Supervised Learning in Probabilistic Environments

Thesis by

Malik Magdon-Ismail

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1998

(Submitted May 19, 1998)

© 1998

Malik Magdon-Ismail

All Rights Reserved

Acknowledgments

> — *A teacher affects eternity, for no one can tell where his influence stops.*
>
> –*Henry Adams*

Foremost is my gratitude for my advisor, professor Yaser Abu-Mostafa, who, through patience, encouragement, stimulation, friendship and, of course, dependable advice is clearly worthy of being called a pillar for this work. It is with fondness that I will remember my stay in the Learning Systems Group. I would also like to thank Dr. Amir Atiya for endless input, discussion and advice. Part II is the result of a collaboration with Dr. Atiya, who contributed to many stimulating discussions. The members of the Learning Systems Group have always extended their friendship and support to me. I benefited from the useful input provided by Joseph Sill, Zehra Cataltepe, Alexander Nicholson (especially in chapter 2) and Xubo Song (especially in chapter 3). Lucinda Acosta, without whom this journey would have been unbearably bumpy, simply made things work. Many thanks to Bob Freeman for friendship and unlimited technical support and to all my friends (most notably Joe, Pregs and Tasos) for the memorable distractions they provided.

I learned an inordinate amount from the many excellent professors whose classes I attended. None of this would have been possible were it not for the Institute's stimulating environment and financial support – the EE department (for my last three years), and the Physics department (for my first two years) where I initially learned a great deal – especially from professor Peter Weichman.

I appreciate the efforts of my thesis committee (professors Y. Abu-Mostafa, D. Psaltis, J. Bruck, R. McEliece, P. Weichman) who took the time to review my thesis.

The other pillar was my family (Umma, Dada, Iqram and Zainab).

Two legs are all that I need to stand.

Abstract

For a wide class of learning systems and different noise models, we bound the test performance in terms of the noise level and number of data points. We obtain $O(1/N)$ convergence to the best hypothesis, the rate of convergence depending on the noise level and target complexity with respect to the learning model. Our results can be applied to estimate the model limitation, which we illustrate in the financial markets. Changes in model limitation can be used to track changes in volatility.

We analyze regularization in generalized linear models, focusing on weight decay. For a well specified linear model, the optimal regularization parameter decreases as $O(\sigma^2/N)$. When the data is noiseless, regularization is harmful. For a misspecified linear model, the "degree" of misspecification has an effect analogous to noise. For more general learning systems we develop EXPLOVA (explanation of variance) which also enables us to derive a condition on the learning model for regularization to help. We emphasize the necessity of prior information for effective regularization.

By counting functions on a discretized grid, we develop a framework for incorporating prior knowledge about the target function into the learning process. Using this framework, we derive a direct connection between smoothness priors and Tikhonov regularization, in addition to the regularization terms implied by other priors.

We prove a No Free Lunch result for noise prediction: when the prior over target functions is uniform, the data set conveys no information about the noise distribution. We then consider using maximum likelihood to predict non-stationary noise variance in time series. Maximum likelihood leads to systematic errors that favor lower variance. We discuss the systematic correction of these errors.

We develop stochastic and deterministic techniques for density estimation based on approximating the distribution function, thus placing density estimation within the supervised learning framework. We prove consistency of the estimators and obtain convergence rates in L_1 and L_2. We also develop approaches to random variate generation based on "inverting" the density estimation procedure and based on a

control formulation.

In general, we use multilayer neural networks to illustrate our methods.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

— *The way that can be walked on is not the perfect way.*
The word that can be said is not the perfect word.
– *Lao-Tzu* ($\sim 3^{rd}$ *century B. C.*)

— *If it weren't for that infamous last minute, nothing would ever get done.*

Learning is a general concept that describes the process of formulating a *hypothesis* from a finite set of *data*. Data is presented as a set of instances drawn from some underlying process, and one attempts to determine the underlying process from the data alone[1]. Some examples include

- *Probability Density Estimation.* The instances correspond to realizations of a random variable from the true density (the underlying process). One attempts to learn the form of the true density from the realizations.

- *Pattern Recognition.* The instances are a pair {feature vector, class label}. As an example, the feature vector might be [brightness, time] and the class label might be day or night[2]. One attempts to determine whether it is day or night given the brightness and the time of day. Admittedly, this is an easy task, but not so easy, and in the same category, are such varied tasks as speech recognition, hand written digit recognition, fault detection in machinery, medical diagnosis, etc.

- *Regression.* The instances correspond to $\{\mathbf{x}, y\}$ pairs and the underlying process

[1]Sometimes other prior information is available, for example, one might know that the true dependence is a monotonically increasing function.
[2]by definition day is the time between sunrise and sunset.

which is to be learned is the mapping $y = f(\mathbf{x})$.

- *Time Series Prediction.* The data usually consists of a single instance of a time series $\mathbf{z}(t) = [x(t), \mathbf{I}(t)]$ for $t = 0, 1, \ldots, T$ and the task is to determine the value x(T+1). One can view the $\mathbf{I}(t)$ as the information available at time t.

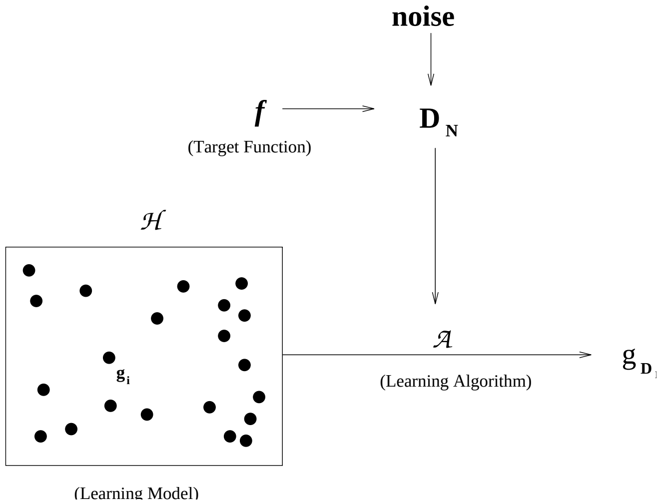
The brain appears to have solved many of these problems efficiently – in the sense that it learns patterns well. However, the brain is a slow processor – it can only process a small number of patterns per second. Can we solve these problems just as efficiently using computers, and gain the benefit of their speed at the same time?

Many computational learning systems exist (neural networks, support vector machines, Gaussian processes, radial basis networks, nearest neighbor rules ...)[3]. We will develop a theory for learning systems in general and use neural networks to illustrate some of our results. A brief description of a neural network can be found in section 1.3. The most general and natural framework in which to set the learning problem is a statistical one, taking into account the probabilistic nature of the information we learn from. To begin, let us first set up a mathematical framework for learning. We will develop a framework for supervised learning, the task of determining the true mapping from one space to another, given a set of instances of the mapping. Most learning problems, including many unsupervised learning problems (for example density estimation) can be posed within this framework.

1.1 Learning Theory Set Up

We are presented with a data set $\mathcal{D}_N$, the **training set**, which consists of N input output pairs $\{\mathbf{x}_i, y_i\}_{i=1}^N$. Each $\mathbf{x}_i \in \mathbb{R}^d$ is drawn from some input probability measure $dF(\mathbf{x})$ which we will usually assume to have compact support. We call the underlying process that we are attempting to learn the **target function**. The target function is generally represented by a conditional probability measure $dF(y|\mathbf{x})$ – each y_i is drawn

[3]A description of neural network and other learning models can be found in [Bishop, 1995a].



The learning algorithm selects from the learning model a hypothesis function $g_{\mathcal{D}_N}$ based on the data set $\mathcal{D}_N$.

Figure 1.1: The learning paradigm

from the probability measure $dF(y|\mathbf{x})$. We will restrict ourselves to the case where

$$y_i = f(\mathbf{x}_i) + \epsilon_i \tag{1.1}$$

where $f(\mathbf{x}_i)$ is some deterministic dependence and ϵ_i is a random component[4]. We would like to model the deterministic dependence $f(\mathbf{x}_i)$.

Learning entails choosing a hypothesis function g from a collection of candidate functions $\mathcal{H}$. We will generally assume that the target function f and the candidate functions $g \in \mathcal{H}$ are continuous. The set $\mathcal{H}$ is called the **learning model** because it reflects how we choose to model the target function. The hypothesis function is chosen by a **learning algorithm**, $\mathcal{A}$, usually based on some performance criterion on the data. The general learning paradigm is illustrated in figure 1.1. We assume that $\mathcal{A}$ is a mapping $\mathcal{A} : \mathcal{D}_N \to \mathcal{H}$. A typical learning algorithm might be to select the hypothesis which minimizes an error measure on the training set. This is generally called **empirical risk minimization**. For example, let $\tilde{\mathcal{E}}(x, y) = (x - y)^2$ be the

[4]For example, if ϵ_i was a normal(0,1) random variable, then $P(y|\mathbf{x}_i)$ =normal$(f(\mathbf{x}_i), 1)$.

error measure. Then, the algorithm might be

$$\mathcal{A}(\mathcal{D}_N) = \underset{g \in \mathcal{H}}{argmin} \sum_{\mathbf{x}_i \in \mathcal{D}_N} (g(\mathbf{x}_i) - y_i)^2 \tag{1.2}$$

Given a learning task, we select a particular *learning system*, which takes as input a data set and produces a hypothesis function (see figure 1.2).

