of the cost of selecting exemplars, in order to determine the percentage of the total cost that is required in overhead. This allows us to analyze the whether or not it is important to try to reduce the overhead of the selection process. However, the bottom line is whether or not the total cost of fitting the given set of candidates may be reduced by active selection.

The ΔISB method requires a close fit over the exemplars. This may be obtained using any kind of gradient descent, such as a stochastic approximation method (e.g., back-propagation), or steepest descent. We tried using a steepest descent algorithm in previous experiments, and found that several restarts were typically necessary for the learning rule to converge to weights near the optimal. This made the experiments costly to run, and comparisons difficult. Using steepest descent upon actively selected exemplars sometimes required restarts, but the cost of each restart was much less, due to the smaller training set size. Our intuitive judgement is that the ΔISB method would be far cheaper in this case, as we expect our technique to become more cost effective as the number of restarts increases. In order to make more reliable comparisons and to obtain the best baseline we chose to use conjugate gradient, which requires far fewer restarts to find a good set of weights. This largely avoided the case in which a large number of restarts introduced an experimental bias favoring our technique. While we do not do so here, it would be interesting to explore whether active selection could make steepest descent competitive with more sophisticated techniques by reducing the cost of restarts.

Now we present the results of comparing the cost of active selection with the baseline on the learning task illustrated in Figures IV.C.2a-d. In this case, the network is static, needing only one hidden unit. We used $d = \mathcal{E}_N^{\max} = 0.01$, and $\mathcal{E}_n^{\max} = \frac{1}{2}\mathcal{E}_N^{\max}$. We experimented with candidate sets of size 200 and 500, to illustrate how the number of available examples affects the cost benefit. For each size candidate set we compared the total cost of training using active selection with the total cost of training on all of the candidates. This gives four cases. We evaluated five runs for each case. To ensure each training run started from the same initial conditions, we generated five sets of random initial weights, using the same set of five initial weights to initialize the training runs in each of the four cases.

As indicated by the results of Tables IV.1, and IV.2, the method selected sets of 4 exemplars on all but one of the runs, indicating the consistent performance of the method. The average overhead of the selection method comprised 58.8% of the total cost on the runs using 200 candidates, and 43.5% on the runs using 500 candidates. Observe that if the overhead due to exemplar selection is subtracted from the total cost, we see that the cost due to training is significantly less than the cost of training over all available examples, corroborating our intuition that concise training sets can reduce computation. The important question, however, is whether or not training by active selection will reduce computation when the overhead is included in the total cost. As indicated on the results, this was indeed the case for this experiment. The median cost of the ΔISB method required 17.9% of the median baseline cost on 200 candidates, and only 10.4% of the baseline on 500 candidates. Note that when N is increased from 200 to 500, the median cost of active selection grew less than the median baseline cost, which more than doubled.

Shifted sigmoid, N = 200.				
	Baseline	Active selection		
Run	cost	cost	overhead	n
1.	2.94	0.47 (16.0%)	46.0%	4
2.	2.63	0.58 (22.1%)	35.6%	4
3.	2.02	0.58 (28.7%)	35.6%	4
4.	9.64	0.46 (4.8%)	46.8%	4
5.	1.85	0.45 (24.3%)	57.5%	5
Med:	**2.63**	**0.47 (17.9%)**	**46.0%**	

Table IV.1: Comparison of the cost of learning the shifted sigmoid function using 200 candidate training examples. The "Baseline" column gives the cost of training over all 200 examples. The next set of columns pertain to active exemplar selection. The total cost of training is given along with the ratio of this cost with the baseline cost in parentheses. The "overhead" gives the portion of the total cost due to exemplar selection. The number of exemplars selected is given by n. Cost is measured in millions of multiplies. The bottom row gives the median values.

Shifted sigmoid, N = 500.				
	Baseline	Active selection		
Run	cost	cost	overhead	n
1.	6.67	0.82 (12.3%)	58.5%	4
2.	8.80	0.89 (10.1%)	53.4%	4
3.	7.34	0.76 (10.4%)	64.4%	4
4.	15.65	0.95 (6.1%)	63.6%	4
5.	6.92	0.95 (13.7%)	55.0%	4
Med:	**7.34**	**0.76 (10.4%)**	**63.6%**	

Table IV.2: Comparison of the cost of learning the shifted sigmoid function using 500 candidate training examples. The "Baseline" column gives the cost of training over all 500 examples. The next set of columns pertain to active exemplar selection. The total cost of training is given next. The ratio of this cost to the baseline cost is given in parentheses. The "overhead" is the portion of the total cost due to exemplar selection. The number of exemplars selected is given by n. Cost is measured in measured in millions of multiplies. The bottom row gives the median values.

IV.C.4 Active selection and complexity regularization

We now illustrate how the sampling criterion behaves upon a network which does not necessarily have the correct number of hidden units to fit the desired function exactly. We generated a mapping using the 3-hidden unit feedforward network with the connectivity and weight values shown in Figure IV.5, (which we refer to as Network 2) using the same type of sigmoided units used in Network 1.

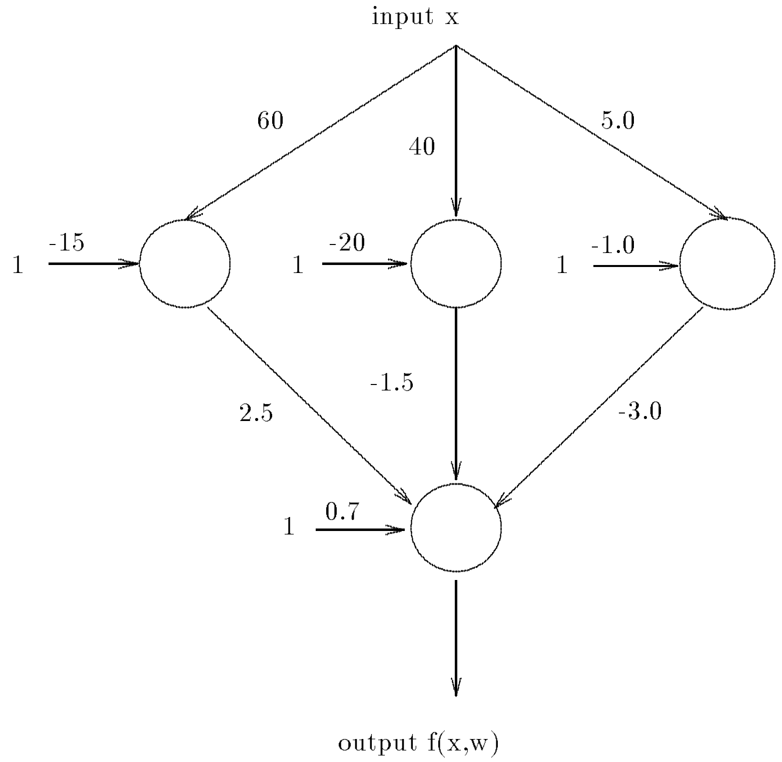


Figure IV.5: Network 2.

Network 2 generates the skewed unimodal mapping $g(x)$ illustrated in Figure IV.6a. This choice of $g(x)$ gives a function difficult for a network of this particular architecture to fit with less than three hidden units. For the first demonstration we used Network 2 to generate $N = 200$ candidate examples uniformly at random over the input space.

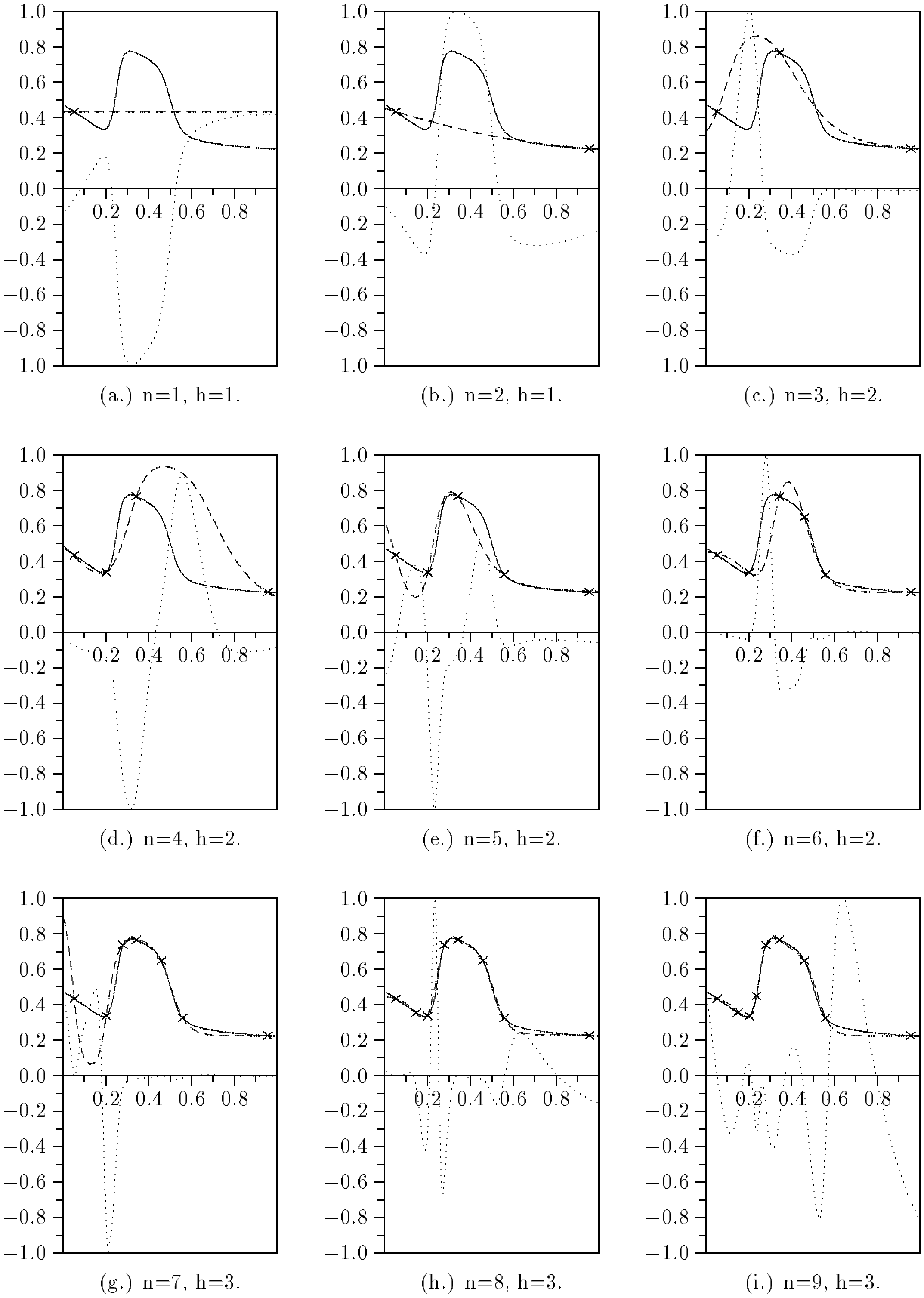


Figure IV.6: The solid line shows the desired function, g, here a skewed hump. The exemplars are given by markers superimposed upon g. For each n, The network function $f(\cdot, w_n)$ is given by the dashed line. For each n, x_{n+1} is given by the smallest x maximizing ΔISB, which is given by the dotted line.

The training regime is the same as for the experiment reported in the previous section, except that here we are allowed to grow the network. The network initially has one hidden unit. The goal is to fit the candidate examples to within a tolerance of $\mathcal{E}_N = 0.01$. We used $\mathcal{E}_n = 0.001$. As training progresses, the algorithm automatically modifies network complexity by adding another hidden unit when the network can no longer fit the exemplars. When the network reached a local minimum without attaining $\mathcal{E}_n \leq \mathcal{E}_N^{\max}$, we restarted the network, randomizing the weights to small values. A maximum of 5 restarts were allowed before growing the network. Examining Figure IV.6c, we expect that this network could not fit the given x^3 with a single hidden unit, since the basis function used here is a sigmoidal function. This complexity regularization technique detected this automatically, growing the network at the appropriate time. This also occurs when $n = 5$, (see Figure IV.6e) where a third hidden unit was added to fit the 5 exemplars. The ΔISB method worked well along with this complexity regularization technique. The experiment concludes with the close fit illustrated in Figure IV.7.

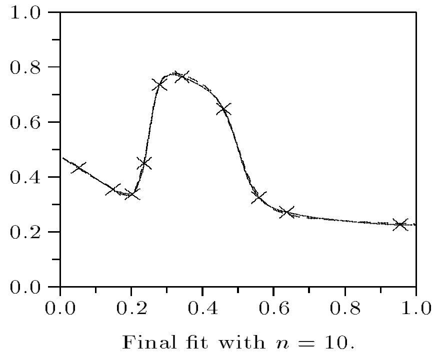


Figure IV.7: Final result of active selection on the skewed hump. 10 exemplars were required overall.

We compared this selection method with uniformly spaced exemplars. This task required 31 uniformly spaced exemplars in order to fit the candidates within the desired tolerance. The set of exemplars selected in this fashion is displayed along with the resulting trained network function in Figure IV.8. This figure illustrates how the uniformly spaced distribution required a close spacing of exemplars in order to sample rapidly changing portions of the function sufficiently well.

IV.C.5 Cost comparison

We now compare the cost of active selection with the baseline of training over all of the candidate examples upon the skewed hump learning task. This task required choosing the correct sized network for the task, so we applied the network growing method described above to the baseline method. We started each method with a single hidden unit, and added an extra unit if the exemplars have not been fit within tolerance after 10 restarts. (See Table IV.C.5.) Likely due to the number of restarts, the overhead of the ΔISB method was quite low, at 2.5%. Training upon actively selected exemplars required

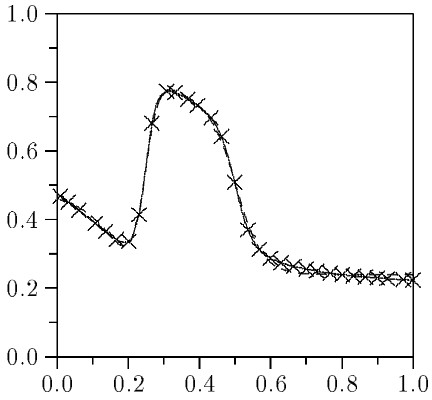


Figure IV.8: Selecting uniformly spaced exemplars required 31 exemplars to fit the skewed hump.

only 11.2% of the multiplies required by the baseline. This number of restarts gave active selection a clear advantage, as the bulk of the baseline cost was due to the high cost of the restarts when the network did not have sufficient complexity for the task.

To evaluate how much of the benefit of active selection is due to the number of restarts, we ran two more experiments in which the baseline was allowed to grow without any restarts if conjugate gradient fell into a local minimum without fitting the exemplars withing tolerance after a single run. This is not unreasonable, given the reliability of conjugate gradient, as we found that conjugate gradient would quite often fit a given candidate set on the first run, given a correctly specified network. However, occasionally conjugate gradient will fall into a local minimum, and so this approach is admittedly less reliable than using restarts. We grew the network trained by active selection using 5 restarts. This is the run illustrated in Figures IV.6a-i and IV.7; the results are tabulated in the second row of Table IV.C.5. On this particular run, conjugate gradient did select the correct size network using zero restarts, fitting the candidates on the first try with 3 hidden units. Active selection still required only 58.3% of the computation required by the baseline.

To evaluate the effect of having more candidate examples, we performed the experiment again using 500 candidates, again allowing the baseline method 0 restarts, while the active selection method used 5 restarts. However, in this case we had to run the baseline again to allow it to converge to the correct size network, as it was unable to fit the candidates on the first try when it grew to 3 hidden units. (We mark this result with an asterisk in Table IV.C.5. to remind that the cost of the previous run is not included.) This run gives a comparison of the typical cost of using active selection with the cost for the baseline when it does converge. The benefit of the ΔISB method was even greater than it was on the smaller set of candidates, requiring only 22.2% of the baseline cost to fit the 500 candidates. In this case, whereas the baseline failed to fit the candidates with smallest network on the first run, the ΔISB algorithm determined it correctly the first time.

To get a better comparison of average results, we ran 5 runs of each method upon a set of 200 candidates, and then repeated the comparison with 5 runs each upon a set of 500 candidates. Each of the four sets training runs was initialized using the same set of 5 initial randomized weights. For a conservative comparison we require the network trained by active selection determine the correct network complexity, but give the baseline method the advantage of being provided with the correct number of hidden units. The baseline method is restarted if it fails to fit the candidates. The cost of the restarts is summed into the total cost for each run. To eliminate any possible influence of our experience biasing the experiment through the initial choice of $\mathcal{E}_n$, we set $\mathcal{E}_n^{\max} = \mathcal{E}_N^{\max}$ in all of these runs. In our experience this is not the best initial setting for $\mathcal{E}_n$ but it is reasonable. (In Chapter V we present a refined algorithm that automatically regulates $\mathcal{E}_n$.) The results are tabulated in in Tables IV.4 and IV.5. Note that active selection typically required much less computation overall, even though it is responsible for the additional task of determining the appropriate network complexity. Note further that while the cost of the baseline method for $N = 500$ was typically much greater than for $N = 200$, the cost of active selection was typically only slightly higher (The average cost of active selection actually decreased.)

These runs (and in particular, Run 4 in Table IV.4) indicate how the choice of $\mathcal{E}_n$ can affect the cost of active selection. Observe that several of the runs selected more than 10 exemplars, the number that was found when $\mathcal{E}_n^{\max} = \mathcal{E}_N^{\max}/10 = 0.001$ (see Table IV.C.5). However, the typical cost of these runs was still on par with the cost that resulted from using $\mathcal{E}_n^{\max} = 0.001$. The exception is Run 4 in Table IV.4, which cost more than the baseline. We repeated this particular run, reducing $\mathcal{E}_n^{\max}$ by half to 0.005. This improved the cost substantially, resulting in 32.24 million multiplies instead of 78.28 million, selecting 12 exemplars rather than 15.

Learning skewed hump.						
	Baseline		Active selection			
N	cost	restarts	cost	restarts	overhead	n
200	495	10	55.5	10	2.5%	11
200	56.1	0	32.7	5	3.2%	10
500	286*	0	63.4	5	4.5%	11

Table IV.3: Comparing the cost of active selection with the baseline of using all of the available examples. The task is to learn the skewed hump generated by Network 2. The first column gives the number of candidates, N. The second pair of columns pertain to the baseline of training on all N candidates. Cost is measured in millions of multiplies. This task involves choosing the correct size network, so the maximum number of restarts is given. The next set of columns apply to active exemplar selection. The total cost is given first; the portion of this used by active selection is given next, followed by the number of exemplars selected, n, and finally the number of restarts used during training. Note*: the third baseline run entered a local minima without fitting x^N on the first try, and required a restart to converge; the number entered here is the cost of the second run.

Skewed hump, N = 200.				
Run	Baseline cost	Active selection cost	overhead	n
1.	83.30	33.46 (40.2%)	2.60%	11
2.	381.43	31.46 (8.2%)	5.73%	16
3.	80.37	18.52 (23.0%)	6.49%	12
4.	61.08	78.28 (128.2%)	2.16%	15
5.	70.96	34.50 (48.6%)	4.81%	14
Med:	**80.37**	**33.46 (41.6%)**	**4.81%**	

Table IV.4: Comparison of the cost of learning the skewed hump function using 200 candidate training examples. The "Baseline" column gives the total cost of training over all 200 examples. The next set of columns pertain to active exemplar selection. The total cost is given along with the ratio of the total cost with the baseline cost in parentheses. The "overhead" gives the portion of the total cost due to exemplar selection. The number of exemplars selected is given by n. Cost is measured in millions of multiplies. The bottom row gives medians over the five runs. Baseline runs 1 and 2 required 1 and 2 restarts, respectively. All the other baseline runs fit the candidates within tolerance on the first attempt.

Skewed hump, N = 500.				
Run	Baseline cost	Active selection cost	overhead	n
1.	940.68	37.26 (4.0%)	8.56%	12
2.	182.67	33.17 (18.2%)	8.86%	11
3.	778.51	30.52 (3.9%)	17.20%	17
4.	1287.39	38.05 (3.0%)	12.51%	16
5.	241.18	44.43 (18.4%)	9.48%	15
Med:	**778.51**	**37.26 (4.8%)**	**9.48%**	

Table IV.5: Comparison of the cost of learning the skewed hump function using 500 candidate training examples. The "Baseline" column gives the total cost of training over all 500 examples. The next set of columns pertain to active exemplar selection. The total cost is given along with the ratio of the total cost with the baseline cost in parentheses. The "overhead" is the portion of the total cost due to exemplar selection. The number of exemplars selected is given by n. Cost is measured in measured in millions of multiplies. The bottom row gives medians over the five runs. Baseline runs 2 and 5 fit the candidates on the first attempt. Baseline runs 1, 3, and 4 required 3, 1, and 2 restarts, respectively.

IV.D Discussion

We now turn to issues relevant to the practical application of our selection method. We emphasize the importance of obtaining a good solution to the training problem at each stage. We experimented briefly with an iterative training regime, in which little optimization was performed between successive examples. There was a tendency for the criterion to sample repetitively near previously sampled points when the optimization between successive examples was not stringent. In some such cases a pathological condition emerged in which the technique entered a cycle, endlessly alternating between two examples. For this reason, we advocate stringent optimization at each step.

We now address the overhead due to the selection process. The cost of selecting a new exemplar is about the same as that of making two iterations through the entire set of candidates with a steepest descent learning rule. This can be cut in half, at the expense of allocating space equal to the size of the set of candidates, by storing partial results and making one iteration through the set of candidates. However, we expect that this will be unreasonable for large data sets. When dealing with large data sets the cost of exemplar selection could be reduced by appropriately subsampling the set of candidates [78].

Observe that the selection cost is incurred only occasionally, because typically network training will process a given set of exemplars for numerous iterations before selecting a new exemplar. When multistarts are used to determine the appropriate network complexity for the task several restarts might be performed before a new exemplar is selected. In our experience we have noticed that when multistarts are used the overhead of exemplar selection tends to be a small percentage of the total cost of training.

In our experience with the technique, we have noticed that the ΔISB criterion can have multiple maxima. It may be possible to exploit this to try to reduce the cost of obtaining exemplars further by selecting more than one exemplar at a time. Rather than just selecting the single exemplar maximizing our criterion, we might select a *set* of exemplars that locally maximize the criterion [47]. This reduces to the task of finding the modes of the selection criterion.

Arbitrary decisions were made in the derivation of the active selection method we obtained. We now discuss the effects of these choices, and how they might be modified to obtain variants of the method for practical use on particular applications. It is an open question as to how sensitive the method is to $H(x^n, w_n)$, the estimate of the Hessian appearing in (IV.8). Observe that the appearance of H is due to our choice of weight update. This choice could be modified, depending upon the degree of approximation that is deemed necessary for the particular learning task. Our experiments demonstrate that the method can work well even when H is approximated. Recall that in our demonstrations we approximated or replaced H for computational simplicity. This was sufficient for the intended purpose of illustrating the method, although we advocate calculating H directly whenever feasible. That the method still selected concise training sets and saved computation indicates that even the crude approximation of H we used may be adequate on some tasks.

When obtaining the most concise set of exemplars is the highest priority, we expect that calculating H directly will be desirable. We may go even further, calculating the Hessian directly, taking appropriate steps to detect when it becomes ill-behaved. The numerical optimization literature offers considerable guidance in this regard. It is a straightforward modification to use the actual Hessian in place of H. This is equivalent to replacing the Gauss-Newton weight update in our derivation with the Newton update.

We employ the least squares performance criterion. It would be straightforward to replace it with another choice. For example, one might use cross entropy for use on classification tasks [97]. The estimator could be modified to include a (differentiable) complexity regularization term as well, such as weight decay [23]. In this case, exemplars would be selected according to their potential worth in minimizing the sum of network complexity and estimation error.

Finally, we summarize our initial experience with the technique on more difficult learning tasks. Space limitations prevent us from discussing all of the experiments we have done so far, but we have applied the technique to learning tasks requiring networks with multiple inputs and outputs with similarly favorable results. More extensive experiments are necessary to corroborate these preliminary results, especially on learning tasks requiring high dimensional data sets.

Chapter V

Experience With Exemplar Selection on Clean Data

V.A Summary

In Chapter IV we develop an *active exemplar selection* method called ΔISB for the case in which training examples are (essentially) noiseless. The method is *active* in that it utilizes network state during learning. Given a fixed set of examples, the method selects a concise subset for training. Fitting the selected exemplars results in the entire set being fit as well as desired. The algorithm also incorporates a method for regulating network complexity, automatically adding exemplars and hidden units as needed.

Here, we present empirical results from applying the technique to predicting two chaotic time series, one with fractal dimension 2.1, the other with fractal dimension 3.5. Note that proper embedding of clean chaotic time series data results in training examples that are essentially noise-free. The results indicate that the overhead of exemplar selection is small compared to the total training cost, that the method routinely obtains small training sets (about 25 for the less difficult time series, from 81 to 97 for the more difficult one, depending upon the fit required). Total training cost is reduced, requiring an order of magnitude fewer floating point operations than training on the entire set of examples. There was also some evidence of improved generalization (which here, means improved interpolation in the embedding space).

We also compare ΔISB with three contending exemplar selection techniques, one being an ad hoc heuristic similar to ΔISB that also qualifies as active exemplar selection. The two active selection methods performed best overall. When tested on the more difficult time series, the ad hoc method performed poorly where ΔISB continued to work well.

The ratio of actively selected exemplars to the number of weights was just over 1.0, providing corroboration of recent lower bounds on number of exemplars necessary for training.

V.B Introduction

In Chapter IV we develop a method of *active selection* of training exemplars for network learning. Active selection uses information about the state of the network when choosing new exemplars. The approach uses the statistical sampling criterion Integrated

Squared Bias (ISB) to derive a "greedy" selection method picking the training example that maximizes the decrement in this measure. (ISB is a special case of the more familiar Integrated Mean Squared Error in the case that noise variance is zero. Although our method was not developed for statistical regression on noisy data, note that many practical problems can be cast in the form of the interpolation learning task considered here; e.g., feedforward networks used for learning identity mappings (i.e., autoassociation) are learning from clean data (cf. [32]. See also Chapter VI, where criteria are proposed for evaluating noisy training according to estimates of IMSE). We refer to this method as ΔISB. The method automatically regulates network complexity by growing the network as necessary to fit the selected exemplars, and terminates when the model fits the entire set of available examples to the desired accuracy. Hence the method is a nonparametric regression technique. In this paper we explore the possible benefits of active exemplar selection by comparing the method with several other methods on a well known learning task, the Mackey-Glass time series prediction task. Note that this learning task generates noiseless examples, if the time series is properly embedded into example space. We first compare ΔISB with the method of training on all the examples. ΔISB consistently learns the time series from a small subset of the available examples, finding solutions equivalent to solutions obtained using all of the examples. The networks obtained by ΔISB consistently perform better on test data for single step prediction, and do at least as well at iterated prediction, but are trained at much lower computational cost.

Having demonstrated that this particular type of exemplar selection is worthwhile, we compare ΔISB with three other exemplar selection methods which are easier to code and cost less to compute. We compare the total cost of training, as well as the size of the exemplar sets selected. One of the three contending methods is an ad hoc method similar to the ΔISB algorithm, and is also an active selection technique, as its calculation involves the network state. Among the four exemplar selection methods, we find that the two active selection methods provide the greatest computational savings and select the most concise training sets, although the ad hoc active selection method can fail where ΔISB continues to work well. There was also evidence of improved generalization. Finally, an analysis of the number of exemplars selected when growing occured showed that active selection tends to require about one exemplar per weight.