Definition 1.1.1 *A* *Learning System* $\mathcal{L}$ *is a pair* $\{\mathcal{A}, \mathcal{H}\}$.

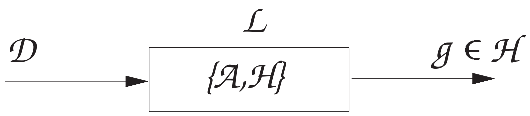


Figure 1.2: The learning setup

Define $g_{\mathcal{D}_N}(\mathbf{x}) \in \mathcal{H}$ as $\mathcal{A}(\mathcal{D}_N)$, the function that was chosen by the algorithm. Usually, we are interested in the performance of $g_{\mathcal{D}_N}(\mathbf{x})$ with respect to some error measure, $Q(g_{\mathcal{D}_N}, f)$. The **risk (test error)**, $E[g_{\mathcal{D}_N}]$, is the expected value of this error measure

$$E[g_{\mathcal{D}_N}] = \langle Q(g_{\mathcal{D}}, f) \rangle_{\mathbf{x}} \tag{1.3}$$

We use $\langle \cdot \rangle$ to denote expectations[5]. Thus, the test error measures the ultimate performance of our system after it has learned from the data. A common test error measure is the squared difference between f and g.

$$E[g_{\mathcal{D}_N}] = \left\langle (g_{\mathcal{D}_N}(\mathbf{x}) - f(\mathbf{x}))^2 \right\rangle_{\mathbf{x}} \tag{1.4}$$

We can further define the **expected risk (expected test error)**, $\mathcal{E}$ as the expecta-

[5]Sometimes $E[\cdot]$ will be used to denote expectations, but in this context, we usually reserve E for error measures.

tion of the test error taken over possible realizations of the data set.

$$\mathcal{E} = \langle E[g_{\mathcal{D}_N}] \rangle_{\mathcal{D}_N} \tag{1.5}$$

This usually involves taking an expectation over the noise and the $\mathbf{x}$ values in the data set. The goal is to minimize $\mathcal{E}$. $\mathcal{E}$ represents the expected test performance of a given learning system.

All the example learning problems mentioned earlier can be fit into this general learning paradigm.

- *Probability Density Estimation.* Let $F(i)$ be the fraction of points less than $\mathbf{x}_i$, where $\mathbf{x} < \mathbf{y}$ if the inequality is satisfied component-wise. Then we attempt to learn the mapping $\mathbf{x}_i \rightarrow F(i)$.

- *Pattern Recognition.* We attempt to learn the mapping $feature\ vector \rightarrow class\ label$.

- *Regression.* This is almost identical to the general setting.

- *Time Series Prediction.* One attempts to learn the mapping $\mathbf{I}(t-1) \rightarrow x(t)$.

1.2 Preview

Having defined the learning paradigm, certain natural questions arise. Under what conditions is it consistent – as the data set becomes larger, will one necessarily converge to a hypothesis that minimizes the expected risk? How does noise affect this convergence? Is it necessarily true that in order to minimize the expected test error, it is always best to minimize the empirical risk for the same error measure? We consider these questions in Part 1.

Intuitively one expects noise to decrease the performance of a learning system. However, the effects of noise should become less pronounced with increasing data availability. We will quantify this intuition in chapter 2 where we will show that for a very broad class of learning systems, the performance is expected to behave as[6]

$$\mathcal{E}_N(\sigma) - \mathcal{E}_0 \leq \frac{\overline{\sigma^2}\mathcal{C} + B}{N} + o\left(\frac{1}{N}\right) \tag{1.6}$$

where $\overline{\sigma^2}$ is the average noise variance and $\mathcal{E}_N(\sigma)$ is the expected test error when the data set has N examples. $\mathcal{E}_0$, B and C are constants that may depend on the target function, learning model and input distribution. $\mathcal{E}_0$ is the model limitation modulo the algorithm. This bound is valid in the non-asymptotic regime as well as for noise (ϵ_i) with different distributions. Using this result, we quantify the trade off between numerous more noisy data, and fewer less noisy data. We will also develop a method for estimating the model limitation, $\mathcal{E}_0$. We present experimental verification of (1.6) on some toy problems as well as on real financial data.

In chapter 3 we look at regularization. Specifically, we investigate the use of learning algorithms that not only minimize an empirical risk, but also penalize the complexity of the learned hypothesis by introducing a regularization term. Unfortunately, there is no well established way to determine the relative weight given to the empirical risk versus the complexity penalty – this relative weighting is usually called the regularization parameter. We extend the one-dimensional linear learning model results of [Koistinen, 1997] to the $n-$dimensional general linear framework, where we derive the form of the regularization parameter both when the learning model is well specified (the target function is in the learning model) and when the learning model is misspecified. We then develop a technique which we call EXPLOVA (explanation of variance) that provides an estimate of the regularization parameter for general learning models and regularization terms. We use EXPLOVA to derive a condition for regularization to improve test performance. We find that the regularization parameter is $\mathrm{O}(\overline{\sigma^2}/N)$. In particular, when no noise is present, regularization can hurt.

[6]We use the the standard order notation. We use $x(N) = O(y(N))$ to mean that $x(N) \leq Cy(N)$ for some constant C and for all N. We use $x(N) = o(y(N))$ to mean $x/y \to 0$ as $y(N) \to 0$ $(N \to \infty)$.

We present experimental verification of our results using neural networks.

The noise variance affects the convergence of learning systems, and plays a role in the choice of regularization parameters. Thus, we will conclude part I with an analysis of the estimation/prediction of noise variance in time series. We will analyze the commonly used maximum likelihood method and show that it is not uncommon for maximum likelihood methods to pick the worse of two models most of the time. We will show that learning models trained to predict $\sigma(t)$ using maximum likelihood will have a tendency to underpredict $\sigma(t)$. We will derive the probability distribution for λ given a sufficiently general learning model, where λ is the underprediction factor. We will then illustrate how this distribution over λ could be used to update future predictions obtained using the learned model to minimize a given error criterion.

In part II we study the density estimation problem. Existing techniques for density estimation involve developing an estimate for the probability density function itself. We present two techniques that learn to implement the sample distribution function instead. The first is a stochastic technique (SLC) and the second (SIC) estimates the density by smoothly interpolating the sample distribution function. We will show that the stochastic method (SLC) converges to a solution of the interpolation method (SIC) with probability 1. Then we analyze the convergence properties of SIC. We will define a class of estimators which we call generalized distribution functions. We will prove that the class of generalized distribution functions converges uniformly in L_1 and L_2 (MISE[7]) with respect to an arbitrary probability measure. The rate of convergence is $\mathrm{O}(1/\sqrt{N})$ for L_1 and $\mathrm{O}(1/N)$ for L_2. We will then analyze the convergence of the density estimate obtained from the distribution estimate. We will prove convergence to the true density when the true density has bounded derivatives up to order K. We will show that the rate of convergence is $\mathrm{O}((\log\log(N)/N)^{(1-1/K)})$ in L_2. This bound is better than that attainable by standard non-parametric techniques that attempt to predict the density when $K > 5$. We show experimental verification of these bounds

[7]Mean Integrated Squared Error.

on a toy problem and we use our technique to estimate the densities for the changes in stock prices. We verify the well established fact that changes in stock prices have a distribution that is fat-tailed compared to a Gaussian distribution.

We then present an application to random variate generation. One can use the same techniques for distribution function estimation to estimate the inverse of the distribution function. Given the inverse, it is easy to generate random variates according to the estimated density. This method for random variate generation is fast and works for general densities.

In part I we consider the issue of priors and how they affect learning systems. First we will add to the existing no free lunch (NFL) results by showing that constructing general algorithms to estimate noise distributions from the data is a fruitless endeavor – No Noisy Free Lunch (NNFL). Specifically we will prove that when the prior over the target function space is uniform, a well behaved prior on the noise distribution space can not be updated (in the Bayesian sense) from any finite data set. Therefore in order to make statements about noise in data, it is *necessary* to make some statement about the target function.

The issue of priors threads its way through this entire work. We will devote the final chapter to the idea of natural priors and their incorporation into the learning process in the form of regularization terms. We will pursue the purely combinatorial question of counting paths on a discretized grid. We will then use these results to develop the form of some common regularizers, notably the Tikhonov regularizers. We will develop a framework for incorporating general priors into the learning process through regularization terms, and explicitly derive these regularization terms for a set of commonly encountered priors. Thus we will explicitly link the use of Tikhonov regularizers to priors on derivatives, and we will provide a principled approach to the use of regularization terms. Our analysis also provides a method for calculating the regularization parameter.

1.3 The Neural Network

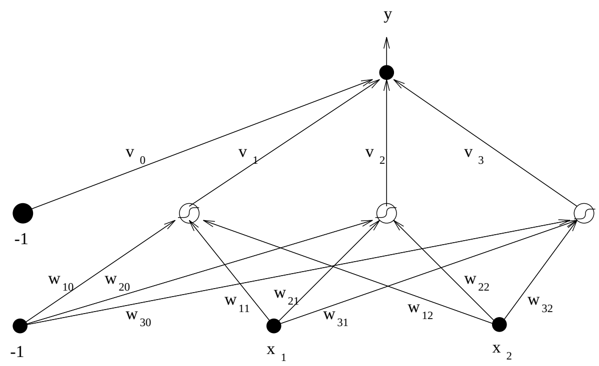


Figure 1.3: A 2-input 3-hidden unit feed forward neural network with one output.