V.C The Method

We are provided with a set of N "candidate" examples of the form $(x_i, g(x_i))$. Given g, we can denote this as x^N. Let $f(\cdot, w)$ denote the network function parameterized by weights w. For a particular subset of the examples denoted x^n, let $w_n = w_n(x^n)$ minimize

$$\frac{1}{n}\sum_{i=1}^{n}(g(x_i) - f(x_i, w))^2.$$

Let w^* be the "best" set of weights, which minimizes

$$\int (g(x) - f(x, w^*))^2 \mu(dx),$$

where μ is the distribution over the inputs. Our objective is to select a subset x^n of x^N such that $n \ll N$, while minimizing $\int (f(x, w_n) - f(x, w^*))^2 \mu(dx)$. Thus, we desire a subset representative of the whole set. To this end, for a particular choice of n we choose $x^n \subset x^N$ giving weights w_n minimizing Integrated Squared Bias (ISB):

$$ISB(x^n) = \int (g(x) - f(x, w_n))^2 \mu(dx). \tag{V.1}$$

We generate x^n incrementally. Given a candidate example $\check{x}_{n+1}$, let $\check{x}^{n+1} = (x^n, \check{x}_{n+1})$. Selecting x^1 optimally with respect to (V.1) is straightforward. Then given x^n minimizing $ISB(x^n)$, we opt to select $x_{n+1} \in x^N$ maximizing $ISB(x^n) - ISB(\check{x}^{n+1})$. Note that using this property for x_{n+1} will not necessarily deliver the globally optimal solution. Nevertheless, this approach permits a computationally feasible and attractive method for sequential selection of training examples.

Choosing x_{n+1} to maximize this decrement directly is expensive. We use the following simple approximation (see [86] for justification): Given x^n, select

$$x_{n+1} \in \arg\max_{\check{x}_{n+1}} \Delta ISB(\check{x}_{n+1}|x^n),$$

where

$$\Delta ISB(\check{x}_{n+1}|x^n) = \Delta\check{w}_{n+1}{}' \sum_{i=1}^{N} \nabla_w f(x_i, w_n)(g(x_i) - f(x_i, w_n)),$$

$$\Delta\check{w}_{n+1}{}' = (g(\check{x}_{n+1}) - f(\check{x}_{n+1}, w_n))' \nabla_w f(\check{x}_{n+1}, w_n)' [H(x^n, w_n)]^{-1},$$

and

$$H(x^n, w_n) = \sum_{i=1}^{n} \nabla_w f(x_i, w_n) \nabla_w f(x_i, w_n)'.$$

In practice we approximate H appropriately for the task at hand. Although we arrive at this criterion by making use of approximations valid for large n, this criterion has an appealing interpretation apart from large sample considerations. Specifically, this method can be viewed as picking the single example having individual error gradient most highly correlated with the average error gradient of the entire set of examples. Learning with this example is therefore likely to be especially informative about the entire set, relative to other available examples. The ΔISB criterion thus possesses heuristic appeal in training sets of any size. As n becomes large, ΔISB approaches $ISB(x^n) - ISB(\check{x}_{n+1})$, providing further support for its use.

V.D The Algorithm

Before presenting the algorithm we first explain certain implementation details.

We integrated the ΔISB criterion with a straightforward method for regulating network complexity. We begin with a small network, and with an initial training set composed of a single exemplar. The procedure regulates network complexity appropriately as new exemplars are added by adding hidden units when the network fails to fit the current training set. When training stalls (by reaching a non-satisfactory minimum,) we randomize network weights and restart training. (For detecting stalls, see for instance the optimization code in [89].) We grow the network when it stalls 5 times without fitting the data, by adding another unit to each hidden layer.

Before we can select a new exemplar, we require that the network fit the current training set "sufficiently well." Let $\mathcal{E}_n(\tilde{x}^m)$ measure the rmse (root mean squared error) network fit over m arbitrary examples $\tilde{x}^m$ when trained on x^n. Let $\mathcal{E}_n \in \Re^+$ denote the rmse fit we require over the current set of n exemplars before selecting a new one. Let $\mathcal{E}_N \in \Re^+$ denote the rmse fit desired over all N examples. (Our goal is $\mathcal{E}_n(x^N) \leq \mathcal{E}_N$.) It typically suffices to set $\mathcal{E}_n = \mathcal{E}_N$, that is, to train to a fit over the exemplars which is at least as stringent as the fit desired over the entire set (normalized for the number of exemplars). However, active selection sometimes chooses a new exemplar "too close" to previously selected exemplars even when this is the case. This is easy to detect, and in this case we reject the new exemplar and continue with training.

We use an "exemplar spacing" parameter d to detect when a new exemplar is too close to a previous selection. Two examples x_i and x_j are "close" in this sense if they are within Euclidean distance d, and if additionally $|g(x_i) - g(x_j)| \leq \mathcal{E}_N$. The additional condition allows the new exemplar to be accepted even when it is close to a previous selection in input space, provided it is sufficiently far away in the output space. In our experiments, the input and output space are of the same scale, so we set $d = \mathcal{E}_N$. When a new selection is too close to a current exemplar, we reject the new selection, reduce $\mathcal{E}_n$ by 20%, and continue training, resetting $\mathcal{E}_n = \mathcal{E}_N$ when a subsequent selection is appended to the current training set. (While there are many alternatives at this point, this method has worked well in practice. [87])

The Algorithm

We now outline the algorithm:

Initialize:

- Specify user-set parameters: initial network size, the desired fit $\mathcal{E}_N$, the exemplar spacing parameter, and the maximum number of restarts.
- Select the first training set, $x^1 = \{x_1\}$. Set $n = 1$ and $\mathcal{E}_n = \mathcal{E}_N$. Train the network on x^1 until $\mathcal{E}_n(x^1) \leq \mathcal{E}_n$.

While$(\mathcal{E}_n(x^N) > \mathcal{E}_N)$
{
    Select a new exemplar, $x_{n+1} \in x^N$, maximizing ΔISB.
    If (x_{n+1} is "too close" to any $x \in x^n$)
    {
        Reject x_{n+1}
        Reduce $\mathcal{E}_n$ by 20%.
    }
    Else
    {
        Append x_{n+1} to x^n.
        Increment n.
        Set $\mathcal{E}_n = \mathcal{E}_N$.
    }
    While$(\mathcal{E}_n(x^n) > \mathcal{E}_n)$
    {
        Train the network on the current training set x^n,
        restarting and growing as necessary.
    }
}

V.E The Problem

V.E.1 Data Generation

We generated the data from the Mackey-Glass equation [74]

$$\dot{x}(t) = \frac{ax(t-\tau)}{1 + x(t-\tau)^{10}} - bx(t).$$

with $a = 0.2$, and $b = 0.1$. We integrated the equation using fourth order Runge-Kutta with step size 0.1. Runge-Kutta integration uses a history, which we initialized to 0.5. We generated three data sets, the first two obtained by setting $\tau = 17$. We iterated the equation for 100 time steps before beginning sampling; this marks $t = 0$. The next 1000 time steps comprise Data Set 1, illustrated in Figure V.1. We generated Data Set 2 from the 2000 examples following $t = 5000$. The first 1000 time steps of Data Set 2 are plotted in Figure V.2, for comparison with Data Set 1. Data Set 3 is obtained by setting $\tau = 30$, iterating the equation for 1000 time steps before sampling, and sampling the next 5100 values. This yields a much more irregular time series (cf. Figure V.3).

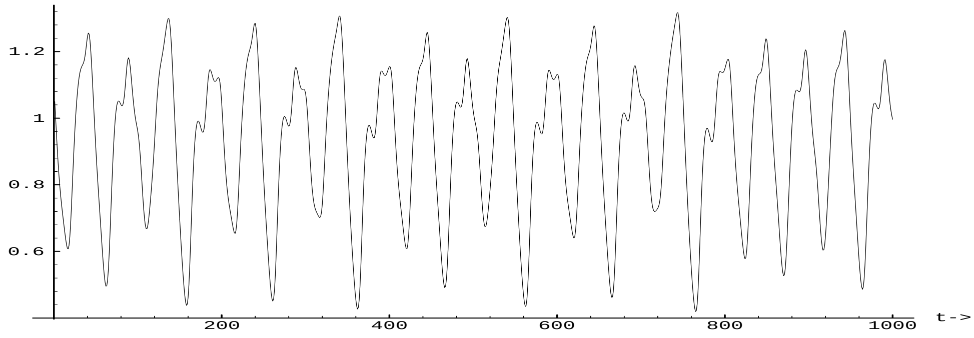


Figure V.1: Data Set 1.

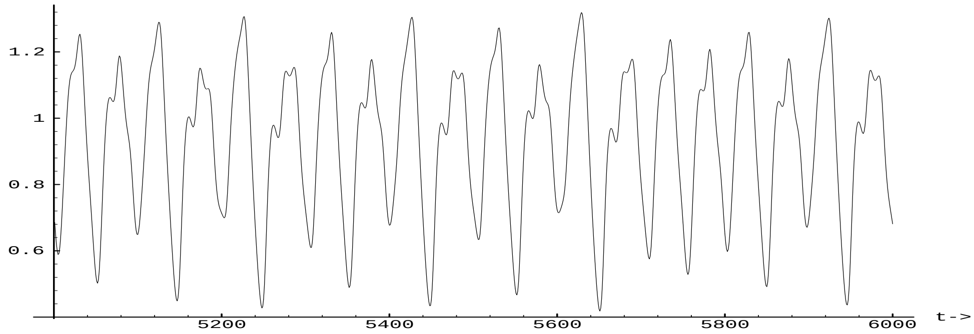


Figure V.2: First half of Data Set 2.

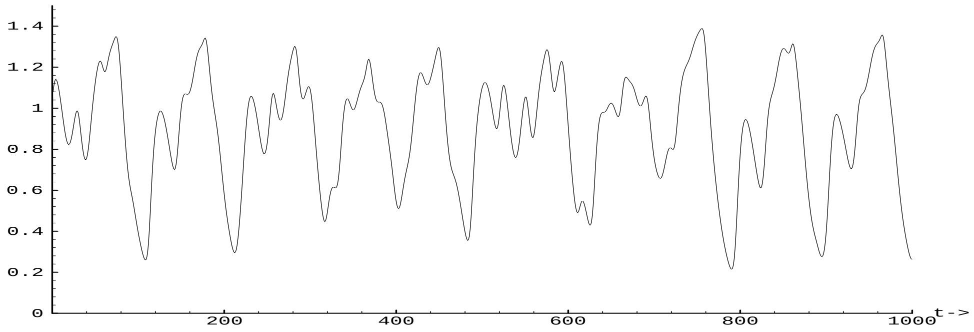


Figure V.3: First 1000 observations of Data Set 3.

V.E.2 Training

We used the usual feedforward network architecture, comprised of fully connected layers whose units' outputs are given by squashing a weighted sum over their respective inputs (cf. [87]). We used the usual squasher, a sigmoid taking values in the interval $[0, 1]$. We used conjugate gradient optimization for all of the training runs. This uses a line search to automate step size adjustment. The line search routine typically required 5 to 7 passes through the data set for each downhill step, and was restricted to use no more than 10.

For learning the first two data sets, the network is provided with 4 values sampled at $t = 0, -6, -12, -18$ and trained on a target value at $t = 6$ [62]. We used two network configurations, one with a single hidden layer, the other with two hidden layers. Initially, the single hidden layer network has a single hidden unit, and the 2 hidden layer network has 2 units per hidden layer. A unit is added to each hidden layer when growing either architecture. All methods use the same growing procedure. Thus, other exemplar selection techniques are implemented by modifying how the next training set is obtained at the beginning of the outer while loop. The method of using all the training examples essentially uses the inner while loop.

In preliminary experiments we evaluated sensitivity of ΔISB to the calculation of H. We compared two ways of estimating H, in terms of the number of exemplars selected and the total cost of training. The first approach uses the diagonal terms of H [87]. The second approach replaces H with the identity matrix. Evaluated over 10 separate runs, fitting 500 examples to an nrmse of 0.044, ΔISB gave similar results for both approaches, in terms of total computation used, as well as in the number of exemplars selected. Since it seemed sufficient for this learning task, we used the second approach for all of the comparisons.

V.F The Comparisons

We performed a number of experiments, each comparing the ΔISB algorithm with competing training methods. The competing methods include the conventional method of using all the examples, henceforth referred to as "the strawman," as well as four other contending data selection techniques. In each comparison we counted the total number of floating point operations as a portable measure of the total cost of training, counting each add, multiply, and divide used during training. It turns out to be sufficient to report only the number of multiplies, as the number of adds and divides was always proportional to this count. Henceforth, cost is always measured in millions of multiplies.

For each comparison we ran two sets of experiments. The first compares the total cost of the competing methods as the fit requirement is varied. The second compares the cost as the size of the "candidate" set (the set of available examples) is varied. To ensure that each method is achieving a comparable fit over novel data, we evaluated each network over a test set. The generalization tests also looked at the iterated prediction error (IPE) over the candidate set and test set [62]. Here we start the network on the first example from the set, and feed the output back into the network to obtain predictions in multiples of 6 time steps. Finally, for each of these we compare the final network sizes.

V.F.1 Varying the Desired Fit

We trained to three tolerances: 0.088, 0.066, and 0.044 nrmse (0.02, 0.015, and 0.01 rmse, respectively). Normalized rmse (nrmse) is given by the rmse divided by the standard deviation of the data. We will continue to measure error in terms of nrmse unless otherwise specified. These tolerances correspond to rather tight fits. For networks trained to 0.044 tolerance, the difference between the network and desired output is barely discernable when plotted together. The candidate set for these experiments is comprised of the first 500 examples chosen from the Data Set 1. We used the second 500 examples as a test set to corroborate generalization (performance over novel data). At each tolerance, we ran 5 different training runs.

V.F.2 Varying the Candidate Set Size

Here we vary the number of available examples. We derived 5 candidate sets, comprised respectively of the first 500, 625, 750, 875, and 1000 examples of Data Set 1. We proceeded similarly to the comparison over varying tolerances, training 5 networks to a tolerance of 0.044 for each of the 5 different size candidate sets. We used Data Set 2 for the test set on all these runs.

V.G Comparison With Using All the Examples

We first compare ΔISB with the conventional method of using all the available examples, which we will refer to as "the strawman." Note, however, that our "strawman" method does differ from conventional approaches in that we automate the selection of network size by incrementing the number of hidden units as necessary to achieve the desired result. The cost of growing is included in our comparisons.

V.G.1 Varying the Desired Fit

We ran this experiment on both network architectures. When using the single hidden layer architecture, each method required 2 hidden units for a fit of 0.088 and 0.066 nrmse, and from 3 to 6 (typically 4) hidden units for a fit of 0.044 rmse. For the two hidden layer architecture, each method required 2 units per hidden layer for a fit of 0.088 and 0.066 rmse, and from 3 to 4 (typically 3) units per hidden layer for a fit of 0.044 rmse.

While both methods did quite well on the generalization tests, ΔISB clearly did better. Whereas the strawman networks do slightly worse on the test set than on the candidate set, networks trained by ΔISB tended to give test set fits close to the desired (training) fit. For instance, at the 0.044 training tolerance, networks trained on all the examples typically fit the test set to 0.048 nrmse, whereas ΔISB typically gave a fit of 0.044 over the test set. This indicates that the networks trained by ΔISB always learned the task at least as well as the strawman. This is partially due to the control flow of the algorithm, which often fits the candidate set better than necessary. (Fitting a concise set of selected exemplars to the desired tolerance often results in a candidate set fit that is better than desired, which can then give better test set performance.) However, we also observed networks trained by ΔISB that exhibited a test set fit better than the fit given by the same network over the candidate set. This occurred 7 times over these 15 training runs, and never occurred over any of the strawman runs.

Overall, networks trained by ΔISB performed at least as well as the strawman with respect to IPE. Figure V.4 shows the test set plotted along with the iterated prediction of a network trained by ΔISB to a fit of 0.088 nrmse, giving an IPE of 0.678 nrmse (0.153 rmse,) which was the median IPE observed for this set of five runs. ΔISB required 12 exemplars to obtain this fit. (The strawman achieved a median IPE of 0.880 nrmse, which of course required training on all the exemplars.) Figure V.5 shows the test set plotted along with the iterated prediction of a network trained by ΔISB to a fit of 0.044, giving an IPE of 0.357 nrmse (0.081 rmse,) the median IPE observed for this set of five runs. This fit required 22 exemplars. (The strawman achieved a median IPE of 0.355 nrmse.)

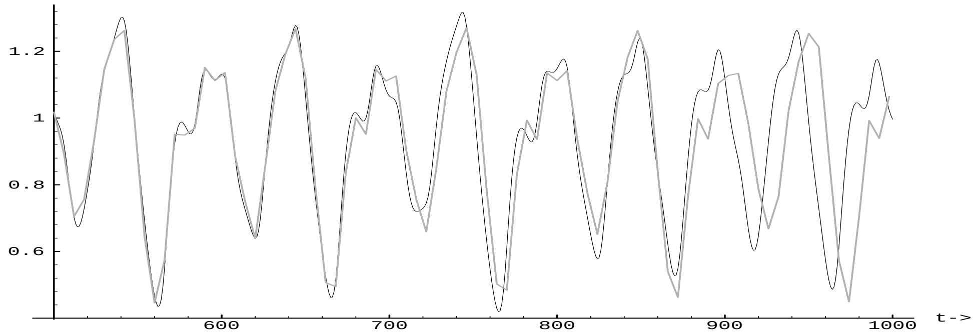


Figure V.4: Iterated prediction of a 2 hidden layer network trained by ΔISB to 0.088 nrmse, requiring 2 units per hidden layer and 12 exemplars. The thinner dark line is the desired output, the dotted line is the iterated prediction in steps of 6. The IPE for this figure is 0.673 nrmse.

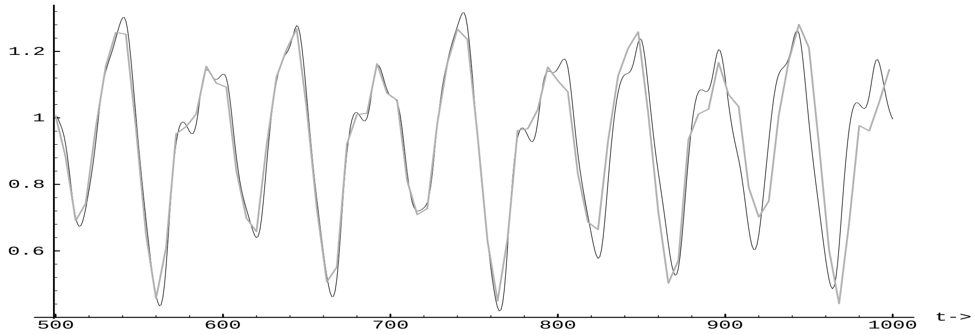


Figure V.5: Iterated prediction of a 2 hidden layer network trained by ΔISB to 0.044 nrmse, requiring 3 units per hidden layer and 22 exemplars. The thinner dark line is the desired output, the dotted line is the iterated prediction in steps of 6. The IPE for this figure is 0.356 nrmse.

These effects suggested a closer analysis, which we perform in Section V.I. At the least, they indicate that the networks trained on data selected by ΔISB performed at least as well as the strawman, ensuring the validity of our cost comparisons.

Figure V.6 summarizes the cost for the two hidden layer architecture; the results for the single hidden layer architecture are similar. We plot the average total cost, in millions of multiplies, against the training tolerance for each method. Each point represents an average over 5 training runs. The error bar widths are twice the standard deviation. For the single hidden layer architecture, the strawman required 201, 483, and 4,210 million multiplies for tolerances of 0.088, 0.066, and 0.044, respectively, whereas ΔISB required 5, 21, and 206 million multiplies, respectively. For the two hidden layer architecture, the strawman required 109, 115, and 4,740 million multiplies for the respective tolerances, whereas ΔISB required 8, 28, and 219 million multiplies, respectively. These results indicate how severely the strawman is penalized by a tighter tolerance requiring a larger network, relative to ΔISB, for which the cost of each restart is reduced due to the smaller number of exemplars processed. However, it should be noted that overall, the cost of growing was about the same order of magnitude as training the final network; therefore, the strawman performs badly even when it is given a correctly sized network.

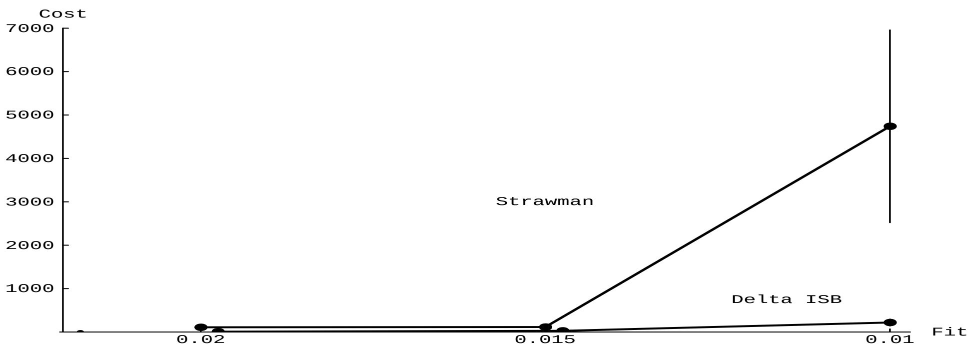


Figure V.6: Cost (millions of multiplies) versus Desired Fit (nrmse), ΔISB and "Strawman." Height of error bars is twice the standard deviation.

V.G.2 Varying the Candidate Set Size

Due to computer time constraints we performed this comparison using only the two hidden layer architecture. Preliminary runs, as well as the results reported in the previous section, indicate that the results will be similar for a single hidden layer architecture.

Data Set 2 was used for the test set for these comparisons. Recall that Data Set 2 is sampled starting from $t = 5000$. Generalization tests were excellent for both methods, although ΔISB was again better overall. Networks trained by ΔISB performed better on the test set than they did on the candidate set 9 times out of the 25 runs; this never occurred for the strawman. (See Section V.I for closer analysis of generalization results.) Figure V.7 shows the iterated prediction for a 2 hidden layer network trained to 0.044 nrmse on the candidate set with 500 examples, over the first 500 time steps of Data Set 2. This is the same network used to obtain Figure V.5, giving an IPE of 0.097 nrmse (0.022 rmse,) whereas the IPE over the second half of Data Set 1 was 0.356 nrmse.

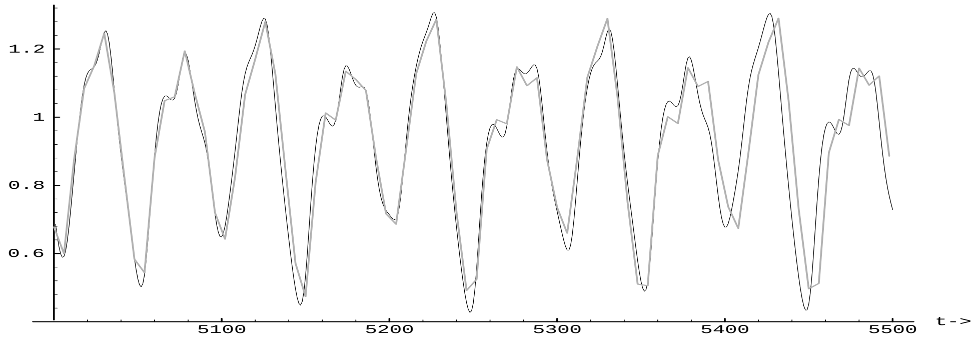


Figure V.7: Iterated prediction of a 2 hidden layer network trained by ΔISB to 0.044 nrmse, over first 500 time steps of Data Set 2. This is the same network used in Figure V.5. The IPE for this figure is 0.097 nrmse.

For candidate sets of size 500, 625, 750 and 875, each method typically required 3 to 4 units per hidden layer, typically 3. ΔISB tended to select smaller networks. In particular, the strawman method had difficulty on the candidate set of 1000 examples, selecting networks with 4 to 6 units per hidden layer. In comparison, the ΔISB method selected networks with more than 3 units per hidden layer (but never more than 4) only 4 times out of 25, where as the strawman required more than 3 units per hidden layer 10 out of the 25 runs. (See Section V.J for closer analysis of network complexity.) This affected the cost dramatically, as can be seen in Figure V.8. This suggests that the growing technique is more likely to fit the data with a smaller network when exemplar selection is used.

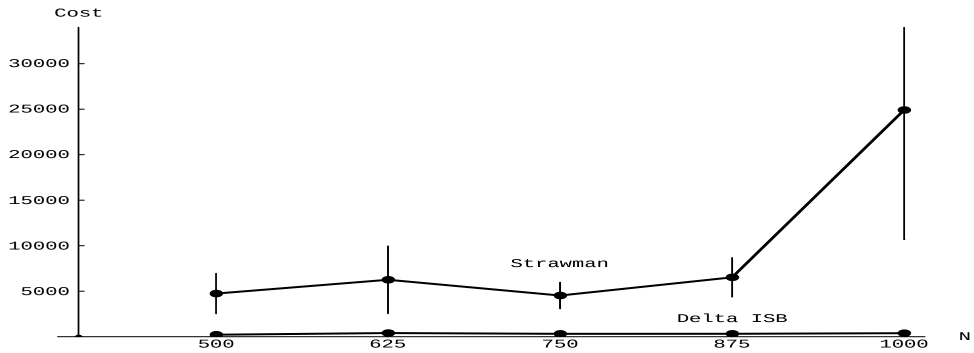


Figure V.8: Cost (millions of multiplies) versus candidate set size ("N") for ΔISB and "Strawman." Height of error bars is twice the standard deviation.

V.H Comparisons With Contending Data Selection Techniques

The results above clearly demonstrate that exemplar selection can cut the cost of training dramatically. In what follows we compare ΔISB with three other ways of selecting training exemplars. Each of these methods is easier to code and cheaper to compute than ΔISB, and are considerably more challenging contenders than the strawman. In addition to comparing the overall training cost we will also evaluate their data compression ability by comparing the size of the exemplar sets each one selects. Since we do not know the optimal size of the data set each method should select, we proceed in the same manner as with ΔISB, sequentially growing the training set as necessary, until the candidate set is fit as desired.

Two of these contending techniques do not depend upon the state of the network, and are therefore are not "Active Selection" methods. *Random Selection* selects an example randomly from the candidate set, without replacement, and appends it to the current exemplar set. *Uniform Grid* exploits the time series representation of our data set to select training sets composed of exemplars evenly spaced at regular intervals in time. Note that *Uniform Grid* does not append a single exemplar to the training set, rather it may select an entirely new set of exemplars each time the training set is grown. Note further that this technique relies heavily upon the type of learning task here, that of predicting time series data, as the problem of selecting exemplars uniformly spaced in the 4 dimensional input space would be much more difficult to compute.

The third method, "*Maximum Error*," was suggested by the ΔISB algorithm, and is also an Active Selection technique, since it uses the network in selecting new exemplars. Note that the error between the network and the desired value is a component of the ΔISB criterion. ΔISB need not select an exemplar for which network error is maximum, due to the presence of terms involving the gradient of the network function. In comparison, the *Maximum Error* method selects an exemplar maximizing network error, ignoring gradient information entirely. It is cheaper to compute, typically requiring an order of magnitude fewer multiplies in overhead cost as compared to ΔISB. This comparison will test, at least for this particular learning task, whether the gradient information is worth the additional overhead it requires.

V.H.1 Comparison with Random Selection

Random Selection fared the worst among the four contenders. However, it still performed better overall than the strawman method. This is probably because the cost due to growing is cheaper, since early on restarts are performed over small training sets. As the network fit becomes better, the likelihood of randomly selecting an informative exemplar decreases, and *Random Selection* typically reaches a point where it adds exemplars in rapid succession, often doubling the size of the exemplar set in order to attain a slightly better fit.