The two layer neural network with one output unit is a parameterized learning model that implements a function of the form

$$y(\mathbf{x}) = S\left(\sum_{i=1}^{H} v_i \Theta\left(\sum_{j=1}^{d} w_{ij} x_j - w_{i0}\right) - v_0\right) \tag{1.7}$$

that maps $\mathbf{x} \in \mathbb{R}^d$ to $y \in \mathbb{R}$. H is the number of hidden units and the parameters are the w_{ij}'s and the v_k's. v_0 and the w_{i0}'s are usually called biases. $\Theta(x)$ is a non-linear function called the threshold function. $\Theta(x)$ is usually chosen to be sigmoidal (eg. tanh(x), $1/(1+e^{-x})$). $S(x)$ is called the output non-linearity (usually $S(x) = x$). The parameters (sometimes called weights) are altered in order to obtain the desired mapping. Figure 1.3 illustrates a typical neural network. Various universal approximation results show that neural networks can approximate most interesting function classes on compacta [Hornik *et al.*, 1990], [Barron, 1993], [Cybenko, 1989], [Funahashi, 1989].

Part I

Noise And Learning

Introduction

— To prove something to someone, it suffices to convince them that it is true.

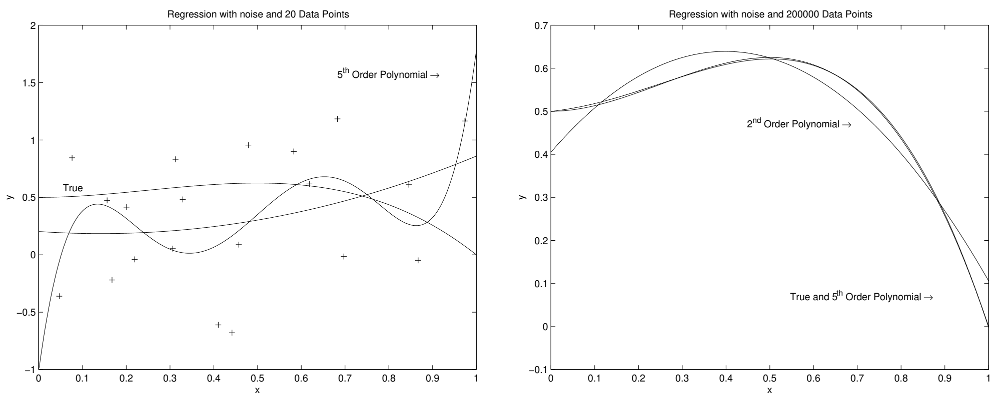
–Jehoshua Bruck

For a given learning system (learning model, learning algorithm pair), there will be a "best" hypothesis (modulo the algorithm) that can be reached. With respect to the test error measure, suppose that this hypothesis achieves an error of $\mathcal{E}_0$. It is natural to ask whether the learning system will necessarily arrive at this hypothesis as the data set size gets large. We do not really care about the hypothesis, only its test error, so it is more reasonable to require that the learning system arrive arbitrarily close to this optimal test error as the data set size becomes large. Figure 1.4 illustrates that a tradeoff exists between having a low value for $\mathcal{E}_0$ and the speed with which this value is reached as the data set size gets larger – the BIAS-VARIANCE tradeoff [Geman and Bienenstock, 1992].

Some of the issues of convergence within the PAC (Probably Approximately Correct) framework have been answered by VC-theory [Vapnik and Chervonenkis, 1971], [Vapnik, 1982], [Vapnik, 1995]. For a classification problem[8], with high probability ($> 1-\delta$) one wishes to find a classifier with low probability of error ($Pr[error] < \epsilon$). Suppose that on m examples, a hypothesis, $g \in \mathcal{H}$, has an empirical misclassification error rate (training error) of $\nu_g^m = \#\ incorrect/m$. Let e_g be the true probability of misclassification (test error). Then, VC-theory provides a bound of the form [Parrondo and Van den Broeck, 1993]

$$Pr[\sup_{g\in\mathcal{H}} \left|\nu_g^m - e_g\right| > \epsilon] < 6e^{2\epsilon}\Delta(2m)e^{-m\epsilon^2} \tag{1.8}$$

[8]A learning problem where the target outputs are ± 1. More general functions can be treated using the ϵ-net, [Vapnik, 1995, pg 41-45].



(a) (b)

(a)A fit with 2^{nd} order and 5^{th} order polynomials to 20 noisy data points generated by a 3^{rd} order polynomial. (b)A fit with 2^{nd} order and 5^{th} order polynomials to 200000 noisy data points generated by a 3^{rd} order polynomial. These two graphs illustrate that with many data points, even with noise present, both models find their respective best fits. It happens to be the case that the more powerful model can implement the target function. With few data points however the more powerful model is very sensitive to a little noise. The less powerful model, despite not being able to implement the target function, seems to do better. Thus, though it has a higher bias, for the small value of N it appears to have a better test error indicating that the weaker model converges more quickly to its optimal error.

Figure 1.4: Tradeoff between more powerful learning systems and higher sensitivity to noise

Here $\Delta(2m)$ is the growth function for the learning model which is essentially a measure of the complexity of the learning model. This inequality states that with high probability, the deviation of the training error from the test error will be less than ϵ if the model complexity ($\Delta(2m)$) is not too large. If in addition one has a good optimization algorithm for minimizing the empirical misclassification rate, then, the condition "$\Delta(2m)$ does not grow too fast" is sufficient for consistency in the PAC sense. Necessary and sufficient conditions for PAC consistency of the learning process, along with a more detailed account can be found in [Vapnik, 1995, chap. 3]. These results are a considerable step forward. They show that consistent learning is possible provided that the complexity of the learning model is controlled – the bound, (1.8), applies for *any* target function and any input distribution.

Can similar statements be made about other test error measures? For example, what properties of a learning system are sufficient to guarantee that the expected squared error of the learned hypothesis will converge to the best possible, modulo the learning system? We will address this issue in chapter 2.

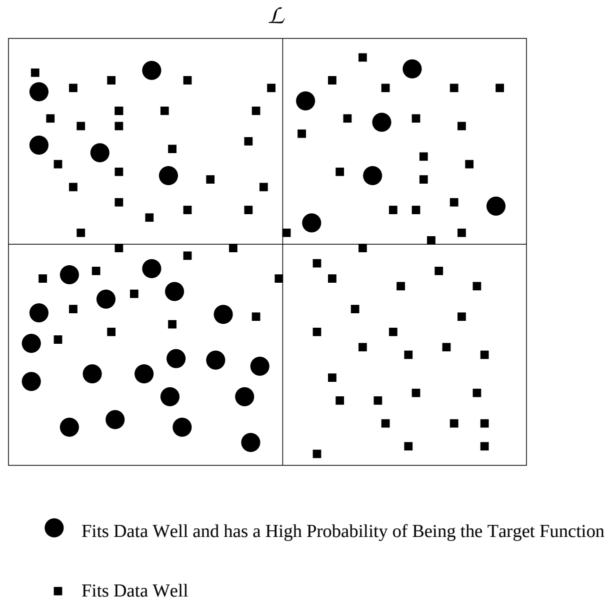
The development of the theory of ill-posed problems has led to the theory of regularization as a means for their solution. Briefly, the functional equation

$$\mathcal{I}(f) = F \tag{1.9}$$

(where f is the unknown function) is ill-posed if a small change in F ($F \to F_\delta$ with $\| F - F_\delta \|$ small) results in a large change in f ($f \to f_\delta$ with $\| f - f_\delta \|$ large). This is not desirable because the solutions to such problems can be very sensitive to the presence of noise, and, in general, it may not even be the case that in the limit $\delta \to 0$, the convergence $f_\delta \to f$ is obtained. Ill-posed problems are not a purely mathematical phenomenon. Important real life problems can be ill-posed, most notably, density estimation (which we will return to in part II), and the inference problem (inversion of the cause–effect relationship). It was discovered that if instead of minimizing an error measure $\mathcal{E}(\mathcal{I}(f), F_\delta)$, one introduces the so called regularization functional

$$\mathcal{E}(\mathcal{I}(f), F_\delta) + \lambda(\delta)\Omega(f) \tag{1.10}$$

where $\Omega(f)$ is some functional and $\lambda(\delta)$ is an appropriately chosen constant, then the desired convergence $\lim_{\delta\to 0} f_\delta \to f$ can be obtained, [Tikhonov and Arsenin, 1977], [Tikhonov, 1963], [Ivanov, 1962]. Thus, we see that it is not always best to minimize an empirical risk (training error). In the presence of noise, learning can be enhanced by using a regularization term. The need for regularization is now evident, but exactly how one finds $\lambda(\delta)$ is yet undiscussed. We will look at effective choices for $\lambda(\delta)$ in chapter 3. Figure 1.5 illustrates the general philosophy behind regularization. We



Regularization effectively restricts the search to a subset of the learning model. Shown are four possible subsets.

Figure 1.5: Illustration of the philosophy of regularization.

start with a large learning model. We use regularization to restrict the search to a subset of the learning model in order to decrease the variance of our search. If we restrict our search to the lower left corner, we are in good shape because most of the functions that are good fits are also likely to be the target function. The top two subsets are not as good. The lower right subset is disastrously bad – we may have succeeded in reducing the variance, but we have done so at expense of a large bias. We expect the regularization to be effective in proportion to how good the restricted subset is at approximating the target function. This in turn will depend on how good our prior information about the the target function is, because we use this prior information to determine which restricted subset to search. It makes no sense to restrict the search to low complexity functions if one is sure that the target function has intermediate complexity – in this case one would want a "complexity" penalty that penalizes high and low complexities. Thus one expects that effective regularization should be intimately connected with good prior information.

Chapter 2

Convergence of Learning Systems[1]

— All roads lead to Rome. The only question is "How fast do they get you there?".

2.1 Introduction

In this chapter, we will tackle the question of learning from noisy data and how learning performance is affected by the presence and variability of noise in the data. In doing so, we do not restrict the distribution or the time-varying nature of the noise, nor do we place severe restrictions on the learning model or learning algorithm that we use. Our results provide quantitative estimates of the optimal performance that can be achieved in the presence of noise. In financial markets, this provides a benchmark for the target performance given a data set. We also quantify the tradeoff between the noise level and the number of data points. Our experiments with real foreign exchange data demonstrate that the results are applicable to the case of finite data, the only case of practical interest. They also provide a means for assessing the change in the level of noise in financial data that can be applied to volatility-based financial instruments.

The chapter is organized as follows. Section 2.2 covers the main results about the impact of noise. Here we will prove convergence results for stable learning systems, and provide bounds for the test error. We present experimental verification of these results for a toy problem. These results are then tested in the four major foreign exchange markets in section 2.4. We will usually defer to appendix 2.6 the complete

[1]Parts of this chapter were done in collaboration with Alexander Nicholson of the Learning Systems Group

proofs of the results when the proofs offer no added insight.

2.2 Impact of Noise on Learning

We begin by specializing the general learning paradigm that has already been defined in chapter 1. We will define the learning systems we are interested in, which will be a fairly general class of learning systems. We will see that the bounds on the performance of these fairly general learning systems take on a surprisingly simple form.

2.2.1 The Learning Problem

We assume the standard learning paradigm. The goal is to learn a target function $f : \mathbf{R}^d \to \mathbf{R}$. The training data set, $\mathcal{D}_N$, consists of N input output pairs $\{\mathbf{x}_i, y_i\}_{i=1}^N$. Each $\mathbf{x}_i \in \mathbf{R}^d$ is drawn from some input probability measure $dF(\mathbf{x})$ which we assume to have compact support. We will assume that the target function f and the candidate functions $g \in \mathcal{H}$ are continuous. Additive noise is present in the training data, $y_i = f(\mathbf{x}_i) + \epsilon_i$. We further assume that the noise realizations are independent and zero mean, so

$$\langle \epsilon \mid \mathbf{x} \rangle_\epsilon = \mathbf{0}, \qquad \langle \epsilon \epsilon^T \mid \mathbf{x} \rangle_\epsilon = diag[\sigma_1^2, \sigma_2^2, \ldots, \sigma_N^2] \tag{2.1}$$

(we use $\langle \cdot \rangle$ to denote expectations, $\sigma = [\sigma_1 \sigma_2 \ldots \sigma_N]$, and $diag[\cdot]$ denotes a diagonal matrix). It should be noted that we allow the noise variance to change from one data point to another.

Let $g_{\mathcal{D}_N}(\mathbf{x}) \in \mathcal{H}$ be $\mathcal{A}(\mathcal{D}_N)$, the function that was chosen by the algorithm. The test error we will be interested in is the expectation of the squared deviation between $g_{\mathcal{D}_N}$ and $f(\mathbf{x})$ taken over the input space. We will denote the test error by $E[g_{\mathcal{D}_N}]$.

$$E[g_{\mathcal{D}_N}] = \left\langle (g_{\mathcal{D}_N}(\mathbf{x}) - f(\mathbf{x}))^2 \right\rangle_{\mathbf{x}} \tag{2.2}$$

The expected test error, $\mathcal{E}_N(\sigma)$, is then given by

$$\mathcal{E}_N(\sigma) = \langle E[g_{\mathcal{D}_N}] \rangle_{\epsilon, \mathcal{D}_N} \tag{2.3}$$

where we have made explicit the noise dependence of the data set in the expectation. The goal is to minimize $\mathcal{E}_N(\sigma)$. $\mathcal{E}_N(\sigma)$ will depend on the detailed properties of the learning system and target function. It would be a daunting task to tackle the behavior of $\mathcal{E}_N(\sigma)$ in general, but as we shall see, under quite unrestrictive conditions, the changes in $\mathcal{E}_N(\sigma)$ as the noise or data set size change can be quantified.

A related quantity of interest is $\mathcal{N}$, the number of data points (with noise added) that are needed to attain an expected test error comparable to that attainable when N noiseless examples are available.

$$\mathcal{N}(\Delta, \sigma, N) \triangleq \min_{N_1} \{ N_1 : \mathcal{E}_{N_1}(\sigma) - \mathcal{E}_N(0) \leq \Delta \} \tag{2.4}$$

$\mathcal{N}(\Delta, \sigma, N)$ is the number of noisy examples that are equivalent to N noiseless examples, and it describes the trade off between numerous, more volatile data, versus fewer and less volatile data. We would like to analyze the behavior of $\mathcal{E}_N(\sigma)$ and $\mathcal{N}(\Delta, \sigma, N)$. We address these questions analytically in section 2.3 where we restrict our analysis to *stable learning systems*, which we will now define.

2.2.2 Stable Learning Systems

One expects that if one has "close" data sets $\mathcal{D}_N = \{\mathbf{x}_i, f(\mathbf{x}_i)\}$ and $\mathcal{D}'_N = \{\mathbf{x}_i, f(\mathbf{x}_i) + e(\mathbf{x}_i)\}$ where $e(\mathbf{x}_i)$ is small, then $\mathcal{A}(\mathcal{D}_N)$ should be "close" to $\mathcal{A}(\mathcal{D}'_N)$. For $\mathcal{A}$ to have this property, $\mathcal{H}$ should be able to implement the two "close" functions.[2] We formalize this notion by defining the class of learning systems that are n^{th} order–*continuously compatible* ($\mathcal{CC}_n$) with respect to the probability measure $dF(\mathbf{x})$. We will use the following notation. Let $\mathcal{S}$ be the compact support for $dF(\mathbf{x})$ and let $\{\mathbf{x}_i\}_{i=1}^N \subset \mathcal{S}$.

[2]These conditions will often be satisfied in practice.

Let $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, $\mathcal{D}' = \{\mathbf{x}_i, y_i + \epsilon_i\}_{i=1}^N$ be any two data sets on $\mathcal{S}$ such that $\max_i \epsilon_i = \epsilon_{max}$. Let $\mathcal{A}(\mathcal{D}) = g(\mathbf{x})$, $\mathcal{A}(\mathcal{D}') = g(\mathbf{x}) + \eta(\mathbf{x})$.

Definition 2.2.1 *$\mathcal{L}$ is $\underline{n^{th}\text{ order–continuously compatible}}$ if $\exists C$ such that*

$$\langle |\eta(\mathbf{x})|^n \rangle_{\mathbf{x}} \leq (C\epsilon_{max})^n$$

with probability 1 (i.e. for almost every $\mathcal{D}$). We will write $\mathcal{L} \in \mathcal{CC}_n$.

Proposition 2.2.2 *If $\mathcal{L} \in \mathcal{CC}_n$ then $\mathcal{L} \in \mathcal{CC}_m$ for $m = 1 \ldots n$.*

PROOF: By Jensen's inequality, $\langle |f(\mathbf{x})|^a \rangle_{\mathbf{x}} \leq \langle |f(\mathbf{x})| \rangle_{\mathbf{x}}^a$ for $0 \leq a \leq 1$. Letting $f(\mathbf{x}) = \eta(\mathbf{x})^n$ as in Definition 2.2.1 and $a = m/n$ for $m \leq n$, the proposition now follows because $\mathcal{L} \in \mathcal{CC}_n$. ■

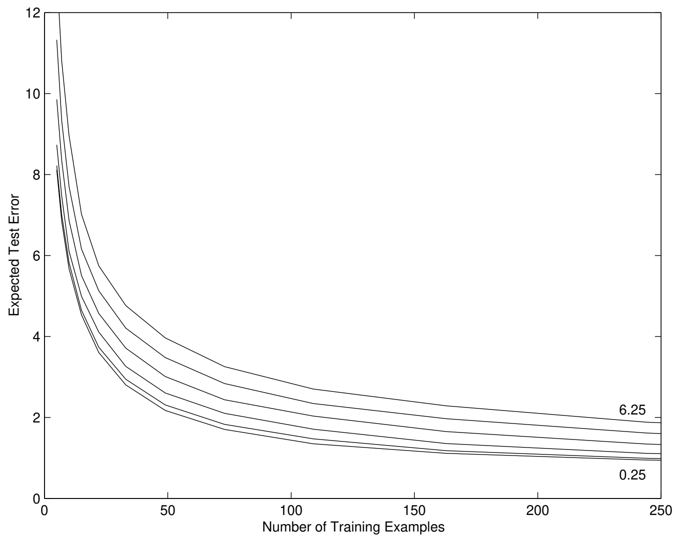
We would like $\mathcal{A}$ to be "unbiased" in the following sense. If we have a data set $\mathcal{D}$ with $\mathcal{A}(D) = g_0$ and we add independent, zero mean noise to the targets to get a new data set $\mathcal{D}'$ then we would like $\langle \mathcal{A}(\mathcal{D}') \rangle_{noise} = g_0$, where this average of functions is taken point wise. This motivates the following definition.