Figure V.9 plots the cost comparison of ΔISB with *Random Selection* as the desired tolerance varies, for a network with 2 hidden layers. Note the large error bars on the *Random Selection* line, indicating that the cost of training in this way can vary wildly, depending upon how lucky *Random Selection* is in selecting a small exemplar set. This is indicated by Figure V.10, which plots the corresponding number of exemplars selected by each method. Recall that the candidate set contains 500 examples. *Random Selection* tends to require a large portion of the examples, and the error bars indicate the wide variance in the number of exemplars selected. The number of exemplars selected by ΔISB increases slightly in comparison. Figure V.11 shows the cost comparison between ΔISB and *Random Selection* as the size of the candidate set varies, with Figure V.12 showing the corresponding numbers of exemplars selected. *Random Selection* also performs worse than ΔISB on generalization tests. Over the total of 40 generalization tests (15 for varying tolerance, 25 for varying candidate set size) only twice did it give a network that gave a fit on test data better than its fit over the candidate set from which it was trained.

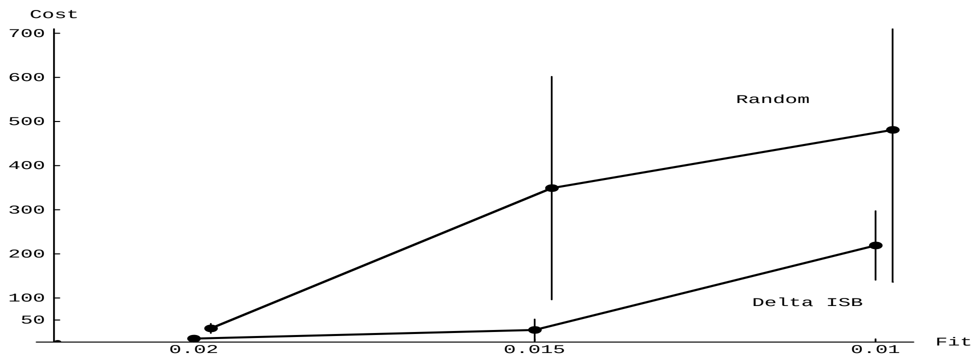


Figure V.9: Cost (millions of multiplies) versus Desired Fit, in rmse, for ΔISB and Random Selection. Height of error bars is twice the standard deviation. Both methods used the same tolerances: the lines are offset slightly so that the error bars don't cover each other. Note: "cost" axis is on a different scale than Figure 6.

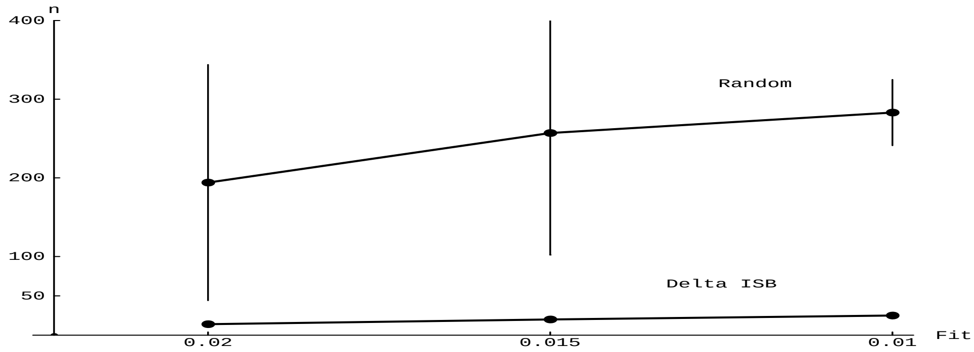


Figure V.10: Number of exemplars selected (n) versus Desired Fit (in rmse), for ΔISB and Random Selection. Height of error bars is twice the standard deviation.

V.H.2 Comparison with Uniform Grid and Max Error

Uniform Grid and *Maximum Error* are comparable with ΔISB in cost as well as in the size of the selected exemplar sets. Overall, ΔISB and *Maximum Error* performed about the same, with *Uniform Grid* finishing respectably in third place. *Maximum Error* was comparable to ΔISB in generalization also, doing better on the test set than on the candidate set 10 times out of 40, whereas ΔISB did so a total of of 16 times. This occurred only 3 times out of 40 for *Uniform Grid*.

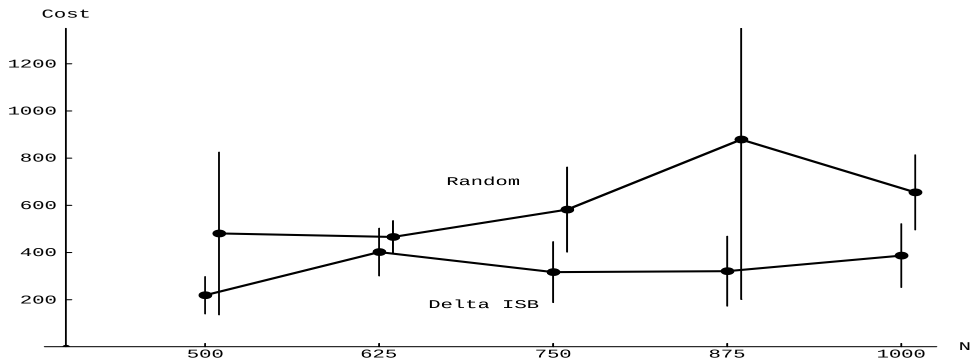


Figure V.11: Cost (millions of multiplies) versus N (candidate set size,) for ΔISB and Random Selection, for the 0.044 nrmse (0.01 rmse) fit. Height of error bars is twice the standard deviation. Both methods used the same tolerances: the lines are offset slightly so that the error bars don't cover each other.

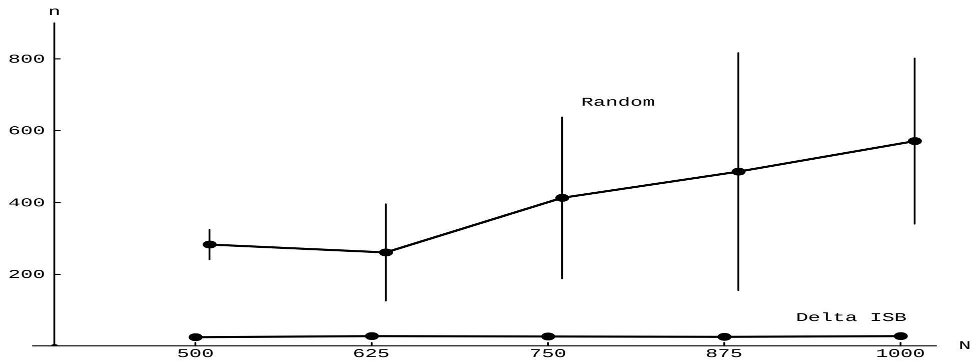


Figure V.12: Number of exemplars selected (n) versus candidate set size (N,) for ΔISB and Random Selection, using 0.044 nrmse (0.01 rmse) as the desired fit.

Figure V.13 shows that the cost of each of the three techniques increases as the desired fit is tightened, although it seems that *Uniform Grid* does so more quickly. Figure V.13 shows that *Uniform Grid* requires more exemplars at all three tolerances, whereas ΔISB and *Maximum Error* select about the same number, requiring about 15, 20, and 25 exemplars, respectively, for a desired fit of 0.088, 0.066 and 0.044.

Figure V.15 shows that the total cost of training for ΔISB, *Maximum Error*, and *Uniform Grid* is fairly comparable. Averaged over all 25 runs, *Uniform Grid* required 444 million multiplies, *Maximum Error* required 340 million multiplies, and ΔISB required 329 million multiplies. *Uniform Grid* also suggests a trend upwards as the candidate set size increases, whereas *Maximum Error* and ΔISB are relatively flat in comparison. Similar results were obtained over the single hidden layer architecture.

Figure V.16 shows that *Uniform Grid* typically requires about twice as many exemplars as the other two. *Maximum Error* and ΔISB selected about the same number of exemplars. Note that the number of exemplars selected by the two Active Selection methods is relatively flat as the number of candidate examples increases, typically selecting about 25 exemplars, plus or minus two. Tables 1 and 2 summarize the results for the above experiments, averaging the results over all runs.

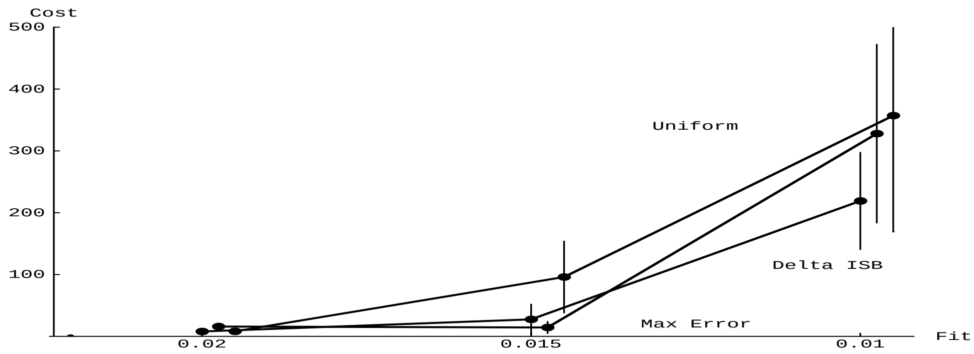


Figure V.13: Cost (millions of multiplies) versus Desired Fit (in rmse,) for ΔISB, Uniform Grid, and Maximum Error. Height of error bars is twice the standard deviation. All methods used the same tolerances: the lines are offset slightly to separate the error bars.

V.I Evidence of Improved Generalization

Active selection can improve generalization performance. We examined the ratio of test set error to training set error for the runs just described above. The active selection methods consistently gave test set performance as good as and often better than the training set performance. Table 3 tabulates the results. The fact that the ratio is less than 1.0 for the active selection methods is less important than the fact that the generalization of these methods is better than that of the other contenders.

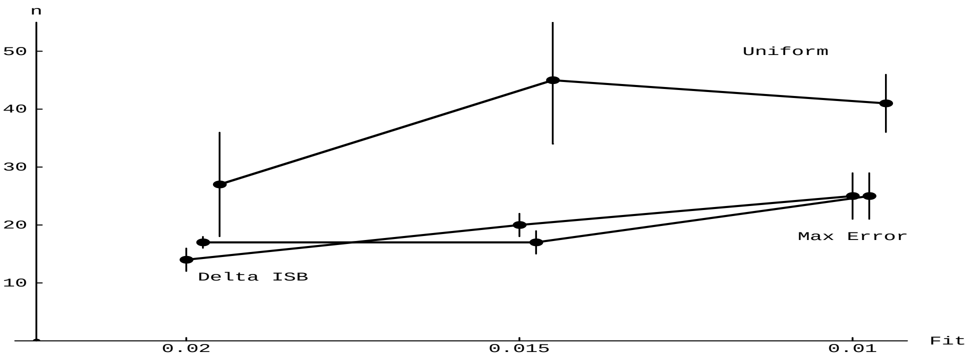


Figure V.14: Number of exemplars selected (n) versus Desired Fit, for ΔISB, Uniform Grid, and Maximum Error. Height of error bars is twice the standard deviation. All methods used the same tolerances. the lines are offset slightly to separate the error bars.

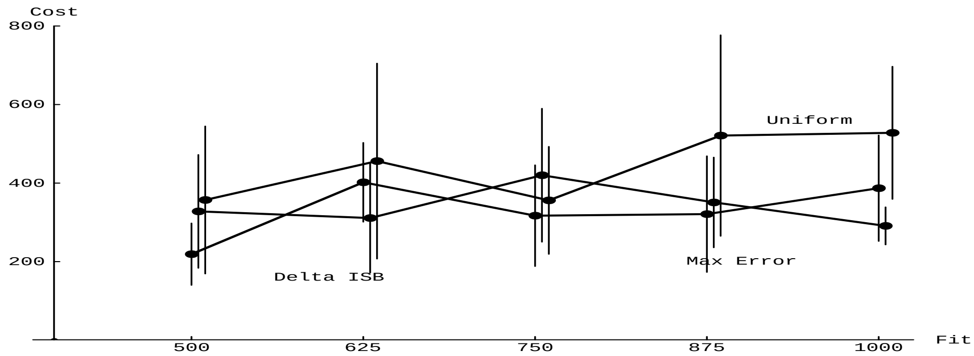


Figure V.15: Cost (millions of multiplies) versus N (candidate set size,) for ΔISB, Uniform Grid, and Maximum Error. Height of error bars is twice the standard deviation. All methods used the same tolerance (0.01 rmse, 0.044 nrmse,) the lines are offset slightly to separate the error bars.

Cost Comparison	
Method	Multiplies
ΔISB	329.2×10^6 (128.5×10^6)
Max Error	340.2×10^6 (210.0×10^6)
Uniform Grid	444.0×10^6 (202.1×10^6)
Random Selection	612.6×10^6 (285.8×10^6)
No Selection	9391.6×10^6 (5366×10^6)

Table V.1: Results are averages over 25 runs with a 2 hidden layer network with 4 inputs and 1 output. Numbers in parentheses are standard deviations. Similar results were obtained for a 1 hidden layer network.

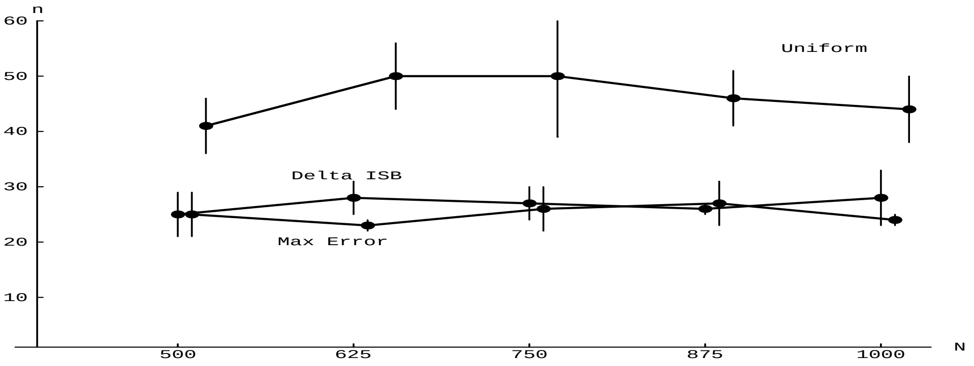


Figure V.16: Number of exemplars selected (n) versus candidate set size (N) for ΔISB, Uniform Grid, and Maximum Error. All methods used the same tolerance (0.01 rmse, 0.044 nrmse,) the lines are offset slightly to separate the error bars. For each N, ΔISB and Maximum Error tend to select 25 exemplars, plus or minus 2.

V.I.1 More Difficult Comparison With Maximum Error

In the results above, *Max Error* and Δ*ISB* were fairly comparable. In order to distinguish between their capabilities, we applied these two selection methods to a more difficult time series prediction task, using the same Mackey-Glass equation as before, but with $\tau = 30$. The purpose of this exercise was to push both techniques until one failed.

We used techniques from nonlinear dynamical systems analysis to embed Data Set 3 into an example space, using Minimum Mutual Information method to determine the lag (delay between inputs) and the method of False Nearest Neighbors to determine the embedding dimension (number of inputs) [1, 58]. We embedded the time series into a 5 dimensional input space, each input lagged by 7 time steps. This gave a set of 5065 examples. We trained on the first 4000 examples of Data Set 3, and tested on the following 1065 examples. We trained the networks to perform Single Step Prediction (SSP), with the goal of predicting a value 7 steps ahead. Based on the results from the experiments on Data Sets 1 and 2, a two hidden layer network was used, initially having 3 units per hidden layer. We combined training with the same straightforward method of determining network complexity as before, growing the network when it could not fit the selected exemplars. This time, a single unit was added to the first hidden layer, until it reached a maximum of 6 hidden units, at which time successive growing resulted in a single unit added to the second hidden layer. This procedure was chosen arbitrarily. As before, after training in feedforward mode, we can then turn the network into a recurrent network by *iterating* its prediction, using its predictions as input to the network. Figures V.I.1 and V.19 show the iterated prediction of a network trained by active exemplar selection to a 0.05 nrmse fit. This network required 91 exemplars out of the 4000 available.

Max Error performed poorly here, selecting exemplars that were (apparently) dif-

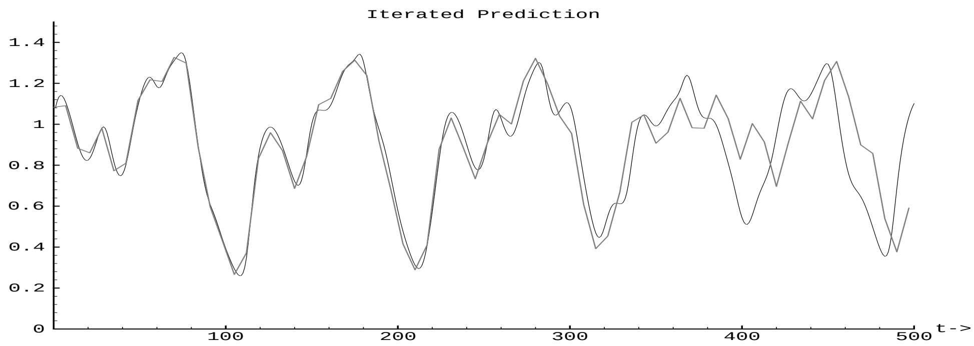


Figure V.17: Iterated Prediction of a typical network trained to 0.05 nrmse (single step prediction error). Here the iterated prediction is over the training set. The solid, smooth line is the target time series, the jagged, broken line is the network output iterated at steps of 7 time units.

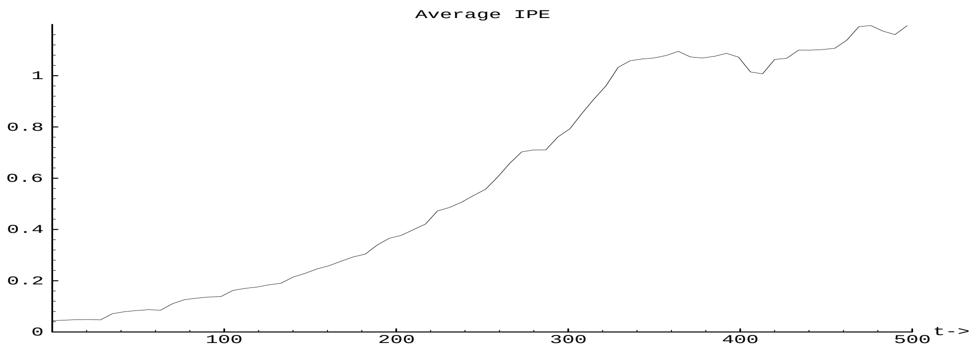


Figure V.18: Average iterated prediction error, averaged over the entire test set for each value of t. This is for the same network as for the previous figure.

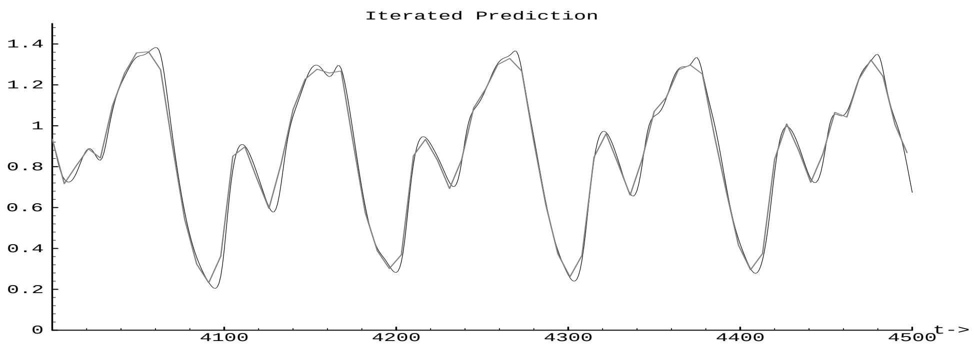


Figure V.19: An example of the iterated prediction given by a typical network trained by ΔISB to 0.05 nrmse (single step prediction error) starting at the beginning of the test set. The solid, smooth line is the target time series, the jagged, broken line is the network output iterated at steps of 7 time units.

ficult to fit, selecting large networks. Over a set of 3 runs, *Max Error* required about 10 extra hidden units, on average, resulting in total cost that was over 5 times what ΔISB required, on average. ΔISB required an average of 97 exemplars out of the 4000 available, whereas *Max Error* selected 118 exemplars, on average over the runs for which it converged. In order to be able to obtain comparable results we relaxed the tolerance to a desired fit of 0.06 nrmse. In this case, *Max Error* again performed well in all categories, requiring from 66 exemplars, on average. From this we may surmise that *Max Error* will work well when the tolerance is sufficiently loose, but can be less reliable when tight tolerances are required. This is because it (apparently) can expend excessive resources focussing on fine detail whereas ΔISB scales the error for a particular exemplar by its importance, in a particular sense, by using a measure of how sensitive the network function is to the presence of the exemplar in the training set.

V.J Number of Exemplars vs. Number of Weights

Recent theoretical results suggest a close relationship between the size of a network and the number of exemplars required for good generalization [9]. If our method of selecting exemplars is "efficient," then the ratio of exemplars to number of weights should be low. Table 4 gives the results of looking at the relationship between the number of exemplars and network complexity for the experiments involving all 4 exemplar selection methods (trained on Data Set 1 and tested on Data Set 2). Table 5 gives the results for the experiments applying ΔISB and *Max Error* to Data Set 3. These numbers give the ratio of the number of exemplars to the number of weights when the network was forced to grow, indicating how many exemplars can be selected before the network becomes "saturated," unable to fit the selected exemplars. This indicates how informative the exemplars are. This ratio tended to be closer to 1.0 for active selection than for uniform grid and random selection.

V.K Conclusions

These results clearly demonstrate that exemplar selection can dramatically lower the cost of training. For this particular learning task, active selection methods were much more efficient than two contending exemplar selection techniques which do not utilize the network state. There was also evidence of improved generalization.

ΔISB and *Maximum Error* consistently selected concise sets of exemplars, with overhead associated with exemplar selection being a small percentage of total training cost. *Maximum Error* is attractive even though we have not justified it analytically, as it performs about as well as ΔISB on the less difficult comparison, but is easier to code and cheaper to compute. When *Max Error* and ΔISB were challenged by a more difficult comparison involving training to a tight tolerance on irregular time series data, *Maximum Error* selected larger networks and more exemplars, hence requiring more in total cost. This suggests that an example with maximal error may not necessarily be the most informative. The gradient information is a means of estimating the importance of such a datum, by scaling the error by the sensitivity of the network function, as averaged along with other available data.

Analysis of the ratio of exemplars to the number of weights showed a close relation between network complexity and the number of exemplars required, with active selection able to support a network with just over 1 exemplar per weight, on average.

Although the results may have limited appeal to practitioners dealing exclusively with small, noisy data sets, they do indicate that judicious selection of training data can hold great promise under ideal circumstances. Hence, they warrant extending these methods to handle noisy data. Also, as several important learning tasks are noiseless the results do hold practical benefit as well (e.g., learning identity maps [32], learning finite state automata from examples of bounded length, learning a time series embedded into an input space such that no two examples have common input but with different output value).

Number of Exemplars		
Method	Number	Portion
ΔISB	25	3.6%
Max Error	25	3.5%
Uniform Grid	47	6.7%
Random Selection	369	49.2%

Table V.2: Results are averaged over 50 runs, selecting exemplars from given sets of 500, 625, 750, 875, and 1000 examples. *ΔISB*, *Max Error*, and *Uniform Grid* selected roughly the same number regardless of how many were available, whereas *Random Selection* selected roughly the same *percentage* of the available exemplars.

Generalization	
Method	Ratio
ΔISB	0.95 (0.018)
Max Error	0.96 (0.020)
Uniform Grid	1.03 (0.028)
No Selection	1.05 (0.015)
Random Selection	1.13 (0.022)

Table V.3: Results are averages over 50 runs for ΔISB, Max Error, and Uniform Grid, 45 runs for Random Selection, and 25 runs for No Selection. "Ratio" gives the ratio of Test Error to Training Error. The number in parentheses is the standard deviation. The training error in each case was 0.04425 nrmse (0.01 rmse) The test set was 500 examples starting from 5000 time steps beyond the beginning of the training set.

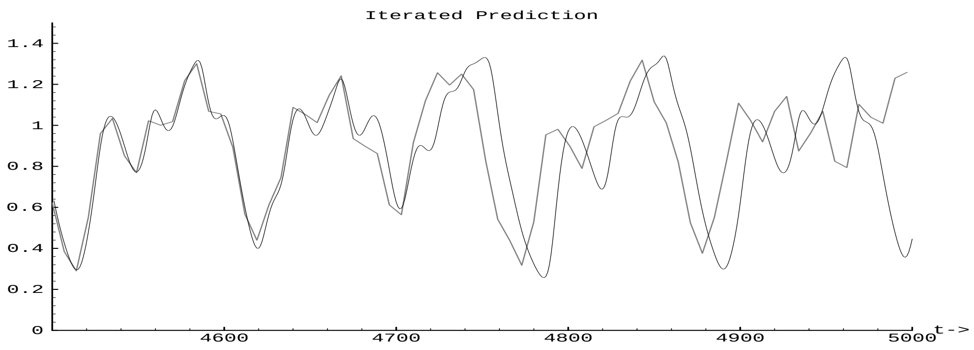


Figure V.20: Another example of the iterated prediction given by a typical network trained by ΔISB to 0.05 nrmse (single step prediction error) starting at 500 steps into the test set. The solid, smooth line is the target time series, the jagged, broken line is the network output iterated at steps of 7 time units.

Exemplars per Weight		
Method	p	n/p
ΔISB	7	0.89 (0.16)
	13	1.18 (0.12)
Max Error	7	0.84 (0.06)
	13	1.13 (0.14)
Uniform Grid	7	1.07 (0.10)
	13	2.01 (0.27)
Random Selection	7	1.39 (0.86)
	13	2.97 (2.78)
ΔISB	19	0.97 (0.12)
Max Error	19	0.96 (0.08)
Uniform Grid	19	1.17 (0.16)
Random Selection	19	2.06 (1.46)

Table V.4: Average number of exemplars per weight when growing occured, averaged over the runs involving Data Sets 1 and 2. Here, p gives the number of network weights at time of growing, and n gives the number of exemplars. The number in parentheses is the standard deviation of the ratio. The values for $p = 7$ and $p = 13$ are from the experiments with the single hidden layer network, and the values for $p = 19$ are from the experiments with the two hidden layer network.

Exemplars per Weight		
Method	p	n/p
ΔISB	34	1.03 (0.09)
	43	1.27 (0.05)
	52	1.04 (0.05)
	Avg	1.11 (0.12)
Max Error	34	1.16 (0.06)
	43	1.24 (0.12)
	52	1.26 (0.11)
	Avg	1.19 (0.12)

Table V.5: Average number of exemplars per weight when growing occured, averaged over 5 runs with Data Set 3, when trained to a desired fit of 0.06 nrmse. Here, p gives the number of network weights at time of growing, and n gives the number of exemplars. The number in parentheses is the standard deviation of the ratio.

Chapter VI

Criterion for Noisy Exemplar Selection

This chapter states theorems demonstrating that cross-validatory methods provide estimates of generalization performance that are statistically accurate and precise. The proofs are in Appendix D.

VI.A Summary

Suppose we are given a fixed network architecture. (We dispense with this assumption later.) Let z^N denote a given set of N training examples. Let $Q_N(z^N)$ denote the expected squared error (the expectation taken over all possible examples) of the network after being trained on z^N. This measures the quality of fit afforded by training on a given set of N examples.

Let $IMSE_N$ denote the Integrated Mean Squared Error for training sets of size N. Given reasonable assumptions, it is straightforward to show that $IMSE_N = E[Q_N(Z^N)] - \sigma^2$, where the expectation is now over all training sets of size N, Z^N is a random training set of size N, and σ^2 is the noise variance.

Let $C_N = C_N(z^N)$ denote the "delete-one cross-validation" squared error measure for a network trained on z^N. C_N is obtained by training networks on each of the N training sets of size $N-1$ obtained by deleting a single example; the measure follows by computing squared error for the corresponding deleted example and averaging the results. Let $G_{N,M} = G_{N,M}(z^N, \tilde{z}^M)$ denote the "generalization" measure obtained by separating the available data of size $N+M$ into a training set z^N of size N, and a validation ("test") set $\tilde{z}^M$ of size M; the measure follows by training on z^N and computing averaged squared error over $\tilde{z}^M$.