Definition 2.2.3 *Let $\mathcal{D}$ and $\mathcal{D}'$ be two data sets related by $\mathbf{y}' = \mathbf{y} + \epsilon$ where the ϵ_i's are independent and zero mean. Then $\mathcal{L}$ is <u>mean preserving</u> or <u>unbiased</u> if $\langle \mathcal{A}(\mathcal{D}') \rangle_\epsilon = \mathcal{A}(\mathcal{D})$.*

Definition 2.2.4 *A learning system $\mathcal{L}$ is <u>stable</u> if it is in $\mathcal{CC}_2$ and it is mean preserving.*

These properties are somewhat intuitive, and we note that, for any learning system $\mathcal{L}$, they can be checked directly. We would like our learning procedure to be robust towards small noise fluctuations in the data so we do not consider learning models that may yield discontinuous behavior. The unbiasedness property may seem fragile, especially given the extremely nonlinear nature of most learning algorithms. Nevertheless, we consider it an important and not overly restrictive condition on a learning system. If the noise is small, then the first order change in $\mathcal{A}(D_N)$ should be proportional to the noise parameter, so that the average change is zero with zero-mean noise. Indeed, experiments with neural networks show that learning with gradient descent and conjugate gradient descent on the mean squared error are unbiased with a reasonable noise level. Thus, linear and neural network learning models give learning systems that are stable.

2.3 Performance of a Learning System



For various noise levels with variances ranging from 0.25–6.25. A non-linear neural network learning model was used with gradient descent on the squared error. Data was created using a non-linear target function.

Figure 2.1: Experiments illustrating the behavior of the test error as a function of N and σ^2.

Intuition tells us that noisier data leads to worse test performance. This is because the learning system attempts to fit the noise (i.e. learn a random effect) at the expense of fitting the true underlying dependence. However, the more data we have, the less pronounced the impact of the noise will be. This intuition is illustrated in figure 2.1. We observe that the higher the noise, the higher the test error, but the curves appear to be getting closer to each other as we use more and more examples for the learning process. The following two theorems quantify this intuition. We will restrict ourselves to *stable learning systems*. Remember that stable learning systems possess the two properties "continuity" and "unbiasedness."

First we consider the case where the learning model is linear – linear regression with noise. The following theorem is valid.

Theorem 2.3.1 *Let $\mathcal{H}$, the learning model, be the set of linear functions* $\mathbf{w} \cdot \mathbf{x} + w_0$ *and let the learning algorithm be minimization of the squared error. Then*

$$\mathcal{E}_N(\sigma) = \mathcal{E}_N(0) + \frac{\overline{\sigma^2}(d+1)}{N} + O\left(\frac{1}{N^2}\right) \tag{2.5}$$

$$\mathcal{E}_N(0) = E_0 + \frac{B}{N} + O(\frac{1}{N^{\frac{3}{2}}}) \tag{2.6}$$

where $E_0 = \lim_{N\to\infty} \{\mathcal{E}_N(0)\}$ *and B is a constant dependent on the input distribution. It follows that*

$$\mathcal{N}(\Delta, \sigma, N) = \frac{\overline{\sigma^2}(d+1) + B}{\Delta + \frac{B}{N}} + O(\frac{1}{N^{\frac{3}{2}}}) \tag{2.7}$$

.

PROOF: See theorem 2.6.1 in appendix 2.6. ■

This theorem can be generalized for learning models that are general linear models (models that are linear in their parameter space) – included here would be Fourier series, polynomial regression, etc. For more general learning models, the following theorem is valid.

Theorem 2.3.2 *Let $\mathcal{L}$ be stable. Then, for any $\epsilon > 0$, $\exists \mathcal{C}_1$ such that using $\mathcal{L}$, it is at least possible to attain a test error bounded by*

$$\mathcal{E}_N(\sigma) < \mathcal{E}_N(0) + \frac{\overline{\sigma^2}\mathcal{C}_1}{N} + \epsilon + O\left(\frac{1}{N^2}\right) \tag{2.8}$$

$$\mathcal{E}_N(0) < E_0 + \frac{\mathcal{C}_2}{N} + \epsilon + o\left(\frac{1}{N}\right) \tag{2.9}$$

where $\lim_{N\to\infty} \mathcal{E}_N(0) = E_0$ *and* $\overline{\sigma^2} = \frac{1}{N}\sum_{i=1}^{N} \sigma_i^2$. $\mathcal{C}_1, \mathcal{C}_2$ *are constants that generally depend on the input distribution, target function, learning system and possibly* ϵ.

PROOF: See theorem 2.6.4 in appendix 2.6. ■

Corollary 2.3.3 *If* $C_1 \leq C_1' \ \forall \epsilon$ *then*

$$\mathcal{E}_N(\sigma) \leq \mathcal{E}_N(0) + \frac{\overline{\sigma^2} C_1'}{N} + \left(\frac{1}{N^2}\right) \tag{2.10}$$

Corollary 2.3.4 $\lim\limits_{N\to\infty} \mathcal{E}_N(\sigma) = \lim\limits_{N\to\infty} \mathcal{E}_N(0)$, *independent of* σ, *where we assume this limit is well defined.*

Combining (2.8) and (2.9) we get

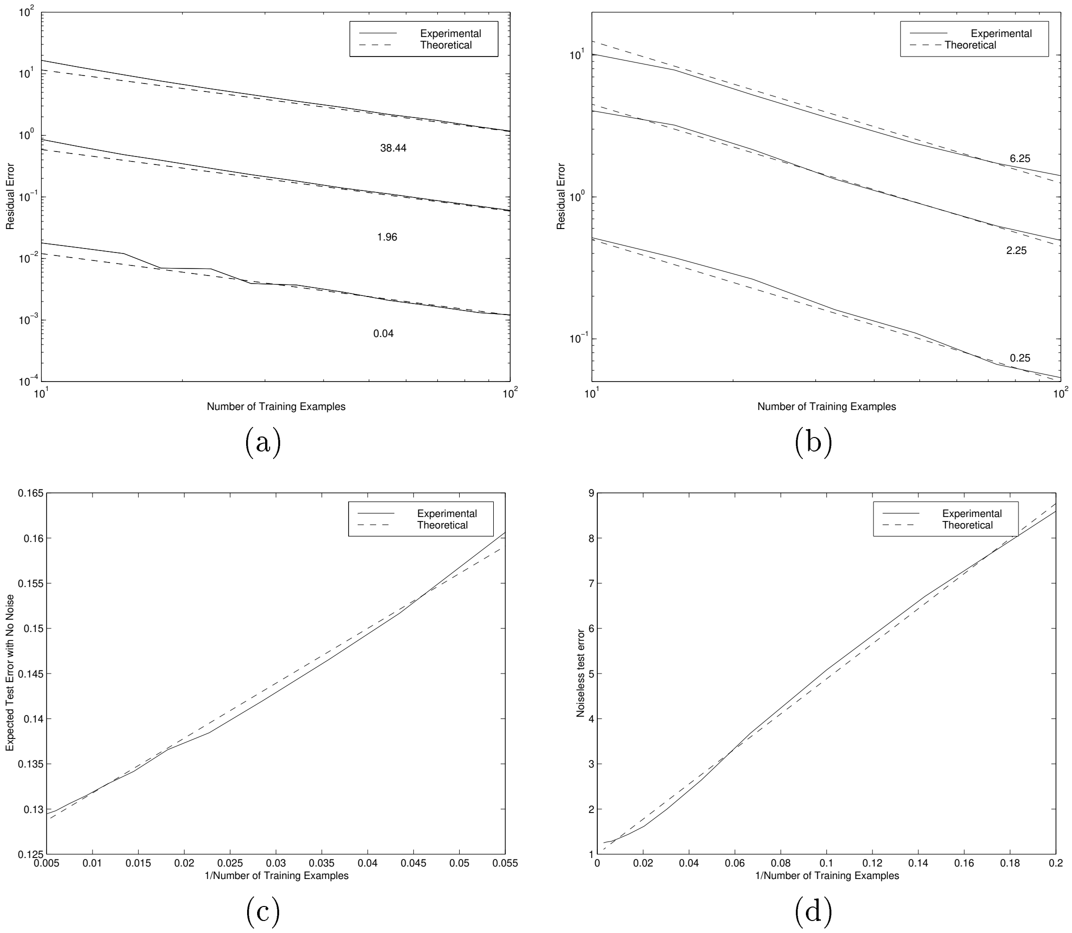
$$\mathcal{E}_N(\sigma) < E_0 + \frac{\mathcal{C}_1\overline{\sigma^2} + \mathcal{C}_2}{N} + o\left(\frac{1}{N}\right) \tag{2.11}$$

The essential content of the theorem is that the expected test error increases in proportion to $\overline{\sigma^2}$, holding everything else constant, and decreases in proportion to $1/N$, holding everything else constant. Corollary 2.3.4 tells us that when $N \to \infty$, the performance approaches the best attainable independent of the noise level. The conditions of theorem 2.3.2 are quite general and are satisfied by a wide variety of learning models and algorithms. For learning models that are linear, $\mathcal{C}_1 = d + 1$. E_0 is the model limitation modulo the learning algorithm when tested on noiseless data. The limiting performance on noisy future data is $E_0 + \overline{\sigma^2}$.

Experimentally we observe that the bounds of theorem 2.3.2 are quite tight even for small N (see figure 2.3) so combining (2.8) and (2.9) we expect the following dependence for $\mathcal{N}(\Delta, \sigma, N)$, the number of noisy examples that are equivalent to N noiseless examples.

$$\mathcal{N}(\Delta, \sigma, N) \sim \frac{\overline{\sigma^2}\mathcal{C}_1 + \mathcal{C}_2}{\frac{\mathcal{C}_2}{N} + \Delta} \tag{2.12}$$

For the results illustrated in figure 2.3, artificial data sets were created from a known



(a)Nonlinear target function and linear learning model – The residual error is shown for a nonlinear target function trained using a linear model. Gaussian noise with $\overline{\sigma^2}$ ranging from 0.04 to 38.44 was added to training sets. The dashed lines show the error level predicted by theorem 2.6.1. (b)Nonlinear target function and nonlinear learning model – The residual error is shown for a nonlinear target function trained using a 2-3-1 network. Gaussian noise with $\overline{\sigma^2}$ ranging from 0.25 to 6.25 was added to training sets. The results correspond very closely to those predicted by theorem 2.3.2 when $C_1 = 20$ (shown with dashed lines). (c)The behavior of the expected test error with no noise for the learning scenario in (a). We observe that for even small N there is close agreement between the theoretical $1/N$ decrease. (d)The behavior of the expected test error with no noise for the non-linear learning scenario in (b). Once again for small N we observe the expected behavior.

Figure 2.2: Experiments with simulated data, testing the theoretical bounds.

target function. Figure 2.3(a) illustrates the results of fitting a linear model to non-linear data. Shown is the residual error ($\hat{\mathcal{E}}_N(\sigma) = \mathcal{E}_N(\sigma) - \mathcal{E}_N(0)$) as a function of N on a log-log scale. The inputs are chosen from $\mathbb{R}^2$, and the dashed lines illustrate that $\hat{\mathcal{E}}_N(\sigma)$ quickly converges to $3\overline{\sigma^2}/N$ as expected from (2.5). Figure 2.3(b) shows similar results for a nonlinear learning model. Gradient descent was used to train the three hidden unit neural network model[3]. Ideally we expect this algorithm/model pair to be continuously compatible, and it was empirically shown to be mean preserving to considerable precision. The residual errors very closely follow $20\ \overline{\sigma^2}/N$, showing that we have approximate equality in (2.8) for $C_1 \sim 20$.[4] Figures 2.3 (c) and (d) show that $\mathcal{E}_N(0)$ also behaves linearly in $1/N$ for both cases (i.e. it quickly approaches the bound in 2.9).

The constants $\mathcal{C}_1,\ B,\ E_0$ in theorem 2.3.2 control the trade off between the BIAS and the sensitivity to noise/convergence rate. Simpler models will have a high value for E_0 but $\mathcal{C}_1$ and B will be small. More complex learning models will have a lower model limitation E_0, but higher convergence parameters $\mathcal{C}_1$ and B. This is the tradeoff that we illustrated in figure 1.4 and here we see that it is rooted in the size of the constants $\mathcal{C}_1$, B and E_0. For a given number of data points, there will be some "optimal model complexity."

2.3.1 Estimating the Model Limitation

When the learning model is linear, we can show (theorems 2.6.1, 2.6.2 in appendix 2.6) that the expected training error $\mathcal{E}_{tr}(\sigma)$ (the error on the data set) and expected test error approach the same limiting value from opposite sides as $N \to \infty$. Further the rates of convergence to this limiting value are the same. [Murata *et al.*, 1993] has obtained a similar asymptotic result in the case of nonlinear models when performing gradient descent on the training error. Using the Murata result, we can use our bound on the test error to bound the training error performance. The expected error on a

[3] A description of neural network and other learning models can be found in [Bishop, 1995a].
[4] This suggests that the condition in corollary 2.3.3 holds.

noisy data set, $\mathcal{E}_{test}$ is related to $\mathcal{E}_N(\sigma)$ by $\mathcal{E}_{test}(\sigma) = \mathcal{E}_N(\sigma) + \overline{\sigma^2}$. The experiments demonstrate that the bounds of theorem 2.3.2 are almost saturated for small N, so, ignoring terms that are $o(1/N)$, and using Murata's result for gradient descent, we have

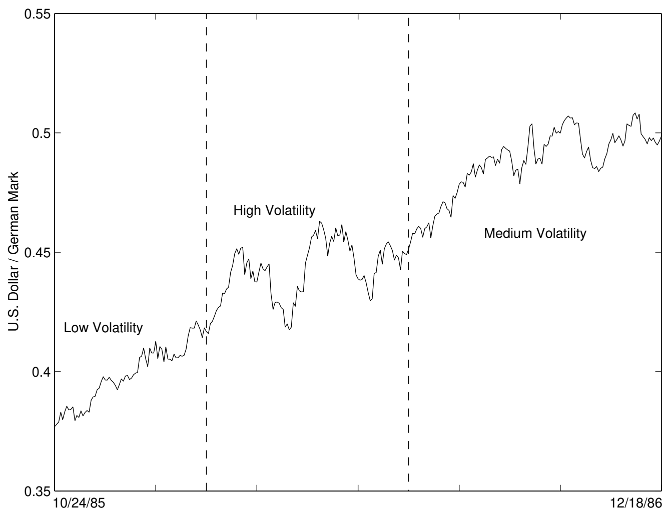
$$E_0 + \overline{\sigma^2} \leq \quad \mathcal{E}_{test}(\sigma) \approx \quad E_0 + \overline{\sigma^2} + \frac{\mathcal{C}_1\overline{\sigma^2} + \mathcal{C}_2}{N} \tag{2.13}$$

$$E_0 + \overline{\sigma^2} \geq \quad \mathcal{E}_{tr}(\sigma) \approx \quad E_0 + \overline{\sigma^2} - \frac{\mathcal{C}_1\overline{\sigma^2} + \mathcal{C}_2}{N} \tag{2.14}$$

(in the case of linear learning models we can replace $\mathcal{C}_1$ by $d+1$). From the data set of size N, for $N_1 < N$, we can randomly pick N_1 data points (perform Bootstrapping [Shao and Tu, 1996] on the training data). By varying N_1 in the training phase and observing the error on the training set, we obtain estimates of the model limitation $E_0 + \overline{\sigma^2}$ and of $\mathcal{C}_1\overline{\sigma^2} + \mathcal{C}_2$ by fitting the training error dependence on N_1 using (2.14). Thus, we can estimate the parameters that are needed for the bound (2.11). This is illustrated in the next section where we apply the results presented here to the case of financial time series.

2.4 Application to Financial Market Forecasting

Information processing of financial data entails the extraction of relevant information from overwhelming noise. Financial markets are dynamic systems so the noise parameters may fluctuate with time. In addition to being a nuisance that complicates the processing of financial data, noise plays a role as a tradable commodity in its own right. Indeed, market volatility is the basis for a number of financial instruments, such as *options* [Black and Scholes, 1973], whose price explicitly depends on the level of volatility in the underlying market. For this reason, it is of economic value to be able to predict the changes in the noise level in financial time series as these changes are reflected in the price changes in tradable instruments. These changes can be significant as one can observe in figure 2.3 where the U.S. Dollar/German Mark market



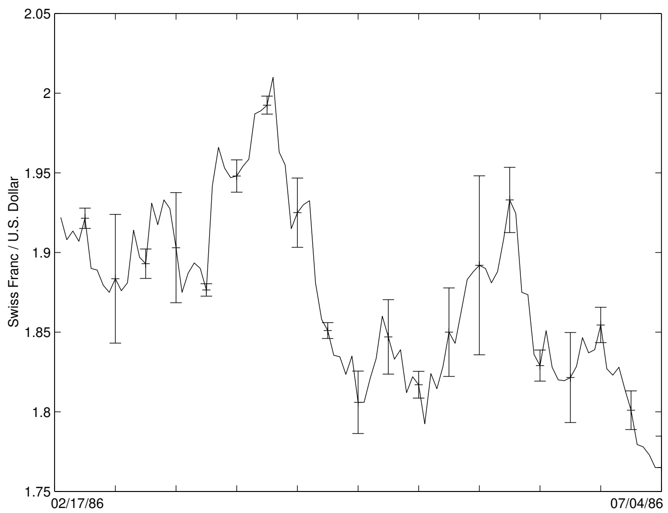
The market volatility can change noticeably over the course of time.

Figure 2.3: The price curve for the U.S. Dollar vs. German Mark, illustrating changes in volatility over time.

has undergone extreme changes in volatility.

In spite of the high levels of noise, financial data are among the best application domains for intelligent processing and advanced learning techniques. These data have been recorded very accurately for very long periods of time. They are available on different time scales, and simultaneously available in many different markets. This provides a very rich environment for analysis and experimentation using advanced processing techniques. Moreover, the payoff for even minute, but consistent, improvements in performance is huge.