We show that C_N is an unbiased estimator of $E[Q_{N-1}(Z^N)]$, and hence, estimates $IMSE_{N-1}$ up to noise variance. Similarly, $G_{N,M}$ is an unbiased estimator of $E[Q_N(Z^N, \tilde{Z}^M)]$. $G_{N,M}$ is an unbiased estimator of $E[Q_N(Z^N)]$. Given reasonable conditions on the estimator and on the data generating process we demonstrate convergence with probability 1 of $G_{N,M}$ and C_N to $E[Q_N(Z^N)]$ as N and M grow large.

A direct consequence of these results is that when choice is restricted to a set of network architectures whose complexity is bounded above a priori, then choosing the architecture for which either C_N (or $G_{N,M}$) is minimized leads to choice of the network for which $IMSE_N$ is nearly minimized for all N (respectively, N, M) sufficiently large.

We also provide results for training sets sampled at particular inputs. Conditional $IMSE$ is an appealing criterion for evaluating a particular choice of training set in the presence of noise. These results demonstrate that delete-one cross-validation estimates average MSE (the average taken over the given set of inputs,) and that hold-out set cross-validation gives an unbiased estimate of $E[Q_N(Z^N)|Z^N = (x^N, Y^N)]$, given a set of N input values x^N for which corresponding (random) output values Y^N are obtained. Either cross-validation measure can therefore be used to select a representative subset of the entire dataset that can be used for data compression, or for more efficient training (as training can be faster on smaller datasets) [86, 87].

VI.B Unbiasedness of Cross-Validation

We now consider the relation between the cross-validation measure and IMSE. We examine delete-one cross-validation first.

Proposition 1 (Unbiasedness of C_N) *Let Assumptions 1 through 5 hold. Then for given N, C_N is an unbiased estimator of $IMSE_{N-1}(p) + \sigma^2$:*

$$E\left[C_N(Z^N; p)\right] = IMSE_{N-1}(p) + \sigma^2. \tag{VI.1}$$

With hold-out set cross-validation, one obtains an assessment of potential network performance independent of the training set.

Proposition 2 (Unbiasedness of $G_{N,M}$) *Let Assumptions 1 through 5 hold. Let $\tilde{Z}^M \equiv (Z_{N+1}, \ldots, Z_{N+M})$. Then for given N and M, $G_{N,M}$ is an unbiased estimator of $IMSE_N(p) + \sigma^2$:*

$$E\left[G_{N,M}(Z^N, \tilde{Z}^M; p)\right] = IMSE_N(p) + \sigma^2. \tag{VI.2}$$

The latter approach is appealing for large M, N. We expect delete-one cross-validation to be more appealing than hold-out set cross-validation, when training data are not abundant.

VI.C Unbiasedness when Sampling at Selected Inputs

We obtain analogous results for training sets obtained by sampling at a given set of inputs x^N. We first consider the result for delete-one cross-validation.

Proposition 3 (Expectation of C_N given x^N) *Let Assumptions 1 through 5 hold. Then, given a particular set of inputs, x^N, C_N is an unbiased estimator of average $MSE_{N-1} + \sigma^2$, the average taken conditional on x^N:*

$$E\left[C_N(Z^N, p)\Big| X^N = x^N\right] \;=\; \frac{1}{N}\sum_{i=1}^{N} M_{N-1}(x_i | x^N_{-i}; p) \;+\; \sigma^2,$$

where x^N_{-i} is a matrix made by deleting the ith row of x^N.

This essentially gives an estimate of MSE_{N-1} limited to $x \in x^N$, losing a degree of freedom while providing no estimate of the MSE away from the training points. For this average to converge to $IMSE_{N-1}$, it will suffice for the empirical distribution of x^N, $\bar{\mu}_N$, to converge to μ, i.e., $\bar{\mu}_N \Rightarrow \mu$.

We obtain a stronger result for hold-out set cross-validation. The hold-out set gives independent information on MSE_N, resulting in an estimate of $IMSE_N$ for given x^N.

Proposition 4 (Expectation of $G_{N,M}$ given x^N) *Let Assumptions 1 through 5 hold. Let $\tilde{Z}^M \equiv (Z_{N+1}, \ldots, Z_{N+M})'$. Then, given a particular set of inputs, x^N, $G_{N,M}$ is an unbiased estimator of of $IMSE_N(x^N; p) + \sigma^2$:*

$$E\left[G_{N,M}(Z^N, \tilde{Z}^M; p) \middle| X^N = x^N \right] \;\; = \;\; IMSE_N(x^N; p) \; + \sigma^2.$$

VI.D Strong Convergence: Hold-Out Set

Our conditions deliver not only unbiasedness, but also convergence of hold-out set cross-validation to $IMSE_N$, with probability 1.

Theorem 1 (Convergence of Hold-Out Set w.p. 1) *Let Assumptions 1 through 5 hold. Also let $\tilde{Z}^M \equiv (Z_{N+1}, \ldots, Z_{N+M})$. If for some $A > 0$ a sequence $\{M_N\}$ of natural numbers satisfies $M_N > AN$ for all $N = 1, 2, \ldots$, then*

$$G_{N,M_N}(Z^N, \tilde{Z}^{M_N}; p) \; - \; E[Q_N(Z^N; p)] \;\; \xrightarrow{a.s.} \;\; 0, \textit{as } N \to \infty.$$

VI.E Strong Convergence: Delete-One

Given an additional condition (uniqueness of optimal weights) we can show strong convergence for delete-one cross-validation.

First we establish uniform convergence of the estimators $\hat{w}^p(Z^N_{-i})$ to optimal weights (uniformly over $1 \leq i \leq N$.) The following lemma is useful for this purpose.

Lemma 2 *Let Θ be a compact set. Suppose that $\{h_i : \Theta \to \Re\}$ is a sequence of non-stochastic functions such that*

$$\sup_{\theta \in \Theta} \left| N^{-1} \sum_{i=1}^{N} h_i(\theta) \right| \to 0 \textit{ as } N \to \infty.$$

Then

$$\max_{k \in (1,\ldots,N)} \max_{\theta \in \Theta} \left| (N-1)^{-1} \sum_{1 \leq i \leq N, i \neq k} h_i(\theta) \right| \to 0 \textit{ as } N \to \infty.$$

Using this lemma, we can show the next theorem about convergence of $l_{N-1}(Z^N_{-i}; p)$ uniformly in i.

Theorem 2 *Let Assumptions 1 through 5 hold. Let Z_{-k}^N be the dataset made by deleting the kth observation from Z^N. Then*

$$\max_{1\leq i\leq N} d\left(l_{N-1}(Z_{-i}^N;p), \mathbf{W}^{p*}\right) \to 0 \ a.s.\text{-}\mathcal{P} \ as \ N \to \infty, \tag{VI.3}$$

where $d\left(w, \mathbf{W}^{p*}\right) \equiv \inf_{w^*\in\mathbf{W}^{p*}} \|w - w^*\|$.

This convergence result leads to the next result that the delete-one cross validation measure does not under-estimate the optimized MSE, namely, $\inf_{w\in\mathbf{W}^p} \lambda(w;p)$.

Theorem 3 *Let Assumptions 1 through 5 hold. Define*

$$C_N(z^N;p) \equiv N^{-1}\sum_{i=1}^{N}\left(y_i - f^p\left(x_i, l_{N-1}\left(z_{-i}^N;p\right)\right)\right)^2.$$

Then

$$\liminf_{N\to\infty} C_N(Z^N;p) \geq \min_{w\in\mathbf{W}^p} \lambda(w;p) \ a.s.\text{-}\mathcal{P}.$$

When the optimum weight w^* is unique, we have a stronger result about convergence of the delete-one cross validation measure.

Assumption 6 $\mathbf{W}^{p*}$ *is a singleton, i.e.,* $\mathbf{W}^{p*}$ *has only one element.*

Theorem 4 *Let Assumptions 1 through 6 hold. Then*

$$C_N\left(Z^N;p\right) - E\left[Q_N(Z^N;p)\right] \to 0 \ a.s. \ as \ N \to \infty.$$

VI.F Conclusion

Our results justify the intuition that cross-validation measures unbiasedly and consistently estimate the expected squared error of networks trained on finite training sets, therefore providing means of obtaining $IMSE$-approximate methods of selecting appropriate network architectures, or for evaluating a particular choice of training set.

Use of these cross-validation measures therefore permits us to avoid underfitting the data, asymptotically. Note, however, that although we also thereby avoid overfitting asymptotically, this avoidance is not necessarily accomplished by choosing a minimally complex architecture. The possibility exists that $IMSE_{N-1}(p) = IMSE_{N-1}(p+1)$. Because our cross-validated estimates of these quantities are random we may by chance observe $C_N(Z^N;p) > C_N(Z^N;p+1)$ and therefore select the more complex network, even though the less complex network is equally good. Of course, because the IMSE's are the same, no performance degradation (overfitting) will result in this solution.

Chapter VII

Techniques for Selecting and Pruning Noisy Exemplars

This chapter presents preliminary results from work in progress. In preliminary experiments, we determined that although the ΔISB method seemed to be more robust in the presence of noise than the maximum error criterion, it is still prone to selecting outliers when the data is very noisy. To correct for this, we modify the ΔISB method (developed for use on clean data) for use on noisy data. The result is an exemplar selection method that selects a *set* of exemplars. This selected set is likely to contain useful exemplars. However, some exemplars will prove more useful than others. For example, the selected set may contain outliers. We counteract this by pruning away the less useful exemplars. The pruning algorithm optimizes a cross-validation measure, so we refer to it as ΔCV.

We evaluate the behavior of ΔCV on a toy problem. The test results indicate that active exemplar selection can make training on noisy data more robust, because the network trained using ΔCV was able to learn a mapping that could not be learned after numerous trials using all the available examples. The results from this toy application suggest that further theoretical and empirical development of the ΔCV method is warranted.

VII.A Performance of ΔISB on Noisy Data

We can gain insight into the behavior of the ΔISB on noisy data by analyzing the ΔISB criterion.

In the presence of noise, $g(x_i)$ is no longer available for particular input x_i. Rather, we are provided with a noisy target $y_i = g(x_i) + \varepsilon_i$. Therefore, the ΔISB criterion itself is inappropriate for use on noisy data. By modifying Equation IV.8, we obtain an analogous (albeit, ad hoc) criterion that is well-defined for use on finite, noisy training sets. To do so, substitute y_i for $g(x_i)$, $i = 1, 2, \ldots, N$. Next, replace the integral (evaluated over all possible examples) with the hold-out set cross validation measure (averaged over the set of all available examples, excluding the current exemplar set). For the moment, we restrict attention to the case where the Hessian matrix can be replaced with the identity matrix (we found this to be an acceptable approximation in practice - see Chapter V for additional details). We now define the $\Delta\overline{ISB}$ criterion for a particular candidate exemplar

$z_i \in z^N, 1 \leq i \leq N$, given a network with p weights obtained by training on z^n, as

$$\Delta\overline{ISB}(z_i|z^n, z^N) \equiv (y_i - f^p(x_i, l_n(z^n;p)))'\nabla_w f^p(x_i, l_n(z^n;p))'$$
$$(N-n)^{-1} \sum_{j=n+1}^{N} \nabla_w f^p(x_j, l_n(z^n;p))(g(x_j) - f^p(x_j, l_n(z^n;p))). \quad \text{(VII.1)}$$

The average is over elements of z^N not in z^n, that is, $z^N \backslash z^n$ (here, "$\backslash$" denotes set difference). Heuristically, we recognize the average over $z^N \backslash z^n$ as the gradient (with respect to network weights) of the hold-out set cross-validation measure. The term in front of the average is the gradient of the squared error for the network function evaluated at candidate example z_i. This suggests that $\Delta\overline{ISB}$ may have heuristic appeal similar to that of ΔISB. However, analysis of the criterion shows that it is adversely susceptable to selecting outliers.

To illustrate, consider a situation where the algorithm is selecting from among several examples for which $\Delta\overline{ISB} > 0$, where the examples under consideration all have the same input. In this case, the $\Delta\overline{ISB}$ criterion will tend to prefer "outlying" targets. To understand why, suppose $\Delta\overline{ISB}$ is evaluating several different candidate examples, $z_{i_1}, z_{i_2}, z_{i_3}, z_{i_4}$, such that $z_{i_j} = (x_i, y_{i_j})$, $j = 1,2,3,4$, for some i, $1 \leq i \leq N$. Furthermore, let $y_{i_j} - f^p(x_i, l_n(z^n;p)) > 0$, $j = 1,2,3,4$, and $0 < y_{i_1} < y_{i_2} < y_{i_3} < y_{i_4}$. In words, we have four candidate examples, with identical inputs, but with distinct positive targets, sorted in ascending order. Furthermore, let the error between the network output and each of the four targets be positive, and $\Delta\overline{ISB}(z_{i_j}|z^n, z^N) > 0$, $j = 1,2,3,4$. Note that in this situation, $\Delta\overline{ISB}$ is maximized by $\max\{y_{i_j}, j = 1,2,3,4\} = y_{i_4}$. Therefore, $\Delta\overline{ISB}$ is maximized in this case by the most outlying target. Situations similar to this example occured frequently in experiments.

In a certain sense, the criterion is overconfident in its estimate of the benefit of a candidate example (in terms of its ability to lower the cross validation measure) because it is prone to "overshooting the mark." This is because it is prone to underestimating the perturbation to the average squared error due to selecting the exemplar.[1]

In our example, if y_{i_4} is closer to $g(x_i)$ than y_{i_1}, an omniscient algorithm might prefer z_{i_4} over z_{i_1}, and so selecting the exemplar with greater error is warranted. On the other hand, if $y_{i_4} \gg g(x_i)$, and if the network is flexible enough, selecting y_{i_4} could result in the network function overshooting g at x_i. This will typically result in a nonlocal effect on the network function, with the network function also becoming out of fit elsewhere in the vicinity of x_i, due to the continuity of the network function. We have corroborated these behaviors in experiments.

On the other hand, experiments did confirm that the location of the exemplar in input space was typically well-chosen. Therefore, in the next section we discuss an ad hoc variant of this method that selects a set of exemplars nearby this location in input space, pruning away outliers.

[1] Intuitively, the effect a particular example has upon the average squared error is estimated indirectly, by predicting the effect of the example upon the weight vector. See Equation (IV.4), which represents an estimate of the perturbation of the weight vector caused by appending a particular exemplar to the current set of exemplars.

VII.B Modifying ΔISB for Noisy Data

We experimented with an ad hoc extension of the $\Delta\overline{ISB}$ criterion which seems to overcome the limitations mentioned above, while retaining the advantages. This new criterion selects *sets* of exemplars. Rather than selecting the single exemplar maximizing $\Delta\overline{ISB}$ (let this exemplar be $\check{z}$), it selects a number of exemplars in the vicinity of $\check{z}$, for which $\Delta\overline{ISB}$ is positive, not including $\check{z}$. We reject $\check{z}$ itself, to conserve computation, because it was typically pruned away from the exemplar set in a later step. (The reasons for this will become more clear below when we discuss the pruning algorithm. Briefly, this is because while $\check{z}$ tended to be at a good location in input space, its target value was typically far away from g, and was typically pruned from the exemplar set.)

Rather than select a fixed number of exemplars at each iteration, we tie the size of the new set to the size of the network, by selecting a number of exemplars equal to a percentage of the number of weights, p, rounding down to the nearest integer. Therefore, more exemplars will be selected for a large network than for a small network.

As this criterion is intended to estimate the decrement in cross-validation measure, we refer to it as ΔCV.

VII.C Active Pruning

Note that ΔCV does not select the single exemplar maximizing $\Delta\overline{ISB}$, selecting nearby exemplars instead. This modification causes the method to select exemplars that are nearby in input space, but with targets that are typically closer to g in output space than either the current network output or the target value for the current $\Delta\overline{ISB}$ selection. However, the new set of exemplars may still contain exemplars that are far away from g. We corrected for this by pruning away exemplars after every iteration. In our test application, this pruning procedure tended to do a good job of excising exemplars that were far away from g, even though g is not available to the algorithm.

It turned out that we were able to calculate the pruning criterion using by-products of network training. During training on z^n, we use delete-1 cross validation to select the appropriate size network for the given set of exemplars (because we are training on noisy examples, the growing method used with the ΔISB method is inappropriate). We evaluate the hold-out set cross validation measure of the network function for each deleted subset (where these network functions are given by $f^p(\cdot, l_{n-1}(z^n_{-i}; p))$, $i = 1, 2, \ldots, n$), and compare it with the hold-out set cross validation error for the network trained on all the exemplars (given by $f^p(\cdot, l_n(z^n; p))$). We then sort the exemplars in order of how much the hold-out set cross validation measure decreases when the exemplar is "pruned."

Note that a set of exemplars is selected at each iteration, a large portion of which may be subsequently pruned. Pruning a single exemplar at a time could result in the number of exemplars growing unnecessarily large. Therefore, we prune several exemplars at each iteration. The number of exemplars that is pruned is adaptively regulated at each iteration. We begin by pruning a number equal to a percentage of the current number of exemplars. (This implies that more exemplars will be pruned from larger exemplar sets.) After pruning, the network is fine-tuned on the new set, if the number of exemplars pruned exceeds one (if only one exemplar is pruned, then no fine-tuning is necessary, because the fine-tuned weights for this exemplar set are available from the delete-1 cross-validation procedure that is used during training). After fine-tuning, the hold-out set cross-validation measure is evaluated over the candidate set. If it has increased, then we recover the original weights, and then prune one fewer exemplar. We repeat this as necessary.

VII.D The ΔCV Algorithm

We now present the algorithm that we used in our preliminary test application. The algorithm requires 3 user-set parameters, C_1, C_2, and C_3. C_1 regulates the size of the initial set of exemplars (as a multiple of the number of weights in the smallest network). C_2 regulates the number of exemplars that are pruned at each iteration (as a percentage of the number of exemplars). C_3 regulates the number of exemplars that are selected each iteration (as a percentage of the number of weights). We intend the algorithm presented to serve only as a description of the procedure we followed in the experiments, as we would expect a general purpose algorithm to automate the setting of these parameters. In our experiments we set $C_1 = 5$, $C_2 = 0.05$, and $C_3 = 0.25$. Here is (the preliminary version of) the ΔCV algorithm:

1. **Initialize Exemplar Set**

   Initialize the exemplar set by selecting $C_1 p_{min}$ exemplars from the set of candidates, uniformly at random without replacement, where p_{min} gives the number of weights in the smallest network in the range of networks being considered. Set $n = C_1 p_{min}$.

2. **Train**

   Train each network on the current exemplar set. Let p_{cv} give the index of the network minimizing the delete-1 cross-validation measure over the exemplar set. Let $\mathcal{E}_1$ be the hold-out set cross-validation measure for network p_{cv}, averaged over the candidate examples that are not yet in the exemplar set.

3. **Prune**

   For network p_{cv}, for each exemplar z_i, $1 \leq i \leq n$, calculate the decrement in hold-out set cross-validation measure (averaged over the remaining candidate examples) that results when each exemplar is deleted from the exemplar set. Sort the exemplars according to this measure, in descending order. Let $D = C_2 n$. Perform the following steps:

   (a) Delete the first D exemplars from the exemplar set.

   (b) If $D = 1$, retrieve the weights for network p_{cv} corresponding to the deleted exemplar, obtained in Step 2). If $D > 1$, retrieve network p_{cv}, and fine-tune it on the new set of exemplars.

   (c) Let $\mathcal{E}_2$ be the hold-out set cross-validation measure for the current network, averaged over the remaining candidate examples.

   (d) Decrement D. If $\mathcal{E}_1 \leq \mathcal{E}_2$ and $D > 0$, return to Step 3a.
   If $\mathcal{E}_1 \leq \mathcal{E}_2$ and $D = 0$, retrieve network p_{cv}.

   Let $n = n - D$.

4. **Evaluate Progress and Quit, if Done**

   Evaluate the hold-out set cross validation measure for the current network. If it is within desired tolerance, exit.

5. **Select New Exemplars and Continue**

   Using the current network, evaluate ΔCV over z^N, and select $C_3 p$ examples, where p is the number of weights in the current network. Set $n = n + C_3 p$. Return to Step 2.

For the sake of obtaining preliminary results quickly, we made two concessions to exact simulation of the above algorithm. Instead of obtaining $l_{n-1}(z_{-i}^{n-1}; p)$ by initializing the weights to small random values, we start from $l_n(z^n; p)$, the weights obtained by training on all n exemplars. This approximation tends to work well when the ratio of the number of exemplars to the number of weights is greater than 1, and will typically reduce the computation required by an order of magnitude, however, it can also bias the delete-1 cross validation procedure under certain circumstances.

We also continued using weights from the previous iteration when training on a new set of exemplars, which may have introduced a similar bias. As the reader may be able to discern from the ensuing figures, these two concessions may have clouded our initial results.

VII.E Preliminary Results

The ΔCV algorithm presented above has some apparent drawbacks. However, it exhibited some interesting and satisfying behavior on a test problem. We will present the test results, then conclude this chapter with a critique of the algorithm.

Figure VII.1 illustrates the target function and a set of 1000 examples generated from the target function after adding gaussian noise (with zero mean and 0.10 standard deviation).

We investigated how well ΔISB could do on the clean data to obtain an indication of the smallest exemplar set possible by active selection. ΔISB chose a network with 5 hidden units, fitting 1000 clean examples to within 0.01 rmse, selecting 22 exemplars. (We repeated this experiment, with similar results.)

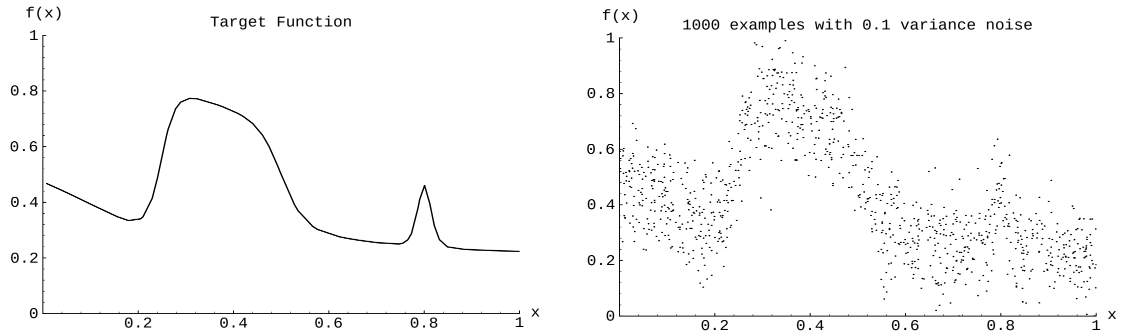


Figure VII.1: The target (left) and 1000 noisy examples (right). The noisy examples are obtained from the target by adding noise from a gaussian distribution with 0.1 standard deviation.

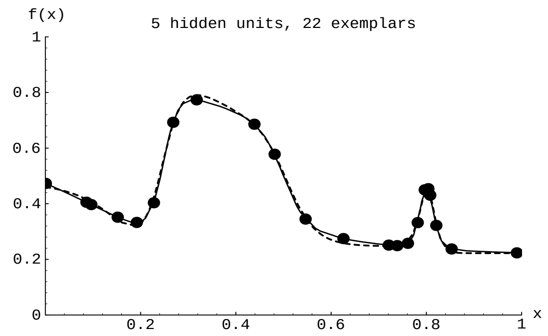


Figure VII.2: Result of applying ΔISB exemplar selection to a set of 1000 clean examples. The method required 22 exemplars.

Figures VII.3-VII.9 show the progress of the ΔCV algorithm over 21 iterations. n gives the number of exemplars (large dots), D gives the number of pruned exemplars (diamonds), and S gives the number of selected exemplars (stars). The figures in the left hand column show the network fit subsequent to Step 2. The diamond markers indicate which exemplars are pruned. The right hand figures show the result of fine-tuning the network in the left column upon the pruned exemplar set, upon entering Step 5. The stars indicate the newly selected exemplars.

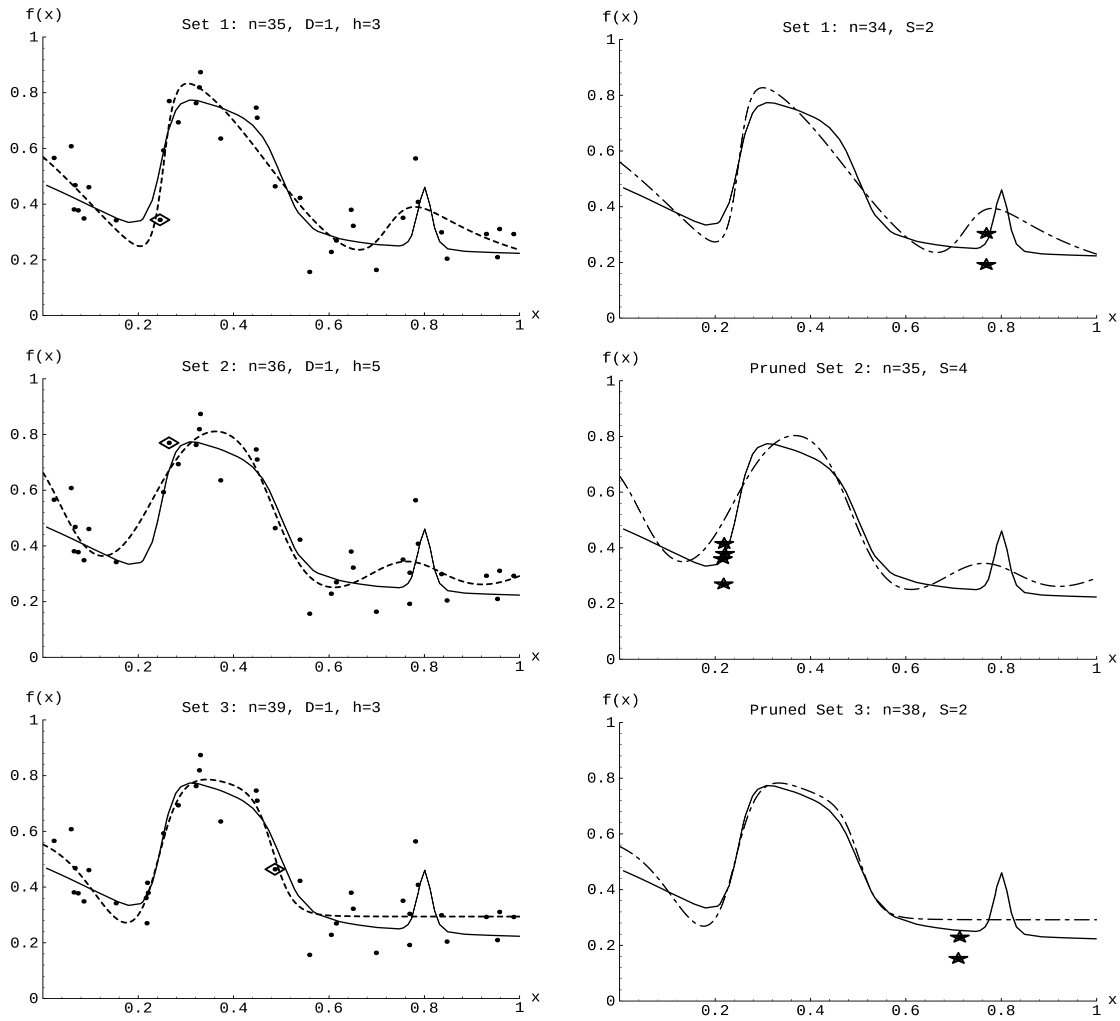


Figure VII.3: These (and plots on following pages) illustrate learning using exemplar selection from noisy examples. In this and subsequent figures, the solid line gives the target $g(x)$. The dashed line gives the network function. n gives the number of exemplars (large dots), D gives the number of exemplars that are to be pruned (diamond markers). h gives the number of hidden units (this is the same for both columns). The network function in the right hand column is obtained by fine-tuning the network in the left hand column on the pruned exemplar set. S gives the number of selected exemplars (stars).

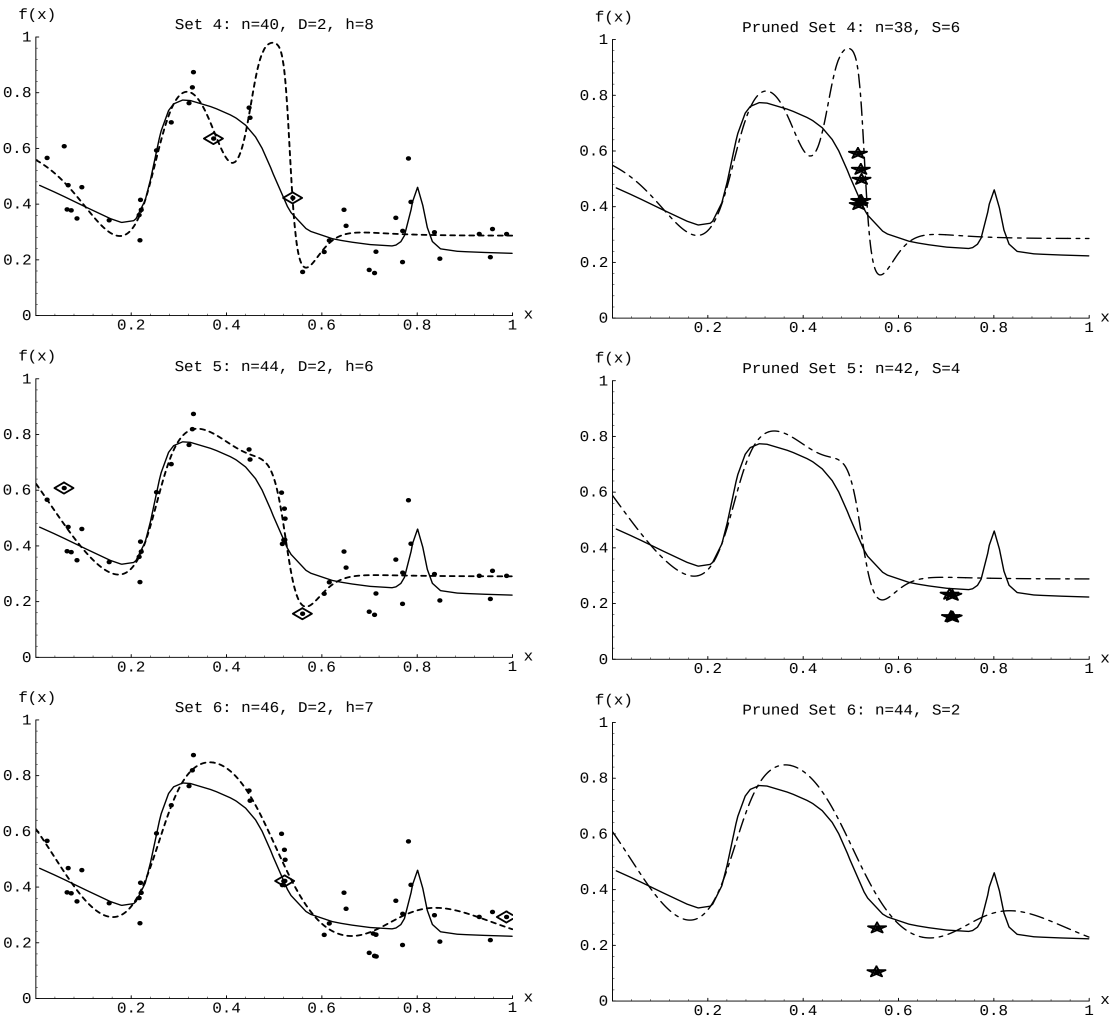


Figure VII.4: Iterations 4, 5, and 6.

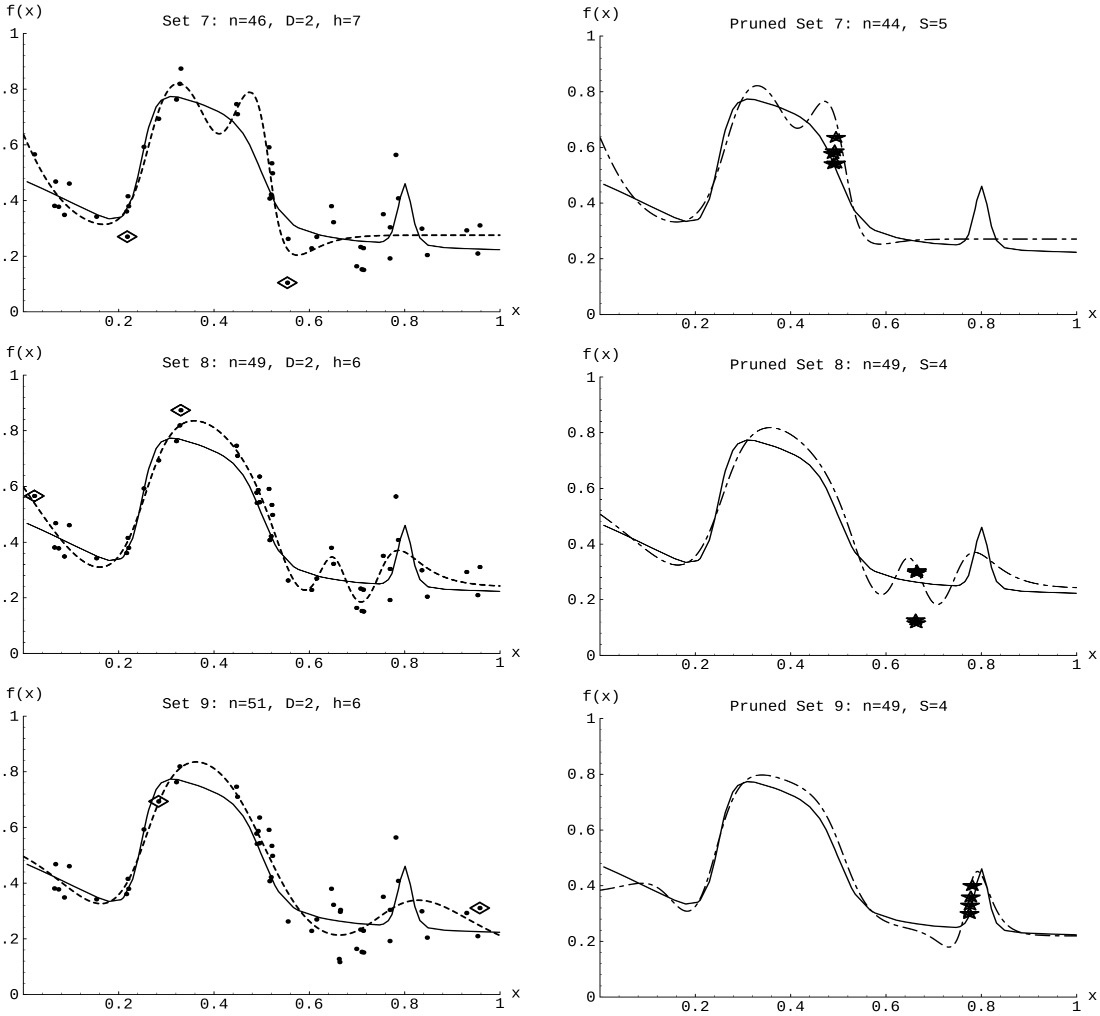


Figure VII.5: Iterations 7, 8, and 9.

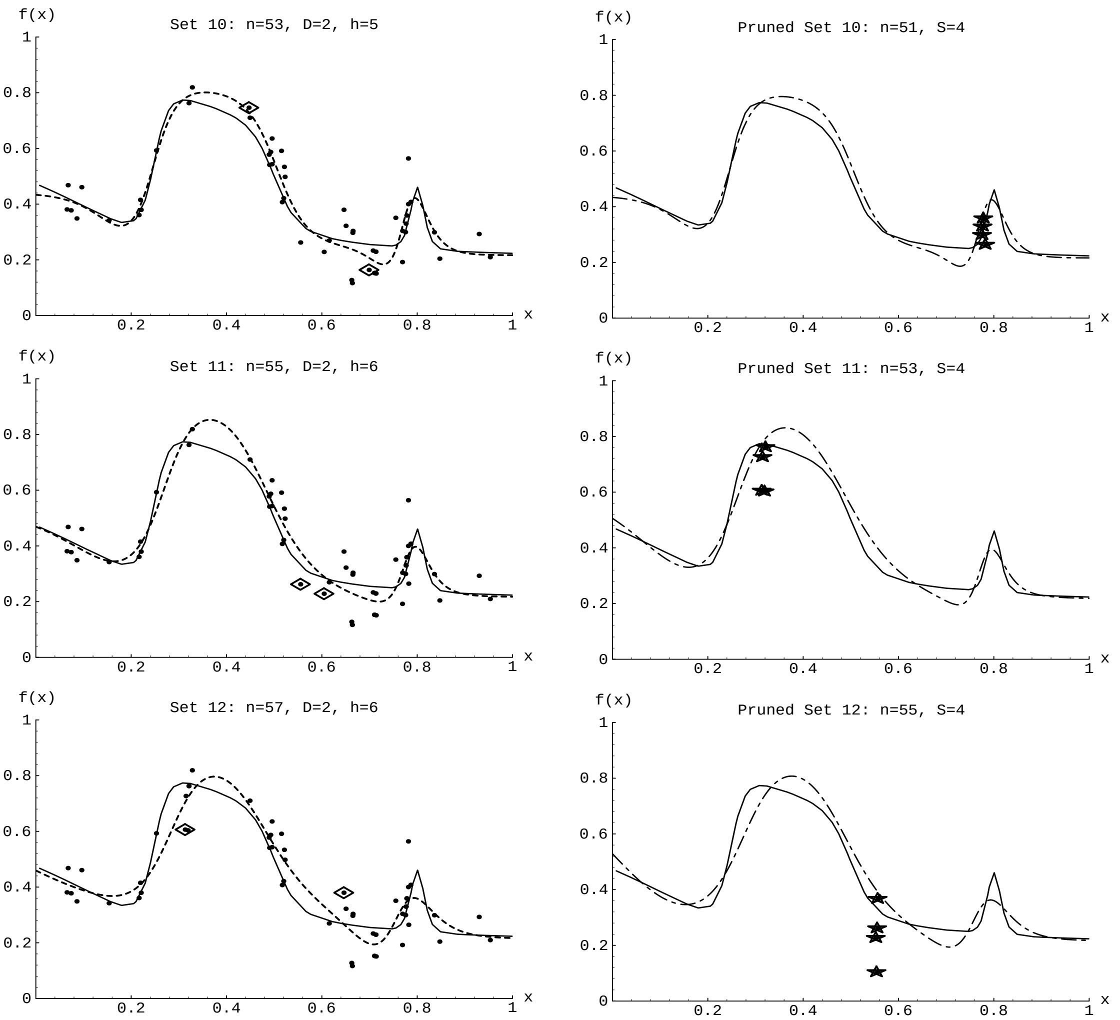


Figure VII.6: Iterations 10, 11, and 12.

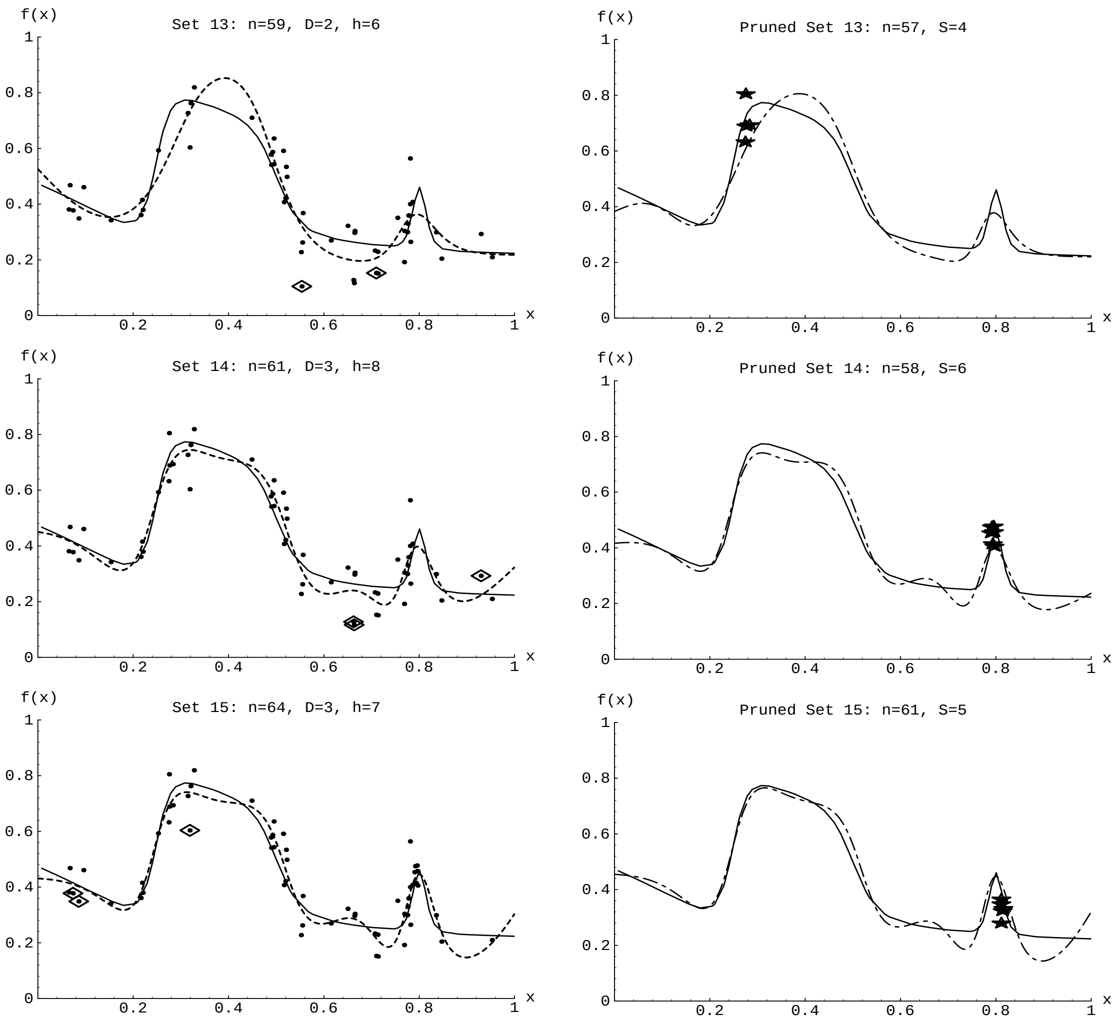


Figure VII.7: Iterations 13, 14, and 15.

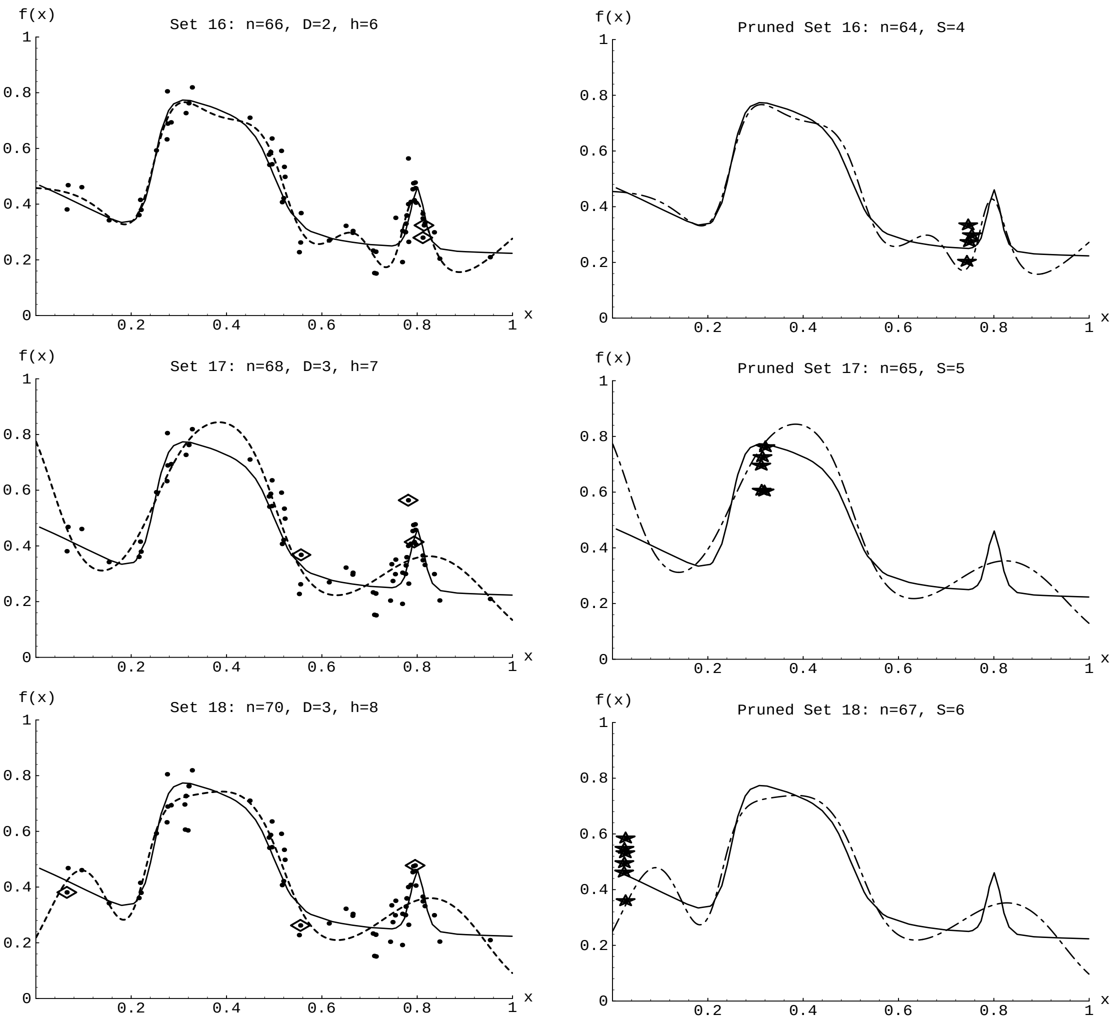


Figure VII.8: Iterations 16, 17, and 18.

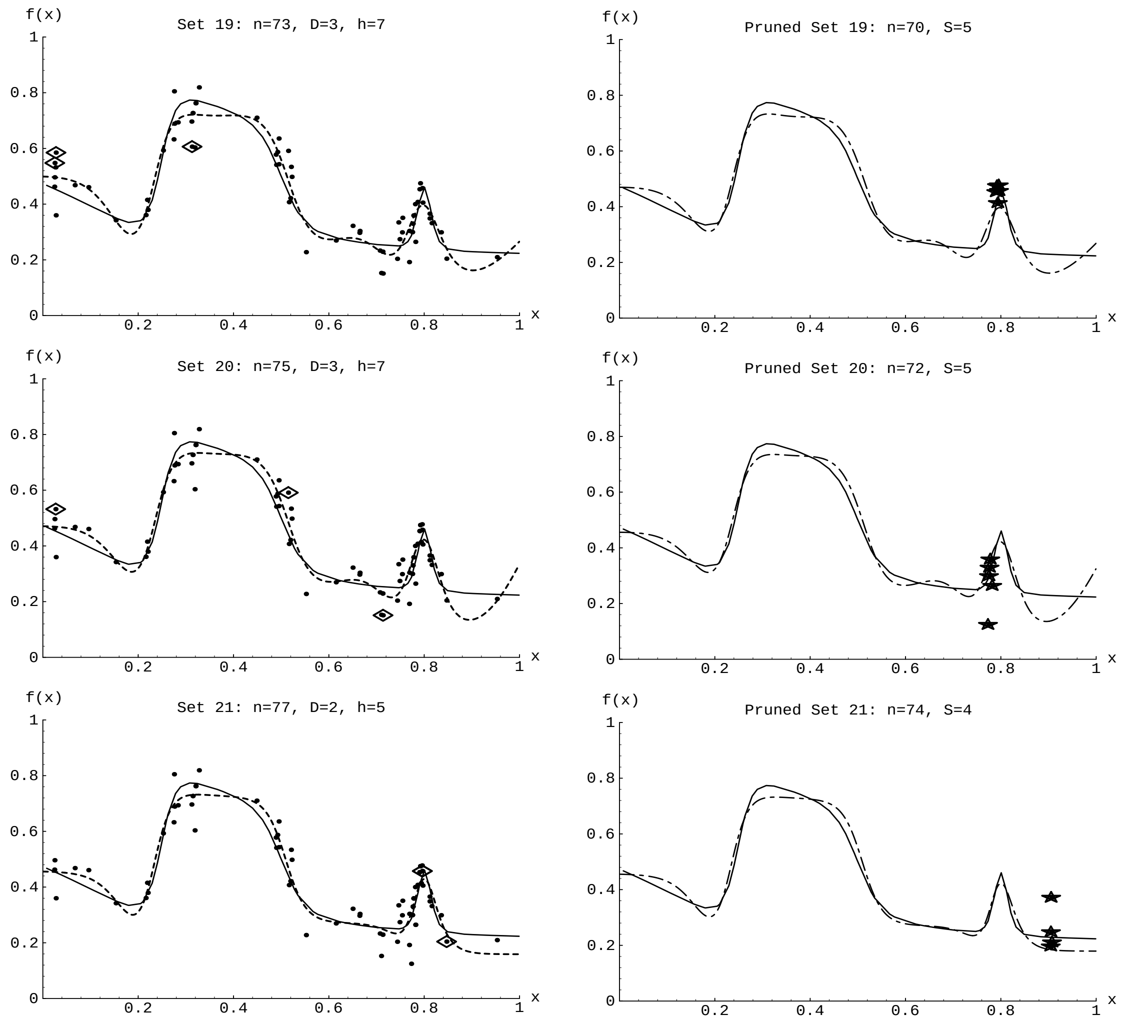


Figure VII.9: Iterations 19, 20, and 21. The best generalization of networks in this sequence is given by the one for Pruned Set 21. (Therefore, the exemplars selected in iteration 21 are not actually added to the exemplar set.)

VII.F Comparison with using all the examples

We ran several baseline experiments for comparison. Above we ran the ΔCV algorithm using the set of 1000 noisy examples as the set of candidate examples. Error over the 1000 examples for the best network obtained by ΔCV was 0.1034 rmse. Here, we compare this with training upon all of the examples.

Because we have the network that generated the target function g, we can closely estimate the amount of error due to bias. The averaged squared error over this test set for the network used to generate the target function is 0.1009 rmse, which is quite close to the expected root mean squared error of 0.1000 (given by the square root of the noise variance). Therefore, the error of the ΔCV network due solely to bias is approximately 0.0025 rmse.

More importantly, access to g allows us to report the actual approximation error (i.e., generalization error), given by the squared error difference between the network function and g. The generalization error for the best ΔCV network was 0.0274 rmse. We will compare the results for the following experiments with this number.

For the first comparison, we ran an experiment using the correct size network, having 5 hidden units. After 10 restarts, the best training error was 0.1041 rmse. The generalization error was 0.0309 rmse. This network is illustrated in Figure VII.10.

Secondly, we ran an experiment using networks ranging in size from 1 hidden unit up to 6 hidden units, restarting each network 3 times. The best fit was given by a network with 5 hidden units, with training error of 0.1041 rmse. The network function is similar to the one obtained in the first comparison, giving a generalization error of 0.0313 rmse.

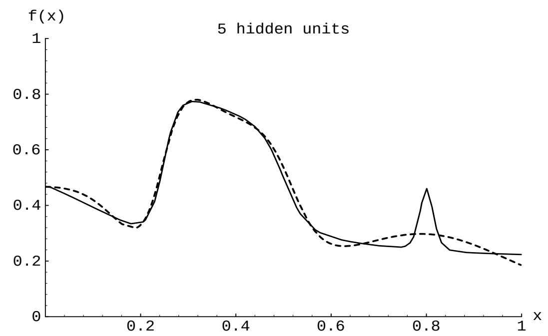


Figure VII.10: This is the best network obtained after training a 5 hidden unit network 10 times on the set of 1000 noisy examples.

At this point, we surmised that there might be insufficient information in the training set of 1000 examples to allow the learning algorithm to locate the small peak. To test this conjecture, we ran a third experiment training networks ranging from 1 hidden unit up to 8 hidden units on 5000 noisy examples. This yielded similar results, with the best network (with 6 hidden units) yielding training error of 0.1041 rmse on the training set, and an error 0.1049 rmse over the set of 1000 noisy examples. The generalization error was 0.0309 rmse. The plot for this fit is similar to the figure shown above for the 5 hidden unit network.

At this point we investigated whether the learning algorithm could learn the underlying function if noise was not present. We trained a network on 1000 clean examples, using the growing method employed in the ΔISB method, with a desired tolerance of 0.01 rmse, and starting with 5 hidden units. It was able to fit the data with the 5 hidden unit network, with an error of 0.0099 rmse over the training set (the plot for this fit is effectively indistinguishable from the plot for the target function). Therefore, the learning algorithm is capable of fitting the target function given a correctly sized network, when provided with clean training examples.

We trained networks ranging in size from 6 hidden units up to 10 hidden units, restarting each one 5 times. The best network had 7 hidden units, giving a training error of 0.1037 rmse. The generalization error was 0.030 rmse. However, it did not locate the small peak. It instead minimized squared error by optimizing the fit on the large skewed hump. This network is shown in Figure VII.11.

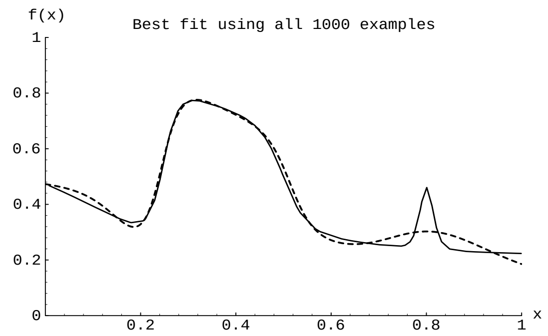


Figure VII.11: This is the best network obtained after training networks ranging in size from 6 to 10 hidden units on the set of 1000 noisy examples, with 5 restarts. This network has 7 hidden units.

VII.G Using ΔCV to Initialize Weights

Although the network fit obtained by ΔCV is better than that obtained by any of the fits obtained by training on all 1000 examples, and clearly is doing a better job at locating the small peak in the underlying target function, the ΔCV algorithm did not appear to be converging on the target function over the sequence of 21 iterations. This could be due to several factors involved in how we implement the method. For instance, a more rigorous pruning could be performed before selecting the next exemplar set. Also, we might employ more restarts during the delete-1 cross validation procedure used to train the networks. Also, in deference to obtaining rough results quickly, we programmed training to continue from weights obtained for the networks obtained from the previous iteration when beginning training on a new exemplar set, rather than initializing weights to small random values at each iteration.

Before continuing to investigate these and other possible factors which could lead to improvements in the method, we explored the possibility that ΔCV training might provide a useful starting point for training on all 1000 examples. We fine-tuned the fit given by the ΔCV algorithm by training on all 1000 noisy examples. The final training error over the 1000 examples was 0.1018 rmse. The network quickly homed in on g, obtaining a close fit (with generalization error of 0.0187 rmse).

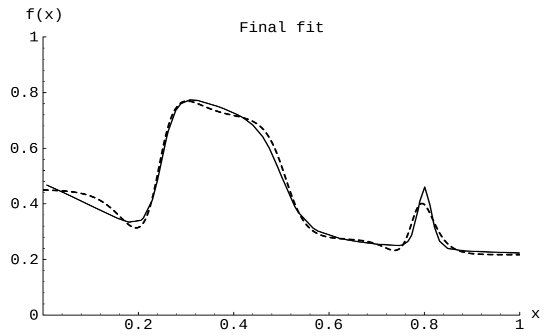


Figure VII.12: Final fit using results of training with ΔCV to provide initial weights for training on all 1000 examples.