Financial markets present us with data in the form of a time series. We might have the daily, hourly or tick-by-tick stock prices or foreign exchange rates. We would like to extract as much information as possible from historical data with the hope of learning the underlying behavior. In general, we can consider the value of a time series $y(t)$ at any time t as a noisy data point $y = f(\mathbf{x}) + \epsilon$. Here f is a deterministic function of a vector $\mathbf{x}(t)$ of market indicators and $\epsilon(t)$ is noise. The task at hand is one of learning $f(\cdot)$ from a finite data set (the history of the series).



Above is the realization of one such time series from the FX markets. Error bars are used to illustrate that each point is the outcome of a noisy event.

Figure 2.4: Financial time series tend to be very volatile.

In modeling the time variation of a stock price S, the standard Black-Scholes model for pricing options based on volatility [Black and Scholes, 1973] assumes the variation to be of the form

$$\mathrm{d}S = \tilde{\mu} S \mathrm{d}t + \tilde{\sigma} S \eta \sqrt{\mathrm{d}t}$$

where $\tilde{\sigma}$ is the market volatility and η has a zero-mean normal distribution with variance 1. Thus, the Black-Scholes model uses only the previous price as the indicator vector $\mathbf{x}$. The price at different times has a deterministic dependence on the past (the $\tilde{\mu}$ term) and a noisy component (the $\tilde{\sigma}$ term). The variance of the noisy component is related to the volatility and need not be constant. The precise relation is given by[5]

$$\sigma_i^2 = \tilde{\sigma}^2 S_i^2 \Delta t \tag{2.15}$$

[5]This relationship is valid for discrete time steps Δt. For continuous evolution a more careful analysis shows that $\Delta \log(S)$ is Gaussian with variance $\tilde{\sigma}^2 \Delta t$ [Wilmott *et al.*, 1995].

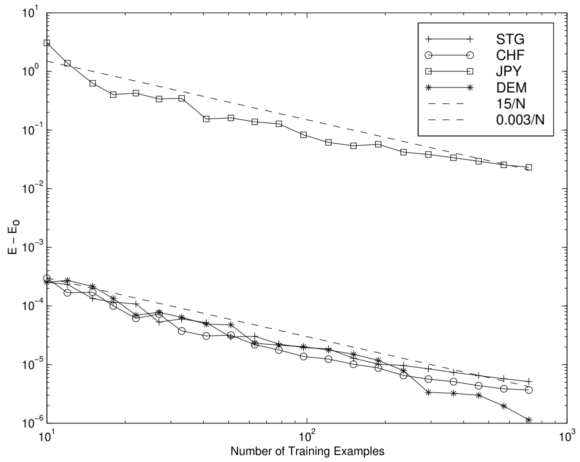
where i is a time index.

Extensive literature already exists on methods for extracting information from noisy time series ([Montgomery *et al.*, 1990], [Abu-Mostafa, 1990], [White, 1988]), [Trippi and Turban, 1993]. The details of such methods are not our present concern. We are interested in determining how our prediction performance depends on the amount of available data and the variability of the data (which is related to market volatility (2.15)) – what change in performance are we to expect if this year's market is more volatile than last year's market? What change in performance relative to some benchmark are we to expect if the market changed recently and hence we only have a few data points to learn from? These quantities can be obtained from $\mathcal{E}_N(\sigma)$. $\mathcal{E}_N(\sigma)$ is related to the "future profit" you expect to make having trained your learning system on the available data. Changes in $\mathcal{E}_N(\sigma)$ will be related to the tradeoff in profit when attempting to learn and predict during more volatile stages of the market compared to less volatile stages.

Pricing information is available on a variety of time scales, which presents us with a data set size vs. variability tradeoff. We could choose to use the tick-by-tick data because we will then have many data points, but the price we have to pay is that these data points are much noisier. The trade off will depend on how much noisier the tick-by-tick data is, and the details of the learning scheme. Market analysts would like to quantify this tradeoff by how it affects performance. This tradeoff is captured by $\mathcal{N}(\Delta, \sigma, N)$

An estimate of the best performance that we can achieve with a given information extraction scheme might also be economically useful. As well as providing a criterion for selecting between different models, knowing the model limitation could be useful for determining whether even an unlimited amount of data will give a system that is financially worth the risk. This would allow analysts to compare trading strategies based on their model limitation.

Can we apply the results of section 2.2 to real financial market data? Our experimental simulations indicate that we can. Figure 2.5 illustrates the $1/N$ behavior of the residual error $\hat{\mathcal{E}}_N(\sigma)$ for foreign exchange rates. Daily close exchange rates



The results are for the British Pound (STG), the Swiss Franc (CHF), the Japanese Yen (JPY) and the German Mark (DEM). Also shown are two lines that show $1/N$ behavior. We see that the test error curves follow the theory well.

Figure 2.5: The dependence of the test error-E_0 on N in some currency markets.

between 1984 and 1995 were used for the Swiss Franc (CHF), German Mark (DEM), British Pound Sterling (STG) and Japanese Yen (JPY). A linear model was used to learn the future price as a function of the close price of the previous five days.

We performed the following experiments. The last 1000 data points of each time series were held out as a test set. The remaining points were used to create a data set

$$\{\mathbf{x}_k = (S_{k-4}, \ldots, S_k), y_k = S_{k+1}\}$$

N_1 points were sampled from this set and used to learn. This was repeated to obtain an estimate of the expected test and training error. We show the dependence of the expected test error on the number of training examples in figure 2.5. Though it is not obvious that the assumptions made to derive the results hold, as with the results on artificial data, the test error seems to not only obey the bound of equation (2.8), but quickly assumes $1/N$ behavior. Assuming the bounds to be tight for both the test error and training error, we are able to estimate the best possible performance of the linear model ($E_0 + \sigma^2$) by finding the line best fitting $\mathcal{E}_N(\sigma)$ as a function of $1/N$. Table 2.1 summarizes these estimates.

Currency	$E_0 + \sigma^2$ Estimate (model lim.)	No Change Predictor (test error)
DEM	0.000499	0.000502
CHF	0.000158	0.000160
STG	0.000134	0.000136
JPY	1.082	1.083

(a)

Currency	$E_0 + \sigma^2$ Est. Estimate (model lim.)	No Change Predictor (test error)
DEM	0.000156	0.000152
CHF	0.000148	0.000151
STG	0.000153	0.000157
JPY	0.851	0.867

(b)

In (a) we use the training error to estimate $E_0 + \overline{\sigma^2}$ and compare to the performance on the training set when we use the simple system: predict no change in price. In (b) we use the test error curve to estimate E_0. Only (a) is possible in practice, but both yield very good estimates(if we assume that this simple strategy is close to the best you can do), thus verifying that the results of section 2.2 can be applied to this learning problem. The change in the estimate from (a) to (b) is due to the fact that the test and training sets are taken from different time intervals, and hence the estimates reflect a change in the market volatility over time.

Table 2.1: Estimate of model limitation and comparison to simple predictor.

We compare the model limitation to the test error obtained by simply predicting the present value as the next value. We find that this simple strategy virtually attains the model limitation suggesting that today's price completely reflects tomorrow's price – that's the best we can expect to achieve systematically. The results in table 2.1 are appealing on two accounts. Firstly, assuming that today's price is the best predictor of tomorrow's price, the technique we use to predict the model limitation is performing well (table 2.1 (a)). That todays price is the best predictor of tomorrows price is illustrated by table 2.1 (b) where the $E_0 + \sigma^2$ estimate is the true model limitation estimated using the test error. We see that the simple strategy basically achieves this model limitation. Secondly, because the model limitation estimates are slightly below the error of the simple strategy, we deduce that there is some information that can be extracted from previous prices.

By training on different time periods, we find that the model limitation may change as in the example in table 2.1. If we assume the underlying dependence to have remained constant so that E_0 has not changed, then the resulting change can only be due to a change in $\overline{\sigma^2}$ thus providing an estimate of the change in the volatility (since the volatility is related to the change in $\overline{\sigma^2}$ (2.15)). It appears from table 2.1 that of the four currencies, the British Pound's volatility seems to have increased while the remaining three markets display decreasing volatility.

Thus, we see that the results of section 2.2 apply to the problem of financial forecasting.

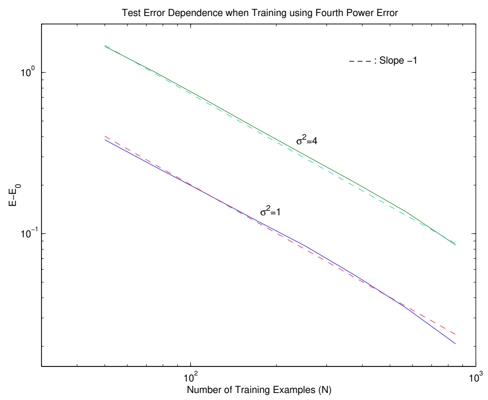
2.5 Comments

The new results in section 2.3 are represented in theorem 2.3.2. The experimental results on artificial data amply support the theory. We have shown that the number of noisy examples required for comparable generalization with N noiseless examples increases as $\overline{\sigma^2}$. Explicitly, the main result bounds the test error for a noisy data set by

$$\mathcal{E}_N(\sigma) \leq E_0 + \frac{\overline{\sigma^2}\mathcal{C}_1 + \mathcal{C}_2}{N} + o\left(\frac{1}{N}\right) \tag{2.16}$$

We also obtained a result that bounds the expected test error relative to a benchmark test error (2.8). Experimentally we showed that this result applies to the non-asymptotic regime – the empirical results show that the bounds hold with almost equality for N as small as 20. Intuitively this is because the non-asymptotic effects that affect $\mathcal{E}_N(\sigma)$ also have a similar effect on $\mathcal{E}_N(0)$.