VII.H Conclusion

This technique displayed intuitively pleasing behavior, both in its selection of exemplars for pruning, as well as in its selection of new exemplars. The selection method will tend to include exemplars that are "outliers" in some sense; however, the complementary pruning excises these. Although the technique did result in better generalization than any network trained on all the examples, the best results were obtained by using the best network obtained by active selection as a starting point for training on all the examples. The network quickly fine-tuned its fit, and homed in on a good solution in a few iterations. However, this version of the algorithm has some apparent deficiencies.

First, no clear stopping criterion is presented. Selecting one is problematic, since the algorithm can result in a sequence of network functions that vary greatly from iteration to iteration. We suspect this may have happened because of an approximation we made in computing the delete-1 cross-validation procedure. Instead of obtaining $l_{n-1}(z_{-i}^{n-1}; p)$ by initializing the weights to small random values, we start from $l_n(z^n; p)$, the weights obtained by training on all n exemplars. We also continued using weights from the previous iteration when training on a new set of exemplars. More experiments need to be done to evaluate the sensitivity of the ΔCV method to these cost-cutting measures.

Second, three additional parameters need to be specified. One regulates the initial size of the exemplar set, the second regulates the number of exemplars to prune at each iteration, and the third regulates the size of the exemplar set selected at each iteration. More work clearly needs to be done to automate the determination of these parameters. We also tie the number of exemplars to be selected to the size of the network, and the number to be pruned to the size of the exemplar set. More work needs to be done to justify or refute these design decisions.

Third, the experiment above illustrates that the algorithm is somewhat unstable, in that the network function can vary widely from iteration to iteration. This could be due to the way weights are initialized, as discussed above. More work needs to be done to obtain a more stable algorithm, such that it converges smoothly to the target function. One way this might be facilitated is by more rigorous pruning between successive selections. We expect that this may also result in the algorithm selecting more concise exemplar sets.

Fourth, the criterion was developed using ad hoc arguments. The reason for this was to facilitate an empirical investigation of intuitive ideas for a selection method for noisy data, to guide further analysis that will hopefully lead to a more stable and robust selection method.

The method did seem to do a fair job of selecting exemplars within close proximity of the target function. We conclude from the preliminary results that the technique merits further refinement.

Chapter VIII

Conclusions

We set out to obtain methods for selecting exemplars for training neural networks by optimizing an objective sampling criterion appropriate for use with the criterion used for learning. Although many interesting questions remain to be investigated, we have accomplished this task to a large degree.

The ΔISB is intuitively satisfying, as it admits a heuristic interpretation which is appealing in training sets of any size.

The empirical results with this method clearly indicate that it has several important benefits in general purpose use. Also, these results corroborate theoretical results that tie the generalization capability to the number of training examples, since more challenging learning tasks required larger networks, which required more exemplars to train.

The size of the resulting exemplar sets selected by this method are arguable concise, given the low ratio of the number of exemplars to the number of network weights. Results which predict neural network capacity in terms of the number of arbitrary exemplars that can be fit to arbitrary precision indicate that there exists a network model with capacity of $\Omega(p \log p)$, whereas ΔISB seemed to select sets smaller than this bound suggests. However, note that the bounds given in [71] are for a different network model, and consider a different learning task (namely, learning boolean mappings). The network models we used here may have VC dimension that grows slower than $\Omega(p \log p)$. Further results are required in order to calculate the VC dimension for the network models used here, to determine the conciseness of the exemplar sets selected by ΔISB.

The theoretical results of Chapter VI lay the foundation for methods of selecting exemplars from noisy data, as well as providing the first known general result of the relation between cross-validatory measures and the statistical measure of generalization known as Integrated Mean Squared Error. These results have their first application to active selection in the (admittedly ad hoc) extension of the ΔISB method, as applied to a noisy learning task in Chapter VII. These preliminary results, while for a method that obviously requires more refinement, are the first known practical method for selecting exemplars for use in general purpose nonlinear regression on noisy training examples.

Appendices

A Sample bound derivation

Here we obtain a sample bound which relates the sample size with the generalization error. By "generalization," here we mean that we wish the network function to be accurate within some small tolerance ϵ with probability $1-\delta$. We can state this δ,ϵ bound in terms of measures of network complexity by applying various results provided in [111]. Then, we apply approximations to simplify the resulting equation and focus on the most relevant components, those which relate to network complexity, such as the number of weights, and the maximum absolute magnitude of network weights.

We begin by instantiating Theorem 4.2(i) for a particular connectionist network consisting of one output, one hidden layer of q hidden units, and r inputs, where each unit employs the usual sigmoidal squashing function. To apply Theorem 4.2(i) to this class of connectionist networks stated requires that several of the conditions stated by the theorem be instantiated appropriately.[1]

Now, we state Theorem 4.2(i) of [111] as applied to a particular class of single output single hidden layer feedforward networks. For simplicity, assume that the networks we consider have a linear output, r inputs, and q sigmoided hidden units. Given a set of weights $w_n \in \Re^{q(r+2)+1}$, the network output for a particular input $x \in \Re^r$ is

$$f(x, w_n) \equiv \beta_0 + \sum_{j=1}^{q} \beta_j \psi(\tilde{x}' h_j), \tag{A.1}$$

where $w_n = (\beta', h')'$ is the $q(r+2)+1$ vector of network weights, $\beta = (\beta_0, \beta_1, \ldots, \beta_q)'$, $h = (h_0, h_1, \ldots, h_q)'$, $h_j = (h_{j0}, h_{j1}, \ldots, h_{jr})'$, for $j = 1, 2, \ldots, q$, h is the (given) hidden unit activation, and $\tilde{x} \equiv (1, x')'$, and ψ is a sigmoided squashing function. Further, let the magnitude of the network weights be constrained by a positive value Δ_n such that $\sum_{j=0}^{q} |\beta_j| \leq \Delta_n$, and $\sum_{j=1}^{q} \sum_{i=0}^{r} |h_{ji}| \leq q\Delta_n$. Let $(\Omega, \mathbf{F}, \mathbf{P})$ be a complete probability space, let (Θ, ρ) be a metric space, and let $\{\Theta_n\}$ be a (nonstochastic) increasing sequence of separable subsets of Θ. Fixing w_n gives a particular choice of $f(\cdot, w_n) = \theta \in \Theta_n$. Let $\{Z_t : \Omega \mapsto [0,1]^{r+1}\}$ be a stochastic sequence on $(\Omega, \mathbf{F}, \mathbf{P})$ such that $\{Z_t\}$ is an independent, identically distributed (i.i.d.) sequence. For concreteness, assume $Z_t = (X_t', Y_t')'$, where $X_t \in [0,1]^r$, and $Y_t \in [0,1]$. We observe a realization of a finite part of the sequence $\{Z_t\}$, a "training set" of size n, i.e., a realization of $Z^n \equiv (Z_1', Z_2', \ldots, Z_n')'$. Let $H_n(\epsilon)$ denote the metric entropy of Θ_n for balls of radius ϵ, and put $G_n(\cdot) = exp(H_n(\cdot))$. It follows from Theorem 4.2(i) for our situation that for any $\epsilon > 0$ and for all n sufficiently large,

$$\begin{aligned} & P\{\sup_{\theta \in \Theta_n} |\frac{1}{n} \sum_{t=1}^{n} (Y_t - \theta(X_t))^2 - E[(Y - \theta(X))^2]| > \epsilon\} \\ & \leq 2G_n(\frac{\epsilon}{24\Delta_n}) \left[exp(\frac{-6}{7} n) + exp(\frac{-\epsilon^2 n}{16\Delta_n^4[18 + 4\epsilon]}) \right]. \end{aligned} \tag{A.2}$$

[1] In particular, we specify Q_n, s_n, and m_n, from which $\bar{s_n}$, $\bar{m_n}$, and λ follow. We assume that the objective function being minimized by the learning rule is squared error, hence, we set $s_n(z, \theta) = (y - \theta(x))^2$. The method of least squares is implemented by setting $Q_n(\cdot, \theta) = \frac{1}{n} \sum_{t=1}^{n} (Y_t - \theta(X_t))^2$. The variable $\theta(\cdot)$ corresponds to the output of the network estimator for a particular input vector. For these conditions, it suffices to put $\bar{m}_n = 4\Delta_n$, $\bar{s}_n = 4\Delta_n^2$, and $\lambda = 1$, (page 544 of [111]) where Δ_n constrains the magnitude of weights.

Let $W_n = q(r+2)+1$, the number of weights in our network. We assume that the squashing function $\psi : \Re \mapsto \Re$ satisfies a Lipschitz condition $|\psi(a) - \psi(b)| \leq L|a - b|$, for all $a, b \in \Re$, for some $L \in \Re^{+}$. This allows us to assume the conditions utilized in the proofs of Lemmas 4.3 and 4.4 of [111] (see pp.547-548). Consequently, for n sufficiently large, we obtain a bound on the metric entropy in terms of the number and absolute magnitude of the network weights:

$$H_n\left(\frac{\epsilon}{24\Delta_n}\right) \leq 6W_n ln\Delta_n. \tag{A.3}$$

By substituting this inequality into Equation (A.2) we obtain

$$\begin{aligned} P\{\sup_{\theta\in\Theta_n} &|\frac{1}{n}\sum_{t=1}^{n}(Y_t - \theta(X_t))^2 - E\left[(Y - \theta(X))^2\right]| > \epsilon\} \\ &\leq 2\Delta_n^{6W_n}\left[exp(\frac{-6n}{7}) + exp(\frac{-\epsilon^2 n}{16\Delta_n^4[18+4\epsilon]})\right]. \end{aligned}$$

Therefore, $P\{\sup_{\theta\in\Theta_n}|\frac{1}{n}\sum_{t=1}^{n}(Y_t - \theta(X_t))^2 - E[(Y - \theta(X))^2]| > \epsilon\} \leq \delta$ for

$$\delta \leq 2\Delta_n^{6W_n}(e^{-6n/7} + e^{-\epsilon^2 n/(16\Delta_n^4(18+4\epsilon))}).$$

We note that $e^{-\epsilon^2 n/(16\Delta_n^4(18+4\epsilon))} > e^{-6n/7}$ when $\Delta_n^4 > (\frac{-\epsilon^2 n}{16\Delta_n^4(18+4\epsilon)})(\frac{7}{-6n}) = \frac{7\epsilon^2}{96(18+4\epsilon)}$. This allows us to eliminate one of the exponential terms, so that for $\Delta_n^4 > \frac{7\epsilon^2}{96(18+4\epsilon)}$,

$$\delta < 4\Delta_n^{6W_n}e^{-\epsilon^2 n/(16\Delta_n^4(18+4\epsilon))}, \tag{A.4}$$

$$ln\delta < ln4 + ln\Delta_n^{6W_n} + -\epsilon^2 n/(16\Delta_n^4(18+4\epsilon)). \tag{A.5}$$

We may determine the number of examples n sufficient to obtain a particular δ,ϵ bound by fixing δ, ϵ and solving for n. We obtain $P\{\sup_{\theta\in\Theta_n}|\frac{1}{n}\sum_{t=1}^{n}(Y_t - \theta(X_t))^2 - E[(Y - \theta(X))^2]| > \epsilon\} \leq \delta$ when

$$n > \frac{16\Delta_n^4(18+4\epsilon)}{\epsilon^2}\left(6W_n ln\Delta_n + ln\frac{4}{\delta}\right), \tag{A.6}$$

for n sufficiently large, and $\Delta_n^4 > \frac{7\epsilon^2}{96(18+4\epsilon)}$. (The latter condition is reasonable when the weights are allowed to have a sufficiently large absolute magnitude, or, for small bounded weights, when ϵ is sufficiently small.)

This bound is quite similar to bounds achieved in other contexts (cf. [44].) This bound is strongly influenced by the bound placed upon the network weights, Δ_n, and on the total number of network weights, W_n. Note that these variables may depend upon n, therefore, to interpret this bound, we need to fix ϵ and δ, and determine the number of examples necessary to achieve this δ,ϵ level of learning performance for a particular network complexity, as measured by Δ_n and W_n. Hence, for reasonable, fixed choices of ϵ and δ, the bound is approximately

$$n = O\left(\Delta_n^4 W_n ln\Delta_n\right).$$

B The benefit of a single example

Consider the class of single hidden layer single output feedforward neural networks referred to in Appendix A. In Appendix A, (Equation (A.5)) we obtained a bound on the probability that the empirical estimate of the sample error differs from the true error by more than ϵ for this class of network estimators. Let $(X_t, Y_t), t = 1, 2, \ldots, n$ be a training set obtained by an i.i.d. sample of the environment according to an environmental distribution. (Refer to Appendix A for precise definitions.) For $\delta > 0$, $\epsilon > 0$, we denote the probability that the empirical estimation error $\frac{1}{n}\sum_{t=1}^{n}(Y_t - \theta(X_t))^2$ differs from the true expectation of the network estimation error $E[(Y - \theta(X))^2]$ by more than ϵ as

$$P\left[\sup_{\theta\in\Theta_n}\left|\frac{1}{n}\sum_{t=1}^{n}(Y_t - \theta(X_t))^2 - E[(Y-\theta(X))^2]\right| > \epsilon\right]$$

This probability is no more than δ_n when

$$\delta_n \quad < \quad 4\Delta_n^{6W_n} e^{-\epsilon^2 n/(16\Delta_n^4(18+4\epsilon))} \tag{B.7}$$

for $\Delta_n^4 > \frac{7\epsilon^2}{96(18+4\epsilon)}$. As in Appendix A, $\theta(\cdot)$ corresponds to a network function, W_n quantifies the number of network weights, and Δ_n is a bound on the magnitude of the network weights. See Appendix A for precise details. W_n and Δ_n can be viewed as quantifying network complexity. We note that the bound given by (B.7) depends heavily upon the network complexity. Note that W_n and Δ_n can vary with the number of examples, n. In what follows, let Δ_n and W_n be fixed, such that for all n, $\Delta_n = \Delta$ and $W_n = W$. For fixed n, ϵ, we rewrite Equation (B.7) as

$$\delta_n < c_1 e^{-nc_2},$$

where $c_1 = 4\Delta^{6W}$ and $c_2 = \epsilon^2/(16\Delta^4(18+4\epsilon))$. As n increases, δ_n decreases at a rate exponential in n. Of interest is how much of a decrement in δ_n is obtained by appending a single example to the training set. that is, what can be said about $\delta_n - \delta_{n+1}$ when all other quantities are fixed. Intuitively, we are evaluating how much a single additional example helps increase the reliability of the estimate. We look at how fast the difference in the respective upper bounds drop with increasing n. For fixed n, ϵ, this difference is

$$\begin{aligned}
\delta_{n+1} - \delta_n &= c_1 e^{-nc_2} - c_1 e^{-(n+1)c_2} \\
&= c_1(e^{c_2} - 1)e^{-c_2(n+1)} \\
&= c_3 e^{-c_2 n}, \qquad \text{(B.8)}
\end{aligned}$$

for some constant c_3. From Equation (B.8) we see that the difference between δ_n and δ_{n+1} decreases as n grows large, but this decrease is exponential in n. Therefore, in this particular sense, the benefit of appending a single example to the training set decreases exponentially in the size of the training set.

C IMSE and Statistical Risk

Here we relate IMSE with another well-known measure of generalization, known as statistical "risk" or "regret." This helps illuminate the meaning of IMSE for those more familiar with the use of statistical risk. Also, convergence results for IMSE are then readily obtained by using a complexity regularization estimator in the definition of IMSE [11, 12, 13]. In particular, we show that $IMSE_n$ is equivalent to the expected squared risk, that is, that

$$\begin{aligned} E\left[\|f(\cdot, l_n(Z^n; p)) - g\|_2^2\right] &= \int IMSE_n(x^n; p)\mu^n(dx^n) \\ &= IMSE_n(p), \end{aligned}$$

and conversely, that the conditional IMSE for a given set of n examples with particular inputs x^n is equivalent to the expected squared risk conditioned on that set of inputs:

$$E\left[\|f(\cdot, l_n(Z^n; p)) - g\|_2^2 \mid x^n\right] = IMSE_n(x^n; p).$$

To obtain these results we need to be more precise in our definitions. Let $\mathbf{Z}_i = \mathbf{X}_i \times \mathbf{Y}_i$. $\mathbf{X}_i$ and $\mathbf{Y}_i$ are finite-dimensional euclidean spaces. Consider the sequence of measure spaces $(\mathbf{Z}_i, \mathcal{B}(\mathbf{Z}_i), \nu_i)$, as well as the sequences of measurable spaces $(\mathbf{X}_i, \mathcal{B}(\mathbf{X}_i))$, and $(\mathbf{Y}_i, \mathcal{B}(\mathbf{Y}_i)), i = 1, 2, \ldots, n$. For each i, $i = 1, 2, \ldots, n$, decompose ν_i into an environmental probability law μ_i and a conditional probability law γ_i. Therefore, μ_i is a measure on $\mathbf{X}_i$ and $\nu_i(A_i \times B_i) = \gamma_i(B_i|A_i)\mu_i(A_i)$, $A_i \in \mathcal{B}(\mathbf{X}_i), B_i \in \mathcal{B}(\mathbf{Y}_i)$.

Abusing notation, let $\mathbf{Z}^n \equiv \prod_{i=1}^n \mathbf{Z_i}$. Let $G \in \mathcal{B}(\mathbf{Z^n})$, where $G = \prod_{i=1}^n G_i, G_i \in \mathcal{B}(\mathbf{Z}_i)$ so that $G_i = A_i \times B_i, A_i \in \mathcal{B}(\mathbf{X}_i)$, $B_i \in \mathcal{B}(\mathbf{Y}_i)$. Assuming that each set G_i is independent of $\mathcal{B}(\mathbf{Z}_j)$ for $i \neq j$, then $\nu^n \equiv \prod_{i=1}^n \nu_i$ is the unique product measure on $(\mathbf{Z^n}, \mathcal{B}(\mathbf{Z^n}))$ ([38], p.62), so

$$\begin{aligned} v^n(G) &= v^n(\prod_{i=1}^n G_i) \\ &= \prod_{i=1}^n v_i(G_i) \\ &= \prod_{i=1}^n v_i(A_i \times B_i). \end{aligned}$$

Therefore, the (product) measure ν^n of a set G is given by the product of measures over orthogonal projections G_i of G. Define

$$\mu^n \equiv \prod_{i=1}^n \mu_i.$$

Note that μ^n is the unique product measure on $(\mathbf{X^n}, \mathcal{B}(\mathbf{X^n}))$, (where $\mathbf{X^n}$ is defined similarly to $\mathbf{Z^n}$). For $A \in \mathcal{B}(\mathbf{X}^n)$, $B \in \mathcal{B}(\mathbf{Y}^n)$, we may (heuristically) consider the decomposition of ν^n given by

$$\nu^n(A^n \times B^n) = \gamma^n(B^n|X^n \in A^n)\mu^n(A^n).$$

In Appendix D, we construct a conditional probability law $\gamma^n(\cdot|X^n)$ allowing the decomposition of ν^n, so that for a given value of the random variable X^n, whose distribution is given by μ^n, $\gamma^n(\cdot|X^n)$ gives a probability measure over $\mathcal{B}(\mathbf{Y}^n)$ for each value taken on by X^n. There, we show that due to the uniqueness of the constructed probability measure, that

$$\gamma^n(A^n|X^n) \equiv \prod_{i=1}^{n} \gamma_i(A_i|X_i).$$

For notational convenience, let $\hat{f}_n \equiv f(\cdot, l_n(Z^n; p))$. We then apply the Fubini-Tonelli theorem to derive the equivalence between the expectation of the squared regret and the expectation of the IMSE in the case of i.i.d. samples:

$$\begin{aligned}
E\left[||\hat{f}_n - g||_2^2\right] &= \int_{\mathbf{Z^n}} \int_{\mathbf{X}} (g(x) - \hat{f}_n(x))^2 \mu(dx) \nu^n(dx^n, dy^n) \\
&= \int_{\mathbf{X^n}} \int_{\mathbf{Y^n}} \int_{\mathbf{X}} (g(x) - \hat{f}_n(x))^2 \mu(dx) \gamma^n(dy^n|x^n) \mu^n(dx^n) \\
&= \int_{\mathbf{X^n}} \int_{\mathbf{X}} \int_{\mathbf{Y^n}} (g(x) - \hat{f}_n(x))^2 \gamma^n(dy^n|x^n) \mu(dx) \mu^n(dx^n) \\
&= \int_{\mathbf{X^n}} \int_{\mathbf{Y^n}} M(x|x^n) \mu(dx) \mu^n(dx^n) \\
&= \int_{\mathbf{X^n}} IMSE_n(x^n; p) \mu^n(dx^n) \\
&= IMSE_n(p).
\end{aligned}$$

Conditioning the expected squared regret upon a particular set of input locations provides equality with IMSE,

$$\begin{aligned}
E\left[||\hat{f}_n - g||_2^2 \,|\, x^n\right] &= \int_{\mathbf{Y^n}} ||g - \hat{f}_n||_2^2 \gamma^n(dy^n|x^n) \\
&= \int_{\mathbf{Y^n}} \int_{\mathbf{X}} (g(x) - \hat{f}_n(x))^2 \mu(dx) \gamma^n(dy^n|x^n) \\
&= \int_{\mathbf{X}} \int_{\mathbf{Y^n}} (g(x) - \hat{f}_n(x))^2 \gamma^n(dy^n|x^n) \mu(dx) \\
&= \int_{\mathbf{X}} M(x|x^n) \mu(dx) \\
&= IMSE_n(x^n; p).
\end{aligned}$$

D Modelling the data generation process.

In this section we describe the formal model of the data generation process in more precise detail. In particular, we demonstrate the construction of the conditional probability spaces used throughout this dissertation.

We model the data generation process in a fashion similar to the more general setup described in Chapter 2 of [112]. Refer to Chapter III for additional definitions. Let $(\Omega, \mathcal{F}, \mathcal{P})$ be a complete probability space, where $\Omega = \Re^{(r+1)\infty} \equiv \times_{i=1}^{\infty} \Re^{r+1}$, r an integer, and $\mathcal{F} = \mathcal{B}(\Omega)$. Training examples are a realization of a stochastic process $Z \equiv \{Z_i : \Omega \to \Re^{r+1}, i = 1, 2, \ldots\}$. For every $i = 1, 2, \ldots$, we partition observation Z_i into the input X_i and the output Y_i, so that $Z_i = (X_i, Y_i)$, $X_i : (\Omega, \mathcal{F}) \to (\mathbf{X_i}, \mathcal{B}(\mathbf{X_i})), \mathbf{X_i} \subset \Re^r$, $Y_i : (\Omega, \mathcal{F}) \to (\mathbf{Y_i}, \mathcal{B}(\mathbf{Y_i})), \mathbf{Y_i} \subset \Re$, $Z_i : (\Omega, \mathcal{F}) \to (\mathbf{Z_i}, \mathcal{B}(\mathbf{Z_i})), \mathbf{Z_i} = \mathbf{X_i} \times \mathbf{Y_i}$.[2] This implies, for instance, that for every Borel set $B \subset \mathcal{B}(\mathbf{Y_i})$ we have $Y^{-1}(B) = \{\omega : X(\omega) \in B\} \in \mathcal{F}$. Recall that $P(Z_i \in B) = P(\{\omega \in \Omega : Z_i(\omega) = z, z \in B\}), B \in \mathcal{B}(\mathbf{Z_i})$. Let ν_i be the distribution on $\mathbf{Z_i}$ induced by Z_i, so that for $B \in \mathcal{B}(\mathbf{Z_i})$, $\nu_i(B) = P(Z_i \in B)$. Likewise, let μ_i be the distribution on $\mathbf{X_i}$ and induced by X_i. The joint distribution of X_i and Y_i is given by ν_i.

Assume that ν_i is absolutely continuous with respect to Lebesgue measure. (This is reasonable, for instance, when Z_i is continuously distributed.) We may now provide an example of a construction of a conditional distribution over $\mathbf{Y}_i$. (Alternatively, if the components of Z_i were restricted to a countable set, we could repeat this construction by instead assuming ν is absolutely continuous with respect to the counting measure. More general conditions can also be handled (cf., [112]).) Let $f(x, y)$ be the density of ν_i with respect to Lebesgue measure such that for $A \in \mathcal{B}(\mathbf{Z}_i)$,

$$\nu_i(A) = \int \int_A f_i(x, y) dx dy.$$

For each $x \in \mathbf{X_i}$, $i = 1, 2, \ldots$, there exists a function $\gamma_i(\cdot|x)$ giving a probability measure on $\mathbf{Y_i}$ with density $g_i(x, y)$, where

$$g_i(x, y) = \frac{f_i(x, y)}{\int_{\mathbf{Y_i}} f_i(x, t) dt}$$

such that for each $F \in \mathcal{B}(\mathbf{Y_i})$,

$$\gamma_i(F|x) = \begin{cases} \int_F g_i(x, y) dy & \text{if } \int_{\mathbf{Y_i}} f_i(x, y) dy \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{D.9}$$

$\gamma_i(F|x)$ is a version of $P[\mathbf{X_i} \times F \| \sigma(X_i)]$, the conditional probability of $\mathbf{X_i} \times F$ given X_i, where $\sigma(X_i) \subset \mathcal{F}$ is the subsigma-field generated by X_i. Thus, $\gamma_i(\cdot|x)$ is the conditional distribution of Y_i given $X_i(\omega) = x$. [cf. Billingsley, 1986, section 33.]

We construct a conditional probability with respect to a sample of size m. For each $m = 1, 2, \ldots, n$, let $z^m = (z_1, z_2, \ldots, z_m)$, and let $P^m = \times_{i=1}^{m} P$. These finite euclidean spaces are separable, therefore $\mathcal{B}(\mathbf{Z^m}) = \mathcal{B}(\mathbf{Z_1}) \bigotimes \mathcal{B}(\mathbf{Z_2}) \bigotimes \ldots \bigotimes \mathcal{B}(\mathbf{Z_m})$. (Proposition

[2] Therefore, X_i, Y_i, and Z_i are measurable random variables, each inducing a corresponding measure space.

(1.5), [Folland, 1984].) Let $B^m = \times_{i=1}^m B_i, B_i \in \mathcal{B}(\mathbf{Z_i})$. Then, when the Z_i are i.i.d., $1 \leq i \leq m$,

$$P^m(B^m) = \prod_{i=1}^m P(B_i). \tag{D.10}$$

Since v^m is absolutely continuous with respect to Lebesgue measure, by the Radon-Nikodym theorem there exists h^m such that

$$v^m(B^m) = \int_{B^m} h^m(z^m)dz^m, \quad B^m \in \mathcal{B}(\mathbf{Z^m}). \tag{D.11}$$

For each $m = 1, 2, \ldots, n$, let $f^m(z^m) \equiv \prod_{i=1}^m f_i(z_i)$. Due to the uniqueness of the Radon-Nikodym derivative, we may expand Equation (D.10) into integral form, and use the Fubini theorem to show that by Equation (D.11), $\int_B h^m(z^m)dz^m = \int_B f^m(z^m)dz^m$, $B \in \mathcal{B}(\mathbf{Z^m})$, which, by the $\sigma-$finiteness of Lebesgue measure, implies that $h^m(z^m) = f^m(z^m)$, a.e. [cf. Billingsley, 1986, section 16.]

Define $\gamma^m(F^m|x^m) = \prod_{i=1}^m \gamma_i(F_i|x_i)$, $F^m \in \mathcal{B}(\mathbf{Y_m})$. We proceed to show that for each $x^m \in \mathbf{X_m}$, $m = 1, 2, \ldots$, $\gamma^m(F^m|x^m)$ gives a conditional probability measure on $\mathbf{Y^m}$ for each $x^m \in \mathbf{X^m}$. To do so, it suffices to demonstrate two properties, which we will do next. For clarity in what follows, let the function $\varphi^m(x^m, F^m) = \gamma^m(F^m|x^m)$, and $\varphi_i(x_i, F_i) = \gamma_i(F_i|x_i)$, $F_i \in \mathcal{B}(\mathbf{Y_i})$, $i = 1, 2, \ldots$.

1. $\varphi^m \in L^1(\Omega, P)$.

   $\varphi_i(\cdot, \mathbf{X_i} \times F_i)$ is $\mathcal{B}(\mathbf{X_i})$-measurable for each $F_i \in \mathcal{B}(\mathbf{Y_i})$, hence is measurable-$\sigma(X_i)$, $i = 1, 2, \ldots$. Therefore, $\varphi^m(\cdot, \mathbf{X^m} \times F^m)$ is $\mathcal{B}(\mathbf{X^m})$-measurable for each $F^m \in \mathcal{B}(\mathbf{Y^m})$, hence is measurable-$\sigma(X^m)$. P finite implies $h^m(x^n, \cdot) \in L^1(\mathbf{Y^m})$, for each $x^m \in \mathbf{X^m}$, so $0 \leq \varphi_i \leq 1$. Clearly $\varphi^m \in L^1(\Omega, P)$.

2. $\int_{A^m} P^m[B^m|X^m = x^m]dP = P(A^m \cap B^m)$, $A^m, B^m \in \mathcal{B}(\mathbf{X^m} \times \mathbf{Y^m})$.

   This follows by Fubini's theorem and by independence of the Z_i. Let $x^m = (x_1, x_2, \ldots, x_m)$, $y^m = (y_1, y_2, \ldots, y_m)$, $F^m = \times_{i=1}^m F_i$, $F_i \in \mathcal{B}(\mathbf{Y_i})$, and $E^m = \times_{i=1}^m E_i$, $E_i \in \mathcal{B}(\mathbf{X_i})$. It suffices to consider $A^m = E^m \times \mathbf{Y^m}$, $B^m = \mathbf{X^m} \times F^m$. Then,

$$\begin{aligned}
&\int_{A^m} P^m[B^m|X^m = x^m]dP \\
=\ & \int_{A^m} \varphi^m(x^m, F^m)P(dz^m) \\
=\ & \int_{A_1}\int_{A_2}\cdots\int_{A_m} \varphi^m(x^m, F^m)f^m(z^m)dz_m \ldots dz_2dz_1 \\
=\ & \int_{\mathbf{X}_1}\int_{E_1}\cdots\int_{\mathbf{X}_m}\int_{E_m} \varphi^m(x^m, F^m)f^m(x^m, y^m)dy_m dx_m \ldots dy_1 dx_1 \\
=\ & \prod_{i=1}^m P[Z_i \in E_i \times F_i] \\
=\ & \prod_{i=1}^m P(A_i \cap B_i) \\
=\ & P(A^m \cap B^m),
\end{aligned}$$

   by independence.

E Proofs of Theorems

Proof of Lemma 1 Part 1 follows from Lemma 2 of Jennrich [52] and a measurable selection theorem. Part 2 follows from the Lebesgue dominated convergence theorem with Assumptions 1 through 5. That $\mathbf{W}^{p*}$ is non-empty follows because λ is continuous and $\mathbf{W}^p$ is compact. □

Proof of Proposition 1 By the law of iterated expectations and the results of subsection III.D.2, we have

$$\begin{aligned}
E\left[C_N(Z^N;p)\right] &= \frac{1}{N}\sum_{i=1}^{N} E\left[\left(Y_i - f^p(X_i, l_{N-1}(Z_{-i}^N;p))\right)^2\right] \\
&= \frac{1}{N}\sum_{i=1}^{N} E\left[E\left[\left(Y_i - f^p(X_i, l_{N-1}(Z_{-i}^N;p))\right)^2 |Z_{-i}^N\right]\right] \\
&= \frac{1}{N}\sum_{i=1}^{N} E\left[Q_{N-1}(Z_{-i}^N;p)\right] \\
&= IMSE_{N-1}(p) + \sigma^2. \ \square
\end{aligned}$$

Proof of Proposition 2 The proof is similar to that for Proposition 1:

$$\begin{aligned}
E\left[G_{N,M}(Z^N, Z^M;p)\right] &= \frac{1}{M}\sum_{i=N+1}^{N+M} E\left[\left(Y_i - f^p(X_i, l_N(Z^N;p))\right)^2\right] \\
&= \frac{1}{M}\sum_{i=N+1}^{N+M} E\left[E\left[\left(Y_i - f^p(X_i, l_N(Z^N;p))\right)^2 |Z^N\right]\right] \\
&= \frac{1}{M}\sum_{i=N+1}^{N+M} E\left[Q_N(Z^N;p)\right] \\
&= IMSE_N(p) + \sigma^2. \ \square
\end{aligned}$$

Proof of Proposition 3 The proof is similar to Proposition 1:

$$\begin{aligned}
&E\left[C_N(Z^N,p)\middle| X^N = x^N\right] \\
&= \frac{1}{N}\sum_{i=1}^{N} E\left[\left(Y_i - f^p(X_i, l_{N-1}(Z_{-i}^N;p))\right)^2\middle| X^N = x^N\right] \\
&= \frac{1}{N}\sum_{i=1}^{N} E\left[E\left[\left(Y_i - f^p(X_i, l_{N-1}(Z_{-i}^N;p))\right)^2\middle| Z_{-i}^N, X^N = x^N\right]\middle| X^N = x^N\right] \\
&= \frac{1}{N}\sum_{i=1}^{N} E\left[\left(g_i(X_i) - f^p(X_i, l_{N-1}(Z_{-i}^N;p))\right)^2 + \sigma^2\middle| X^N = x^N\right] \\
&= \frac{1}{N}\sum_{i=1}^{N} E\left[\left(g_i(X_i) - f^p(X_i, l_{N-1}(Z_{-i}^N;p))\right)^2\middle| X^N = x^N\right] + \sigma^2 \\
&= \frac{1}{N}\sum_{i=1}^{N} M_N(x_i|x_{-i}^N;p) + \sigma^2. \ \square
\end{aligned}$$

Proof of Proposition 4

$$
\begin{aligned}
& E\left[G_{N,M}(Z^N, \tilde{Z}^M; p)\Big| X^N = x^N\right] \\
&= \frac{1}{M}\sum_{i=N+1}^{N+M} E\left[\left(Y_i - f^p(X_i, l_N(Z^N;p))\right)^2 \Big| X^N = x^N\right] \\
&= \frac{1}{M}\sum_{i=N+1}^{N+M} E\left[E\left[\left(Y_i - f^p(X_i, l_N(Z^N;p))\right)^2 \Big| Z^N, X^N = x^N\right]\Big| X^N = x^N\right] \\
&= \frac{1}{M}\sum_{i=N+1}^{N+M} E\left[\left(g(X_i) - f^p(X_i, l_N(Z^N;p))\right)^2 + \sigma^2 \Big| X^N = x^N\right] \\
&= \frac{1}{M}\sum_{i=N+1}^{N+M} E\left[\left(g(X_i) - f^p(X_i, l_N(Z^N;p))\right)^2 \Big| X^N = x^N\right] + \sigma^2 \\
&= \frac{1}{M}\sum_{i=N+1}^{N+M} E\left[E\left[\left(g(X_i) - f^p(X_i, l_N(Z^N;p))\right)^2 \Big| X_i\right]\Big| X^N = x^N\right] + \sigma^2 \\
&= \frac{1}{M}\sum_{i=N+1}^{N+M} E\left[M_N(X_i|X^N;p)\Big| X^N = x^N\right] + \sigma^2 \\
&= \frac{1}{M}\sum_{i=N+1}^{N+M} IMSE_N(x^N;p) + \sigma^2. \ \square
\end{aligned}
$$

Proof of Theorem 1 By Assumption 1, 4 and 5, $(y - f^p(x,w))^2$ is continuous in w and dominated by $2|y|^2 + D(x)^2$ for each $x \in \mathbf{X}$ and $y \in \mathbf{Y}$ with $E\left[2|Y|^2 + 2D(X)^2\right] < \infty$. This allows us to apply Jennrich's uniform law of large numbers (ULLN, Jennrich [52, Theorem 2]) to $\{(Y_i - f^p(X_i, \cdot))^2 : \mathbf{W}^p \to \mathbf{Y}\}_{i\in\mathbf{N}}$. Therefore $\lambda(\cdot;p)$ is continuous on $\mathbf{W}^p$ and there exists $F_1 \in \mathcal{F}$ with $\mathcal{P}(F_1) = 1$ such that for each $\omega \in F$

$$
\sup_{w\in\mathbf{W}^p}\left|N^{-1}\sum_{i=1}^{N}(Y_i(\omega) - f^p(X_i(\omega), w))^2 - \lambda(w;p)\right| \to 0
$$

as $N \to \infty$. In addition, for each N, there exists $F_{2N} \in \mathcal{F}$ with $\mathcal{P}(F_{2N}) = 1$ such that for each $\omega \in F_{2N}$ $l_N(Z^N(\omega);p)$ solves

$$
\min_{w\in\mathbf{W}^p} N^{-1}\sum_{i=1}^{N}(Y_i(\omega) - f^p(X_i(\omega), w))^2.
$$

Let $F \equiv F_1 \cap (\cap_{N=1}^{\infty} F_{2N})$. Then $\mathcal{P}(F) = 1$.

Let ω be arbitrary in F and $\epsilon > 0$ be arbitrary. Then there exists N_0 such that for any $N > N_0$

$$
\sup_{w\in\mathbf{W}^p}\left|N^{-1}\sum_{i=1}^{N}(Y_i(\omega) - f^p(X_i(\omega), w))^2 - \lambda(w;p)\right| < \frac{\epsilon}{1 + (2/A)}
$$

and

$$
\sup_{w\in\mathbf{W}^p}\left|(N + M_N)^{-1}\sum_{i=1}^{N+M_N}(Y_i(\omega) - f^p(X_i(\omega), w))^2 - \lambda(w;p)\right| < \frac{\epsilon}{1 + (2/A)}.
$$

This implies that for any $N > N_0$

$$\begin{aligned}
&\sup_{w\in\mathbf{W}^p}\left|M_N^{-1}\sum_{i=N+1}^{N+M_N}(Y_i(\omega)-f^p(X_i(\omega),w))^2-\lambda(w;p)\right| \\
\le\ & \sup_{w\in\mathbf{W}^p}\left|\frac{N+M_N}{M_N}(N+M_N)^{-1}\sum_{i=1}^{N+M_N}\{(Y_i(\omega)-f^p(X_i(\omega),w))^2-\lambda(w;p)\}-\right. \\
& \left.\frac{N}{M_N}N^{-1}\sum_{i=1}^{N}\{(Y_i(\omega)-f^p(X_i(\omega),w))^2-\lambda(w;p)\}\right| \\
\le\ & \frac{N+M_N}{M_N}\sup_{w\in\mathbf{W}^p}\left|(N+M_N)^{-1}\sum_{i=1}^{N+M_N}\{(Y_i(\omega)-f^p(X_i(\omega),w))^2-\lambda(w;p)\}\right|+ \\
& \frac{N}{M_N}\sup_{w\in\mathbf{W}^p}\left|N^{-1}\sum_{i=1}^{N}\{(Y_i(\omega)-f^p(X_i(\omega),w))^2-\lambda(w;p)\}\right| \\
\le\ & \frac{N+M_N}{M_N}\frac{\epsilon}{1+(2/A)}+\frac{N}{M_N}\frac{\epsilon}{1+(2/A)} \\
\le\ & \left(\frac{N+M_N}{M_N}+\frac{N}{M_N}\right)\frac{\epsilon}{1+(2/A)} \\
\le\ & \left(1+\frac{2N}{M_N}\right)\frac{\epsilon}{1+(2/A)} \\
\le\ & \left(1+\frac{2N}{AN}\right)\frac{\epsilon}{1+(2/A)} \\
=\ & \epsilon.
\end{aligned}$$

Thus for any $N > N_0$, we have

$$\sup_{w\in\mathbf{W}^p}\left|N^{-1}\sum_{i=1}^{N}(Y_i(\omega)-f^p(X_i(\omega),w))^2-\lambda(w;p)\right|<\epsilon.$$

and

$$\sup_{w\in\mathbf{W}^p}\left|M_N^{-1}\sum_{i=N+1}^{N+M_N}(Y_i(\omega)-f^p(X_i(\omega),w))^2-\lambda(w;p)\right|<\epsilon.$$

Because $l_N(Z^N(\omega);p)$ minimizes the sum of the squared residuals, the first inequality implies that

$$|\lambda(l_N(Z^N(\omega);p);p)-\lambda(w^{p*};p)|<\epsilon$$

as easily verified, where w^{p*} is an arbitrary element of $\mathbf{W}^{p*}$. By using the second inequality with this relationship, we have

$$\begin{aligned}
&\left|G_{N,M_N}(Z^N(\omega),\tilde{Z}^{M_N}(\omega);p)-\lambda(w^{p*};p)\right| \\
=\ & \left|M_N^{-1}\sum_{i=N+1}^{N+M_N}(Y_i(\omega)-f^p(X_i(\omega),l_N(Z^N(\omega);p)))^2-\lambda(w^{p*};p)\right| \\
\le\ & \left|M_N^{-1}\sum_{i=N+1}^{N+M_N}(Y_i(\omega)-f^p(X_i(\omega),l_N(Z^N(\omega);p)))^2-\lambda(l_N(Z^N(\omega);p))\right|+
\end{aligned}$$

$$
\begin{aligned}
& \left|\lambda(l_N(Z^N(\omega);p)) - \lambda(w^{p*};p)\right| \\
\leq \;& 2\epsilon.
\end{aligned}
$$

Because ϵ is arbitrary, this implies that

$$|G_{N,M_N}(Z^N(\omega), \tilde{Z}^{M_N}(\omega);p) - \lambda(w^{p*};p)| \to 0$$

as $N \to \infty$. Further this holds for all $\omega \in F$. The result therefore follows. $\square$

Proof of Lemma 2 Suppose that the claim is not true. Then there must exist a constant $M > 0$, a strictly increasing sequence $\{N_j\}_{j \in \mathbf{N}}$ in $\mathbf{N}$, a sequence $\{\theta^j\}_{j \in \mathbf{N}}$ in Θ and a sequence $\{k_j\}_{j \in \mathbf{N}}$ in $\mathbf{N}$ such that $k_j \leq N_j$ and

$$\left|(N_j - 1)^{-1} \sum_{1 \leq i \leq N_j, i \neq k_j} h_i(\theta^j)\right| \geq M$$

for each $j \in \mathbf{N}$. This implies that

$$
\begin{aligned}
\frac{M}{2} &\leq \left|N_j^{-1} \sum_{1 \leq i \leq N_j, i \neq k_j} h_i(\theta^j)\right| \\
&= \left|N_j^{-1} \sum_{i=1}^{N_j} h_i(\theta^j) - N_j^{-1} h_{k_j}(\theta)\right| \\
&\leq \left|N_j^{-1} \sum_{i=1}^{N_j} h_i(\theta^j)\right| + \left|N_j^{-1} h_{k_j}(\theta^j)\right|
\end{aligned}
$$

By rearranging this relation, we have

$$N_j^{-1} \left|h_{k_j}(\theta^j)\right| \geq \frac{M}{2} - \left|N_j^{-1} \sum_{i=1}^{N_j} h_i(\theta^j)\right|.$$

The second term of the RHS converges to zero as $j \to \infty$. Therefore for all j sufficiently large, it holds that

$$N_j^{-1} \left|h_{k_j}(\theta^j)\right| \geq \frac{M}{4}. \tag{E.12}$$

On the other hand, if there exists $N_0 \in \mathbf{N}$ such that $k_j \leq N_0$ for all $j \in \mathbf{N}$, we have

$$N_j^{-1} \left|h_{k_j}(\theta^j)\right| \leq N_j^{-1} \max_{j \in \{1,\ldots,N_0\}} \max_{\theta \in \Theta} h_i(\theta) \to 0 \text{ as } j \to \infty.$$

Hence the left hand side of inequality (E.12) converges to zero. It therefore must hold that $k_j \to \infty$ as $j \to \infty$. By assumption and this fact, we have

$$\left|k_j^{-1} \sum_{i=1}^{k_j} h_i(\theta^j)\right| < \frac{M}{8}$$

and

$$\left|(k_j - 1)^{-1} \sum_{i=1}^{k_j - 1} h_i(\theta^j)\right| < \frac{M}{8}$$

for all j sufficiently large. This implies that

$$\begin{aligned}\left|k_j^{-1} h_{k_j}(\theta^j)\right| &= \left|k_j^{-1}\sum_{i=1}^{k_j} h_i(\theta^j) - k_j^{-1}\sum_{i=1}^{k_j-1} h_i(\theta^j)\right| \\ &\le \left|k_j^{-1}\sum_{i=1}^{k_j} h_i(\theta^j)\right| + \left|k_j^{-1}\sum_{i=1}^{k_j-1} h_i(\theta^j)\right| \\ &\le \left|k_j^{-1}\sum_{i=1}^{k_j} h_i(\theta^j)\right| + \left|(k_j-1)^{-1}\sum_{i=1}^{k_j-1} h_i(\theta^j)\right| \\ &< \frac{M}{4}. \end{aligned} \tag{E.13}$$

Equations (E.12) and (E.13) imply that

$$\frac{M}{4} \le N_j^{-1}\left|h_{k_j}(\theta^j)\right| \le k_j^{-1}\left|h_{k_j}(\theta^j)\right| < \frac{M}{4},$$

for all j sufficiently large, which is obviously impossible because $M > 0$. The result therefore follows. $\square$

Proof of Theorem 2 By Assumption 1, 4 and 5, $(y - f^p(x,w))^2$ is continuous in w and dominated by $(D(x,y))^2$ for each $x \in \mathbf{X}$ and $y \in \mathbf{Y}$ with $E\left[(D(X,Y))^2\right] < \infty$, given Assumptions 2 and 5. This allows us to apply Jennrich's uniform law of large numbers (ULLN, Jennrich [52, Theorem 2]) to $\{(Y_i - f^p(X_i,\cdot))^2 : \mathbf{W}^p \to \Re\}_{i\in\mathbf{N}}$. Thus there exists $F_1 \in \mathcal{F}$ such that for each $\omega \in F_1$

$$\sup_{w\in\mathbf{W}^p}\left|N^{-1}\sum_{i=1}^{N}(Y_i(\omega) - f^p(X_i(\omega),w))^2 - \lambda(w;p)\right| \to 0 \text{ as } N \to \infty. \tag{E.14}$$

By Lemma 2 this implies that for each $\omega \in F_1$

$$\max_{k\in(1,\ldots,N)}\sup_{w\in\mathbf{W}^p}\left|(N-1)^{-1}\sum_{1\le i\le N, i\ne k}(Y_i(\omega) - f^p(X_i(\omega),w))^2 - \lambda(w;p)\right| \to 0 \text{ as } n \to \infty. \tag{E.15}$$

By Lemma 1, for each $N \in \mathbf{N}$ and $i \in \{1,\ldots,N\}$, there exists $F_{2Ni} \in \mathcal{F}$ with $\mathcal{P}(F_{2Ni}) = 1$ such that for each $\omega \in F_{2Ni}$, $l(Z_{-i}^{N-1}(\omega);p)$ solves

$$\min_{w\in\mathbf{W}^p}(N-1)^{-1}\sum_{1\le i\le N, i\ne k}(Y_i(\omega) - f^p(X_i(\omega),w))^2. \tag{E.16}$$

Let $F_2 \equiv \cap_{N\in\mathbf{N}}\cap_{i=1}^N F_{2Ni}$ and $F_0 \equiv F_1 \cap F_2$. Then $\mathcal{P}(F_0) = 1$. Now we show that for each $\omega \in F_0$,

$$\max_{i\in(1,\ldots,N)}\inf_{w^*\in\mathbf{W}^{p*}}\left\|l_{N-1}\left(Z_{-i}^N(\omega);p\right) - w^*\right\| \to 0 \text{ as } N \to \infty.$$

Suppose that this does not hold for some $\omega_0 \in F_0$. Then there must exist a constant $M > 0$, a strictly increasing sequence $\{N_j\}_{j\in\mathbf{N}}$ in $\mathbf{N}$ and a sequence $\{k_j\}_{j\in\mathbf{N}}$ in $\mathbf{N}$ such that $1 \le k_j \le N_j$ and

$$\inf_{w^*\in\mathbf{W}^{p*}}\left\|l_{N_j-1}\left(Z_{-k_j}^{N_j}(\omega_0);p\right) - w^*\right\| \ge M$$

for any $j \in \mathbf{N}$. Let

$$\eta_c = \{w \in \mathbf{W}^p : \|w - w^*\| \geq M \text{ for any } w^* \in \mathbf{W}^{p*}\}.$$

Then $l_{N_j-1}\left(Z_{-k_j}^{N_j}(\omega_0); p\right) \in \eta^c$. Obviously η_c is compact and $\lambda(\cdot; p)$ attains a minimum on η_c, i.e., there exists $w^c \in \eta^c$ such that $\lambda(w^c; p) \leq \lambda(w; p)$ for any $w \in \eta_c$. Because $w^c \notin \mathbf{W}^{p*}$, for any $w^* \in \mathbf{W}^{p*}$,

$$\epsilon \equiv \lambda(w^c; p) - \lambda(w^*; p) > 0.$$

Thus

$$\lambda\left(l_{N_j-1}\left(Z_{-k_j}^{N_j}(\omega_0); p\right); p\right) - \lambda(w^*; p) \geq \epsilon > 0 \tag{E.17}$$

for any $j \in \mathbf{N}$ and $w^* \in \mathbf{W}^{p*}$.

On the other hand, it follows from equation (E.15) that there exists $j_0 \in \mathbf{N}$ such that for each $j > j_0$

$$\left|(N_j-1)^{-1} \sum_{1 \leq i \leq N_j, i \neq k} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N_j-1}\left(Z_{-k_j}^{N_j-1}(\omega_0); p\right)\right)\right)^2 - \lambda\left(l_{N_j-1}\left(Z_{-k_j}^{N_j-1}(\omega_0); p\right); p\right)\right| < \frac{\epsilon}{2},$$

and

$$\left|(N_j-1)^{-1} \sum_{1 \leq i \leq N_j, i \neq k} (Y_i(\omega) - f^p(X_i(\omega), w^*)))^2 - \lambda(w^*; p)\right| < \frac{\epsilon}{2}.$$

These two inequalities and inequality (E.17) imply that for any $j \geq j_0$,

$$\begin{aligned}
&(N_j-1)^{-1} \sum_{1 \leq i \leq N_j, i \neq k} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N_j-1}\left(Z_{-k_j}^{N_j-1}(\omega_0); p\right)\right)\right)^2 \\
&\quad -(N_j-1)^{-1} \sum_{1 \leq i \leq N_j, i \neq k} (Y_i(\omega) - f^p(X_i(\omega), w^*))^2 \\
= \;& \left\{(N_j-1)^{-1} \sum_{1 \leq i \leq N_j, i \neq k} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N_j-1}\left(Z_{-k_j}^{N_j-1}(\omega_0); p\right)\right)\right)^2\right\} \\
&\quad -\lambda\left(l_{N_j-1}\left(Z_{-k_j}^{N_j-1}(\omega_0); p\right); p\right) \\
&\quad +\left\{\lambda\left(l_{N_j-1}\left(Z_{-k_j}^{N_j-1}(\omega_0); p\right); p\right) - \lambda(w^*; p)\right\} \\
&\quad +\left\{\lambda(w^*; p) - (N_j-1)^{-1} \sum_{1 \leq i \leq N_j, i \neq k} (Y_i(\omega) - f^p(X_i(\omega), w^*))^2\right\} \\
> \;& -\frac{\epsilon}{2} + \epsilon - \frac{\epsilon}{2} = 0.
\end{aligned}$$

Therefore we have

$$\begin{aligned}
&(N_j-1)^{-1} \sum_{1 \leq i \leq N_j-1, i \neq k_j} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N_j-1}\left(Z_{-k_j}^{N_j-1}(\omega_0); p\right)\right)\right)^2 \\
&\quad > (N_j-1)^{-1} \sum_{1 \leq i \leq N_j, i \neq k_j} (Y_i(\omega) - f^p(X_i(\omega), w^*))^2.
\end{aligned}$$

for all $j > j_0$. But this cannot happen because $\omega_0 \in F_0 \subset F_2 \subset F_{2N_jK_j}$ for each $j \in \mathbf{N}$. The desired result therefore must hold. $\square$

Proof of Theorem 3 Let F_0 be the same set as in the proof of Theorem 2. Then for each $\omega \in F_0$, it holds that for each $N \in \mathbf{N}$, $k \in (1, \ldots, N)$,

$$\sum_{1 \le i \le N, i \ne k} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N-1}\left(Z^N_{-k}(\omega); p\right)\right)\right)^2$$

$$\le$$

$$\sum_{1 \le i \le N, i \ne k} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_N\left(Z^N(\omega); p\right)\right)\right)^2$$

and

$$\sum_{i=1}^{N} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N-1}\left(Z^N_{-k}(\omega); p\right)\right)\right)^2$$

$$\ge$$

$$\sum_{i=1}^{N} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_N\left(Z^N(\omega); p\right)\right)\right)^2.$$

These inequalities imply that for each $\omega \in F_0$, $N \in \mathbf{N}$ and $k \in (1, \ldots, N)$,

$$\begin{aligned}
&\left(Y_k(\omega) - f^p\left(X_k(\omega), l_{N-1}\left(Z^N_{-k}(\omega); p\right)\right)\right)^2 \\
= \ & \sum_{i=1}^{N} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N-1}\left(Z^N_{-k}(\omega); p\right)\right)\right)^2 \\
& - \sum_{1 \le i \le N, i \ne k} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N-1}\left(Z^N_{-k}(\omega); p\right)\right)\right)^2 \\
\ge \ & \sum_{i=1}^{N} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_N\left(Z^N(\omega); p\right)\right)\right)^2 \\
& - \sum_{1 \le i \le N, i \ne k} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_N\left(Z^N(\omega); p\right)\right)\right)^2 \\
= \ & \left(Y_k(\omega) - f^p\left(X_k(\omega), l_N\left(Z^N(\omega); p\right)\right)\right)^2.
\end{aligned}$$

It follows that for each $\omega \in F_0$, $N \in \mathbf{N}$

$$C_N\left(Z^N(\omega); p\right) \ge \sum_{i=1}^{N} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_N\left(Z^N(\omega); p\right)\right)\right)^2.$$

Thus it holds that for each $\omega \in F_0$

$$\liminf_{N \to \infty} C_N\left(Z^N(\omega); p\right) \ge \liminf_{N \to \infty} \sum_{i=1}^{N} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_N\left(Z^N(\omega); p\right)\right)\right)^2,$$

and we know that the RHS of this inequality is $\min_{w \in \mathbf{W}^p} \lambda(w; p)$ by equation (E.14) and the fact that

$$d\left(l_N\left(Z^N(\omega); p\right), \mathbf{W}^{p*}\right) = d\left(l_N\left(Z^{N+1}_{-(N+1)}(\omega); p\right), \mathbf{W}^{p*}\right) \to 0 \text{ as } N \to \infty$$

which we have established in Theorem 2. $\square$

Proof of Theorem 4 Let w^* be the unique maximum assumed in Assumption 6. Then

$$\left\| l_N\left(Z^N;p\right) - w^*\right\| = \left\| l_N\left(Z^{N+1}_{-(N+1)};p\right) - w^*\right\| \to 0 \text{ a.s.-}\mathcal{P} \text{ as } N \to \infty.$$

by Theorem 2. Because $\lambda(\cdot;p)$ is bounded on $\mathbf{W}^p$, it follows from the Lebesgue dominated convergence theorem that

$$\begin{aligned}\lim_{N\to\infty} E\left(Q_N(Z^N;p)\right) &= \lim_{N\to\infty} E\left[\lambda\left(l_N\left(Z^N;p\right);p\right)\right] \\ &= E\left[\lim_{N\to\infty}\lambda\left(l_N\left(Z^N;p\right);p\right)\right] = E\left[\lambda(w^*;p)\right] = \lambda(w^*;p).\end{aligned}$$

Thus it suffices to show that

$$\left|C_N\left(Z^N;p\right) - \lambda(w^*;p)\right| \to 0 \text{ a.s.-}\mathcal{P} \text{ as } N \to \infty.$$

We further know that

$$\left|N^{-1}\sum_{i=1}^{N}(Y_i - f^p(X_i,w^*))^2 - \lambda(w^*;p)\right| \to 0 \text{ a.s.-}\mathcal{P} \text{ as } N \to \infty.$$

Therefore it is sufficient to show that

$$\left|N^{-1}\sum_{i=1}^{N}\left(Y_i - f^p\left(X_i, l_{N-1}\left(Z^N_{-i};p\right)\right)\right)^2 - N^{-1}\sum_{i=1}^{N}(Y_i - f^p(X_i,w^*))^2\right| \to 0,$$

a.s.-$\mathcal{P}$ as $N \to \infty$.

Define a correspondence $\bar{B}$ mapping each point in $[0,1]$ to a subset of $\mathbf{W}^p$ by

$$\bar{B}(e) \equiv \{w \in \mathbf{W}^p : \|w - w^*\| \le e\}$$

for each $e \in [0,1]$. Then it is easy to verify that $\bar{B}(\cdot)$ is a compact-valued continuous correspondence, namely, $\bar{B}(e)$ is a compact subset of $\mathbf{W}^p$ for each $e \in [0,1]$ and $\bar{B}(\cdot)$ is upper and lower hemi-continuous at every $e \in [0,1]$. Also define $V : [0,1]\times\mathbf{X}\times\mathbf{Y} \to [0,\infty)$ by

$$V(e,x,y) \equiv \max_{w\in\bar{B}(e)}\left|(y - f^p(x,w))^2 - (y - f^p(x,w^*))^2\right|$$

for each $e \in [0,1]$, $x \in \mathbf{X}$ and $y \in \mathbf{Y}$.

By Lemma 2 of Jennrich [52] (See also White [112, 2.11]) and Lemma 13.2.3 of Klein and Thompson [60], $V(e,X_i,Y_i) : \Omega \to [0,\infty)$ is measurable for each $i \in \mathbf{N}$ and $e \in [0,1]$. We can also verify by Assumption 4 and the Theorem of Maximum that $V(\cdot,X_i,Y_i) : [0,1] \to$ is continuous a.s.-$\mathcal{P}$. In addition, we have

$$0 \le V(e,x,y) \le 2(D(x,y))^2$$

for each $x \in \mathbf{X}$ and $y \in \mathbf{Y}$ with

$$E\left[(D(X,Y))^2\right] < \infty.$$

Given Assumptions 2 and 5, these properties allow us to apply Jennrich's ULLN (Jennrich [52, Theorem 2]) to $\{V(e, X_i, Y_i)\}$. Thus $\bar{V} : [0,1] \to [0,\infty)$ defined by

$$\bar{V}(e) \equiv E\left[V(e, X, Y)\right]$$

for each $e \in [0,1]$ is continuous on $[0,1]$ and there exists $F_1 \in \mathcal{F}$ with $\mathcal{P}(F_1) = 1$ such that for each $\omega \in F_1$

$$\sup_{e\in[0,1]} \left| N^{-1} \sum_{i=1}^{N} V(e, X_i(\omega), Y_i(\omega)) - \bar{V}(e) \right| \to 0.$$

In Theorem 2, we have established that there exists $F_2 \in \mathcal{F}$ with $\mathcal{P}(F_2) = 1$ such that for each $\omega \in F_2$,

$$\max_{i\in(1,\ldots,N)} \left\| l_{N-1}\left(Z_{-i}^N(\omega); p\right) - w^* \right\| \to 0.$$

Now let $F_0 \equiv F_1 \cap F_2$; pick an arbitrary point $\epsilon \in (0,\infty)$ and an arbitrary point $\omega \in F_0$. Then the following properties follow from what we have established.

- There exists $\delta_\epsilon \in (0,1]$ such that for any $e \in [0, \delta_\epsilon)$,

$$\bar{V}(e) = \bar{V}(e) - \bar{V}(0) < \frac{\epsilon}{2};$$

- There exists $N_1 \in \mathbf{N}$ such that for any $N \geq N_1$,

$$\sup_{e\in[0,1]} \left| N^{-1} \sum_{i=1}^{N} V(e, X_i(\omega), Y_i(\omega)) - \bar{V}(e) \right| < \frac{\epsilon}{2};$$

- There exists $N_2 \in \mathbf{N}$ such that for any $N \geq N_2$,

$$\max_{i\in(1,\ldots,N)} \left\| l_{N-1}\left(Z_{-i}^N(\omega); p\right) - w^* \right\| < \frac{\delta_\epsilon}{2}.$$

Therefore, for any $N \geq N_0 \equiv \max\{N_1, N_2\}$,

$$\begin{aligned}
&\left| N^{-1} \sum_{i=1}^{N} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N-1}\left(Z_{-i}^N(\omega); p\right)\right)\right)^2 - N^{-1} \sum_{i=1}^{N} \left(Y_i(\omega) - f^p(X_i(\omega), w^*)\right)^2 \right| \\
\leq\ & N^{-1} \sum_{i=1}^{N} \left| \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N-1}\left(Z_{-i}^N(\omega); p\right)\right)\right)^2 - \left(Y_i(\omega) - f^p(X_i(\omega), w^*)\right)^2 \right| \\
\leq\ & N^{-1} \sum_{i=1}^{N} V\left(\frac{\delta_\epsilon}{2}, X_i(\omega), Y_i(\omega)\right) \\
\leq\ & \left| N^{-1} \sum_{i=1}^{N} V\left(\frac{\delta_\epsilon}{2}, X_i(\omega), Y_i(\omega)\right) - \bar{V}\left(\frac{\delta_\epsilon}{2}\right) \right| + \left| \bar{V}\left(\frac{\delta_\epsilon}{2}\right) \right| \\
<\ & \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.
\end{aligned}$$

Thus, for any $N \geq N_0$,

$$\left| N^{-1} \sum_{i=1}^{N} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N-1}\left(Z_{-i}^N(\omega); p\right)\right)\right)^2 - N^{-1} \sum_{i=1}^{N} \left(Y_i(\omega) - f^p(X_i(\omega), w^*)\right)^2 \right|$$

$$< \epsilon.$$

Because ϵ is an arbitrary strictly positive real number, this implies that

$$\left| N^{-1} \sum_{i=1}^{N} \left(Y_i(\omega) - f^p\left(X_i(\omega), l_{N-1}\left(Z_{-i}^N(\omega); p\right)\right)\right)^2 - N^{-1} \sum_{i=1}^{N} \left(Y_i(\omega) - f^p(X_i(\omega), w^*)\right)^2 \right|$$

$$\to 0$$

as $N \to \infty$. Because this holds for all $\omega \in F_0$ and $\mathcal{P}(F_0) = 1$, we conclude that

$$\left| N^{-1} \sum_{i=1}^{N} \left(Y_i - f^p\left(X_i, l_{N-1}\left(Z_{-i}^N; p\right)\right)\right)^2 - N^{-1} \sum_{i=1}^{N} \left(Y_i - f^p(X_i, w^*)\right)^2 \right| \to 0 \text{ a.s.-}\mathcal{P}$$

as $N \to \infty$. $\square$

Bibliography

[1] Abarbanel, Henry D. I., Reggie Brown, John J. Sidorowich, and Lev Sh. Tsimring. (1993). "The Analysis of Observed Chaotic Data in Physical Systems." *Reviews of Modern Physics.* October, 1993.

[2] Abu-Mostafa, Yaser S. (1989). "The Vapnik-Chervonenkis Dimension." *Neural Computation*, **1** 3, 312-317, MIT press, Cambridge MA.

[3] Ahmad, Subutai, and Stephen Omohundro. (1990). "A network for extracting the locations of point clusters using selective attention." *12th Annual Conference of the Cognitive Science Society.* Also available as: *TR# 90-011, International Computer Science Institute, University of California, Berkeley.*

[4] Albertini, Francesca, Eduardo D. Sontag, and Vincent Maillot. "Uniqueness of weights for neural networks." (1993). To appear in *Artificial Neural Networks with Applications in Speech and Vision* (R. Mammone, ed.), Chapman and Hall, London.

[5] Andrews, Donald W.K. (1991). "Asymptotic optimality of generalized C_L, cross-validation, and generalized cross-validation in regression with heteroskedastic errors." *Journal of Econometrics.* **47** (1991) 359-377. North-Holland.

[6] Ash, A., and A. Hedayat. (1978). "An introduction to design optimality with an overview of the literature." *Communications in Statistics, Part A - Theory and Methods*, 7, 1295-1325.

[7] Bailey, Timothy L. and Charles Elkan. (1993). "Estimating the accuracy of learned concepts." *To appear in Proc. International Joint Conference on Artificial Intelligence.*

[8] Baum, E.B. (1991). Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Transactions on Neural Networks.* **2** 1, 5-19.

[9] Baum, Eric and David Haussler. (1989). "What size network gives valid generalization." In Touretzky, D. (ed.), *Advances in Neural Information Processing Systems 1*, San Mateo, CA: Morgan Kaufmann Publishers.

[10] Baum, Eric and David Haussler. (1989). "What size network gives valid generalization." *Neural Computation*, **1** 1, 151-160. MIT press, Cambridge MA.

[11] Barron, Andrew R. (1989). "Statistical properties of artificial neural networks." *Proceedings of the 28th Conference on Decision and Control*, 280-285.

[12] Barron, Andrew R. (1991). "Universal approximation bounds for superpositions of a sigmoidal function." Proc. IEEE International Symposium on Information Theory, Budapest, June, 1991.

[13] Barron, Andrew R. (1991). "Complexity regularization with application to artificial neural networks." *Proceedings of the NATO ASI on Nonparametric Functional Estimation.* Dordrecht, The Netherlands: Kluwer, 1990.

[14] Barron, Andrew R., and Thomas M. Cover. (1991). "Minimum complexity density estimation." Submitted to the Transactions on Information Theory.

[15] Bartle, Robert G. (1966). **The Elements of Integration.** Wiley, New York.

[16] Billingsley. (1986) **Probability and Measure.** Wiley, New York.

[17] Blum, Avrim, and Ronald L. Rivest, (1988). "Training a 3-Node Neural Network is NP-Complete." *Computational Learning Theory (COLT).* MIT, August, 1988. Editors: Haussler (UCSC) and Pitt (Illinois). Sponsored by ACM SIGACT/SIGART, UC Santa Cruz, 1988.

[18] Bowman, Adrian W. (1980). "A note on consistency of the kernel method for the analysis of categorical data." *Biometrika* (1980), **67**, 3, pp. 682-4.

[19] Bowman, Adrian W. (1984). "An alternative method of cross-validation for the smoothing of density estimates." *Biometrika* (1984), **71**, 2, pp. 353-60.

[20] Bowman, Adrian W., Peter Hall, D.M. Titterington. (1984). "Cross-validation in nonparametric estimation of probabilities and probability densities." *Biometrika* (1984), **71**, 2, pp. 341-51.

[21] Box,G., and N.Draper. (1987). **Empirical Model-Building and Response Surfaces.** Wiley, New York.

[22] Box, G.E.P. and Tiao, G.C. (1968). "A Bayesian approach to some outlier problem". *Biometrika*, **55**, 119-129.

[23] Buntine, W.L, and A.S. Weigend. (1991). "Bayesian Back-Propagation." *Complex Systems.* **5**, 1, pp. 603-643.

[24] Cachin, Christian. (1994). "Pedagogical pattern selection strategies." *Neural Networks.* **7**, 1, pp.175-181.

[25] Chow, Y.-S. and S.Geman, L.-D. Wu. (1983). "Consistent cross-validated density estimation." *The Annals of Statistics.* **11**, 1, 25-38.

[26] Cohn, David. (1990). "A local approach to optimal queries." *Connectionist Models Summer School (CMSS-90).* Proc. of the 1990 Summer School in San Diego. David S. Touretzky, et al., ed. Morgan Kaufmann, San Mateo, California.

[27] Cohn, David, Les Atlas, and Richard Ladner. (1990). "Training connectionist networks with queries and selective sampling." *Advances in Neural Information Processing Systems 2,* Proc. of the Neural Information Processing Systems Conference. Morgan Kaufmann, San Mateo, California.

[28] Cohn, David. (1994) "Neural Network Exploration Using Optimal Experiment Design." In Cowan, J.D., Tesauro, G., and Alspector, J. (eds.), *Advances in neural information processing systems 6* San Mateo, CA: Morgan Kaufmann Publishers.

[29] Cooray-Wijesinha, M., and A.I. Khuri. (1987). "The sequential generation of multiresponse D-optimal designs when the variance-covarance matrix is not known." *Comm. Statistics - Simulation*, 16(1), 239-259.

[30] Cottrell, Garrison W., and Paul Munro, David Zipser. (1989). "Image compression by back propagation: an example of extensional programming." Chapter 9, pp. 208-240, in **Models of cognition: a review of cognitive science.** Noel E. Sharkey, ed. vol.1. Ablex Publishing Corporation. Norwood New Jersey.

[31] Cover, Thomas M. (1965). "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition" *IEEE Transactions on Electronic Computers*, EC-14, 326-334.

[32] DeMers, David, and Garrison Cottrell. (1993). "Non-Linear Dimensionality Reduction." In Giles, C.L., Hanson, S.J., and Cowan, J.D. (eds.), *Advances in Neural Information Processing Systems 5*, pp. 580-587. San Mateo, CA: Morgan Kaufmann Publishers.

[33] Ehrenfeucht, Andrzej, and David Haussler, Michael Kearns, Leslie Valiant. (1988). "A general lower bound on the number of examples need for learning."

[34] Elman, J. (1990). "Finding structure in time." *Cognitive Science.*

[35] Faraway, Julian J. (1990). "Sequential design for the nonparametric regression of curves and surfaces." Technical Report #177, Department of Statistics, The University of Michigan.

[36] Farmer, Doyne, and J. Sidorowich. (1987). "Predicting chaotic time series." *Physical Review Letters* **59**, 845-848.

[37] Fedorov, V.V. (1972) **Theory of Optimal Experiments**. Academic Press, New York.

[38] Folland, Gerald B. (1984). **Real Analysis**. Wiley. New York.

[39] Funahashi, Ken-ichi. (1989). "On the approximate realization of continuous mappings by neural networks." *Neural Networks.* vol.2, no.3, pp.183-192. Pergamon Press, New York.

[40] Gasarch, W.I. and M.B. Pleszkoch. (1989). "Learning via queries to an oracle." *Proc. COLT'89 (Second Annual Workshop on Computational Learning Theory.)*

[41] Geman, Stuart, Elie Bienenstock, and René Doursat. (1992). "Neural networks and the bias/variance dilemma." *Neural Computation.* **4**, 1, 1-58.

[42] Guevara, M.R., Glass, L., Mackey, M.C., Shrier, A. (1976) "Chaos in Neurobiology." *IEEE Transactions on Systems, Man, and Cybernetics.* 13, 790-798.

[43] Hall, Peter. (1983). "Large sample optimality of least squares cross-validation in density estimation." *The Annals of Statistics.* **11**, 4, 1156-1174.

[44] Haussler, David. (1990). "Decision theoretic generalizations of the PAC model for neural net and other learning applications." Technical report UCSC-CRL-91-02, Baskin Center for Computer Engineering and Information Sciences, University of California, Santa Cruz, California.

[45] Haussler, David., M.J. Kearns, and R. E. Schapire. (1991) "Bounds on the sample complexity of bayesian learning using information theory and the VC dimension." *Proc. Neural Networks for Computing.* Snowbird, Utah.

[46] Hellerstein, Lisa, and Marek Karpinski. (1989). "Learning read-once formulas using membership queries." *Proc. COLT'89 (Second Annual Workshop on Computational Learning Theory.)*

[47] Hinton, G. (1991). personal communication.

[48] Hornik, Kurt, and M. Stinchcombe, H. White. (1989). "Multilayer feedforward networks are universal approximators." *Neural Networks.* **2** 5, pp.359-366. Pergamon Press, New York.

[49] Huber, Peter J. (1977). **Robust Statistical Procedures**. John Wiley and Sons.

[50] Hwang, Jenq-Neng, J.J. Choi, Seho Oh, and Robert J. Marks II. (1990). "Query learning based on boundary search and gradient computation of trained multilayer perceptrons. " *Proc. IJCNN 1990, San Diego. The International Joint Conference on Neural Networks.* IEEE press.

[51] Hwang, Jenq-Neng, J.J. Choi, Seho Oh, and Robert J. Marks II. (1991). "Query-based learning applied to partially trained multilayer perceptrons. " *IEEE Transactions on Neural Networks.* **2** 1, 131-136. January, 1991. IEEE press.

[52] Jennrich, R. (1969). "Asymptotic properties of nonlinear least squares estimators." *Ann. Math. Stat.* **40**, 633-643.

[53] Jolliffe, I.T. (1986). **Principal Component Analysis.** Springer-Verlag, New York.

[54] Judd, Stephen, (1988). "On the Complexity of Loading Shallow Neural Networks." *Journal of Complexity*, **4**, pp.177-192.

[55] Judd, Stephen, (1989). "Learning in Neural Networks (extended abstract)." *Computational Learning Theory (COLT)*, MIT, August, 1988. Editors: Haussler (UCSC) and Pitt (Illinois). Sponsored by ACM SIGACT/SIGART, UC Santa Cruz, 1989.

[56] Kashyap, R.L., and S. Maiyuran. (1993). "Robust regression and outlier set estimation using likelihood reasoning." Technical Report TR-EE 93-8. February 1993. School of Electrical Engineering. Purdue University. West Lafayette, Indiana 47907-1285.

[57] Kearns, Michael J., and Robert E. Schapire. (1994). "Efficient distribution-free learning of probabilistic concepts." Appears in **Computational Learning Theory and Natural Learning Systems: Constraints and Prospects**. MIT/BRADFORD 1994. Editors, S. Hanson, G. Drastal, R. Rivest.

[58] Kennel M.B., Brown, R., and Abarbanel H.D.I. (1992). "Determining embedding dimension for phase-space reconstruction using a geometrical construction", *Physical Reviews A* **45**, 6, p. 3403.

[59] Khuri, A.I., and J.A.Cornell. (1987). **Response Surfaces (Designs and Analyses)**. Marcel Dekker, Inc., New York.

[60] Klein, Erwin and Anthony C. Thompson. (1984) **Theory of correspondences : including applications to mathematical economics.** Wiley.

[61] Kurtz, S.A. and C.H. Smith. (1989). "On the role of search for learning." *Proc. COLT'89 (Second Annual Workshop on Computational Learning Theory.)*

[62] Lapedes, Alan, and Robert Farber. (1987). "Nonlinear signal processing using neural networks. Prediction and system modelling." Los Alamos technical report LA-UR-87-2662.

[63] Lapedes, A. and R. Farber. (1988) "How neural nets work." In *Neural Information Processing Systems*, (Denver 1987), ed. D.Z. Anderson. 442-456. New York: American Institute of Physics.

[64] Levin, Esther, N. Tishby, and S.A. Solla. (1989) "A statistical approach to learning and generalization." *Proc. COLT'89 (Second Annual Workshop on Computational Learning Theory.)*

[65] Lewis, David D., and William A. Gale. (1994). Proc. of ACM SIGIR '94.

[66] Li, Ker-Chau. (1986). "Asymptotic optimality of C_L and generalized cross-validation in ridge regression with application to spline smoothing." *The Annals of Statistics.* **14**, 3, 1101-1112.

[67] Li, Ker-Chau. (1987). "Asymptotic optimality for C_p, C_L, cross-validation, and generalized cross-validation: discrete index set." *The Annals of Statistics.* **15**, 3, 958-975.

[68] Liu, Yong. (1993). "Neural network model selection using asymptotic jackknife estimator and cross-validation method." In Giles, C.L., Hanson, S.J., and Cowan, J.D. (eds.), *Advances in neural information processing systems 5*, San Mateo, CA: Morgan Kaufmann Publishers.

[69] Liu, Yong. (1994). Communication broadcast over the *connectionists* electronic mailing list.

[70] Liu, Y.(1994). "Robust Parameter Estimation and Model Selection for Neural Network Regression." In Jack D. Cowan, Gerald Tesauro and Joshua Alspector (editors) *Advances in neural information processing system*, volume 6. Morgan Kaufmann, San Mateo, CA.

[71] Maass, Wolfgang. 1994. "Neural nets with superlinear vc-dimension." Extended abstract appears in *Proc. of the Int. Conf. on Artificial Neural Nets*, Sorrento (Italy), 1994. To appear in *Neural Computation.*

[72] MacKay, D.J.C., (1992). "The evidence framework applied to classification networks." *Neural Computation.* **4**, 720-736. MIT Press.

[73] MacKay, D.J.C., (1992). "Information-based objective functions for active data selection." *Neural Computation.* **4**, 590-604. MIT Press.

[74] Mackey, M.C., and L. Glass. (1977) "Oscillation and chaos in physiological control systems." *Science* **197**, 287.

[75] Marron, M. (1987). "A comparison of cross-validation techniques in density estimation." *The Annals of Statistics.* **15**, 1, 152-162.

[76] McCaffrey, D.F., and A.R. Gallant. (1994). "Convergence rates for single hidden layer feedforward networks." *Neural Networks.* **7**, 1, pp.147-158.

[77] Møller, Martin. (1993). "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", *Neural Networks*, **6**, 4, pp. 525-533.

[78] Møller, Martin. (1993). "Supervised Learning on Large Redundant Training sets", *International Journal of Neural Systems*, **4**, 1, pp. 15-25.

[79] Møller, Martin. (1993). "Efficient training of feed-forward neural networks." (1993). Doctoral dissertation. Computer Science Department. Aarhus University. Ny Munkegada, Building 540. DK-8000 Aarhus C, Denmark. ISSN 0105-8517. DAIMI PB-464.

[80] Moody, John E. (1992). "The effective number of parameters, an analysis of generalization and regularization in nonlinear learning system." In Moody, J.E., Hanson, S.J., and Lippmann, R.P., (eds.), *Advances in neural information processing systems 4*, San Mateo, CA: Morgan Kaufmann Publishers.

[81] Müller, Hans-Georg. (1984). "Optimal designs for nonparametric kernel regression." *Statistics and Probability Letters 2.* pp.285-290.

[82] Munro, Paul W. (1992). "Repeat until bored: A pattern selection strategy." In J.E. Moody, S.J. Hanson, and R.P. Lippman (Eds.), *Advances in neural information processing systems* **4**, pp. 1001-1008. San Mateo, CA: Morgan Kaufmann Publishers.

[83] Myers, Raymond H., and A.I. Khuri, W.H. Carter, Jr. (1989). "Response Surface Methodology: 1966-1988." *Technometrics.* **31** 2.

[84] Office of Graduate Studies and Research. *Instructions For Preparation and Submission of Doctoral Dissertations and Masters' Theses.* University of California, San Diego. (1991).

[85] Platt, John. (1991). "A Resource-Allocating Neural Network for Function Interpolation." *Neural Computation.* **3**, 2, 213-225.

[86] Plutowski, Mark E., and Halbert White. (1991) "Active selection of training examples for network learning in noiseless environments." *Technical Report No. CS91-180,* Department of Computer Science and Engineering, The University of California, San Diego. La Jolla, California.

[87] Plutowski, Mark E., and Halbert White. (1993). "Selecting concise training sets from clean data." *IEEE Transactions on Neural Networks.* **4**, 3, pp.305-318.

[88] Plutowski, Mark E., Garrison Cottrell, and Halbert White. (1993). "Learning Mackey-Glass from 25 examples, plus or minus 2." *Advances in Neural Information Processing Systems 6.* Giles, C.L., Hanson, S.J., and Cowan, J.D. (eds.) (San Mateo, CA: Morgan Kaufmann Publishers.

[89] Press, W.H., and B.P. Flannery, S.A. Teukolsky, W.T. Vetterling. (1988). **Numerical Recipes in C.** Cambridge University Press, Cambridge.

[90] Racine, Jeff, and M.E. Plutowski. (1993). "Neural Network Modelling and Optimal Model Complexity: The IMSE Model Criterion." Submitted.

[91] Racine, Jeff, and Halbert White. (1993). "Statistical inference and neural network modelling." Manuscript.

[92] Paul Refenes. (1994). Communication broadcast over the *connectionists* electronic mailing list.

[93] Röebel, A. (1993). "The Dynamic Pattern Selection Algorithm: Effective Training and Controlled Generalization of Backpropagation Neural Networks." Technical Report 93/23. Department of Computer Science. Technical University of Berlin (Subsets of this Report will also appear in the Proceedings of the International Conference on Neural Networks, 1994.)

[94] Röebel, A. (1993). Personal communication.

[95] Schuster, E.F., and G.G. Gregory. (1981). "On the nonconsistency of maximum likelihood nonparametric density estimators". *Comp.Sci. & Statistics: Proc. 13th Symp. on the Interface.* W.F. Eddy ed. 295-298. Springer-Verlag.

[96] Seber, G.A.F, and C.J. Wild. (1989). **Nonlinear Regression.** Wiley, New York.

[97] Sejnowski, T. (1991). personal communication.

[98] Sollich, Peter. (1994). "Query construction, entropy, and generalization in neural network models." *To appear: Physical Review E.*

[99] Stone, M. (1959). "Application of a measure of information to the design and comparison of regression experiments." *Annals Math. Stat.* **30** 55-69

[100] Stone, M. (1977). "An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion." *J. Roy. Stat. Soc. Ser B*, **39**, 1, 44-47.

[101] Stone, M. (1977). "Asymptotics for and against cross-validation." *Biometrika.* **64**, 1, 29-35.

[102] Stone, M. (1974). "Cross-validatory choice and assessment of statistical predictions." *J.R. Statist. Soc. B.* **36**, 2, 111-47.

[103] Stone, Charles J. (1984) "An asymptotically optimal window selection rule for kernel density estimates." *The Annals of Statistics.* **12**, 4, 1285-1297.

[104] Sussman, H.J. (1992). "Uniqueness of the weights for minimal feedforward nets with a given input-output map." *Neural Networks* **5**, 589-593.

[105] Tsung, Fu-Sheng and Cottrell, G. (1989). A sequential adder using recurrent networks. In *Proceedings of the International Joint Conference on Neural Networks*, Washington, D.C.

[106] Utreras, Florencio I. (1987). "On generalized cross-validation for multivariate smoothing spline functions." *SIAM J. Sci. Stat. Comput.* **8**, 4, July 1987.

[107] Valiant, L.G. (1984). "A theory of the learnable." *Comm. of the ACM.* **27** 11.

[108] Wahba, Grace. (1990). **Spline Models for Observational Data**. v. 59 in the CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, PA, March 1990. Soft-cover, 169 pages, bibliography, author index. ISBN 0-89871-244-0

[109] White, Halbert. (1984). **Asymptotic Theory for Econometricians**. Academic Press.

[110] White, H. (1989). "Learning in Artificial Neural Networks: A Statistical Perspective." *Neural Computation*, **1** 4, pp.425-464. MIT Press, Cambridge, MA.

[111] White, H. (1990). "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings." *Neural Networks*, **3** 5, pp.535-549. Pergamon Press, New York.

[112] White, Halbert. (1993). **Estimation, Inference, and Specification Analysis**. *Manuscript.*

[113] Yakowitz, S., and E. Lugosi. (1990). "Random search in the presence of noise, with application to machine learning." *Society for Industrial and Applied Mathematics Journal on Scientific and Statistical Computing.* **11**, 4, pp. 702-712.

[114] Yamada, Keiji, and Garrison Cottrell. (1994) Working Paper. Department of Computer Science and Engineering. University of California, San Diego. La Jolla, California. 92093.