We began with the goal of answering two questions: Relative to a benchmark scenario (that of learning with no noise), how does the performance change with the noise level and with the number of examples? This dependence is represented by the expression for $\mathcal{E}_N(\sigma)$ above. This is a similar result to those derived by [Murata *et al.*, 1993] and [Moody, 1991]. However the differences are significant. [Murata *et al.*, 1993] compares the training error when descending on a given error function to the expectation of *that* error when you have finished learning. The learning algorithm is specific but the form of the error function may vary. Moody considers minimization of an error term plus a complexity term and assumes that the input distribution is a sum of delta functions at the training data points. In this chapter, we derive a convergence result for the expected squared error without severely restricting the learning algorithm or the input distribution. We also quantified what change in performance is to be expected when noise is added to a data set. The results were also presented in the context of financial time series analysis, but we stress that they are applicable to the general learning problem, independent of most of the details of the learning model and learning algorithm. In particular, we do not require the learning algorithm to minimize a simple training error measure – optimizing a general regularized training error (as in [Plaut *et al.*, 1986]) should produce an algorithm that still satisfies the conditions of theorem 2.3.2. As an example, we minimized the fourth power of the



The results are for the two noise levels illustrated. The value of E_0 is 0.14 for both noise levels. For comparison are also shown the slope -1 fits.

Figure 2.6: The dependence of the test error on N when minimizing the fourth power of the residual as the training error measure.

training error as our learning algorithm. The expected squared error (our test error measure) should still converge at a rate $O(1/N)$, according to our theoretical results. This convergence is illustrated in figure 2.6

We provided an estimate of the model limitation. We used this estimate to estimate the best possible performance when learning in the FX markets. The results were consistent with the assumption that today's price reflects all the information about tomorrow's price. Using this method for predicting the model limitation, we could detect changes in the market volatility, which is of economic use.

It would be useful to explore the relationship between the constants $(E_0, \mathcal{C}_1, \mathcal{C}_2)$ that parameterize the expected test error dependence.

2.6 Appendix

2.6.1 Proofs of Results

We use the notation

$$\mathbf{X} = [\,\mathbf{x}_1 \quad \mathbf{x}_2 \quad \ldots \quad \mathbf{x}_N\,], \qquad \mathbf{y} = \mathbf{f} + \epsilon$$

$$\Sigma \equiv \left\langle \mathbf{x}\mathbf{x}^T \right\rangle_{\mathbf{x}}, \qquad \mathbf{q} = \left\langle \mathbf{x} f(\mathbf{x}) \right\rangle_{\mathbf{x}}$$

The law of large numbers gives us that $\mathbf{X}\mathbf{X}^T \underset{N\to\infty}{\longrightarrow} N\Sigma$ and $\mathbf{X}\mathbf{f} \underset{N\to\infty}{\longrightarrow} N \left\langle \mathbf{x} f(\mathbf{x}) \right\rangle_{\mathbf{x}}$, where we assume that the conditions for this to happen are satisfied.

Theorem 2.6.1 *Let $\mathcal{H}$, the learning model, be the set of linear functions $\mathbf{w} \cdot \mathbf{x} + w_0$ and let the learning algorithm be minimization of the squared error. Then*

$$\mathcal{E}_N(\sigma) = \mathcal{E}_N(0) + \frac{\overline{\sigma^2}(d+1)}{N} + O\left(\frac{1}{N^2}\right) \tag{2.17}$$

$$\mathcal{E}_N(0) = E_0 + \frac{B}{N} + O(\frac{1}{N^{\frac{3}{2}}}) \tag{2.18}$$

where $E_0 = \lim\limits_{N\to\infty} \{\mathcal{E}_N(0)\}$ and B is a constant dependent on the input distribution. It follows that $\mathcal{N}(\Delta, \sigma, N) = \frac{\overline{\sigma^2}(d+1)+B}{\Delta + \frac{B}{N}} + O(\frac{1}{N^{\frac{3}{2}}})$.

Proof: $g \in \mathcal{L} \Leftrightarrow g(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$. The Least Squares estimate of $\mathbf{w}$ is given by

$$\hat{\mathbf{w}} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y} \tag{2.19}$$

from which we calculate

$$\begin{aligned} \mathcal{E}_N(\sigma) &= \left\langle \hat{\mathbf{w}}^T\mathbf{x}\mathbf{x}^T\hat{\mathbf{w}} - 2\hat{\mathbf{w}}^T\mathbf{x} f(\mathbf{x}) + f(\mathbf{x})^2 \right\rangle_{\mathbf{x},\mathbf{X},\epsilon} \\ &= \left\langle f^2 \right\rangle - 2\left\langle \mathbf{f}^T\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1} \right\rangle_{\mathbf{X}} \mathbf{q} + \left\langle \mathbf{f}^T\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\Sigma(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{f} \right\rangle_{\mathbf{X}} \\ &\quad + \left\langle \epsilon^T\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\Sigma(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\epsilon \right\rangle_{\mathbf{X},\epsilon} \end{aligned}$$

$$= \quad \mathcal{E}_N(0) + \sum_i \sigma_i^2 \left\langle \left[\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\Sigma(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\right]_{ii} \right\rangle_{\mathbf{X}}$$

where we have used (2.1) and $\mathbf{q} = \langle \mathbf{x} f(\mathbf{x}) \rangle_{\mathbf{x}}$. By the law of large numbers, we note that $(\mathbf{X}\mathbf{X}^T)^{-1} \underset{N\to\infty}{\longrightarrow} N\Sigma$ and $\mathbf{X}\mathbf{f} \underset{N\to\infty}{\longrightarrow} N\mathbf{q}$, so we write

$$\mathbf{X}\mathbf{X}^T = N\Sigma + \sqrt{N}\mathbf{V}(\mathbf{X}), \qquad \mathbf{X}\mathbf{f} = N\mathbf{q} + \sqrt{N}\mathbf{a}(\mathbf{X}) \tag{2.20}$$

where $\langle \mathbf{V} \rangle_{\mathbf{X}} = \langle \mathbf{a} \rangle_{\mathbf{X}} = 0$ and $\mathrm{Var}(\mathbf{V})$ and $\mathrm{Var}(\mathbf{a})$ are $O(1)$ by the central limit theorem[6]. Using (2.20) and the identity $[1 + \lambda\mathbf{A}]^{-1} = 1 - \lambda\mathbf{A} + \lambda^2\mathbf{A}^2 + O(\lambda^3)$, we obtain the expansion

$$\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\Sigma(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X} = \frac{\mathbf{X}^T\Sigma^{-1}\mathbf{X}}{N^2} - 2\frac{\mathbf{X}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{X}}{N^{\frac{5}{2}}} + 3\frac{\mathbf{X}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{X}}{N^3} + O\left(\frac{1}{N^{\frac{7}{2}}}\right)$$

From the definition of $\mathbf{X}$ we find from the first term

$$\left[\left\langle \mathbf{X}^T\Sigma^{-1}\mathbf{X} \right\rangle\right]_{ii} = \left\langle \sum_{k,l} \Sigma^{-1}_{kl}(\mathbf{x}_i)_k(\mathbf{x}_i)_l \right\rangle = [\sum_{k,l} \Sigma^{-1}_{kl}\Sigma_{kl}]_{ii} = 1$$

by taking the expectation of the trace of both sides of the equation, the second term can be shown to be of the same order as the third term. So

$$\mathcal{E}_N(\sigma) = \mathcal{E}_N(0) + \frac{\sum_i \sigma_i^2}{N^2} + O(\frac{1}{N^2}) \tag{2.21}$$

The first part of the theorem now follows. Using similar techniques for $\mathcal{E}_N(0)$, we find

$$\begin{aligned}
\mathcal{E}_N(0) \quad &= \quad \left\langle f^2 \right\rangle - \mathbf{q}^T\Sigma^{-1}\mathbf{q} + \overbrace{\frac{\left\langle \mathbf{q}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q} - \mathbf{a}^T\Sigma^{-1}\mathbf{q} \right\rangle}{N^{\frac{1}{2}}}}^{0} \\
&\quad + \frac{\left\langle \mathbf{q}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q} + \mathbf{a}^T\Sigma^{-1}\mathbf{a} - 2\mathbf{a}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q} \right\rangle}{N} + O\left(\frac{1}{N^{\frac{3}{2}}}\right) \\
&= \quad E_0 + \frac{B}{N} + O\left(\frac{1}{N^{\frac{3}{2}}}\right)
\end{aligned}$$

[6]We assume that the conditions for the central limit theorem to hold are valid.

with $E_0 = \langle f^2 \rangle - \mathbf{q}^T \Sigma^{-1} \mathbf{q}$ and B depending on the input distribution. This gives the N dependence of $\mathcal{E}_N(0)$. Finally we have

$$\mathcal{E}_{\mathcal{N}}(\sigma) - \mathcal{E}_N(0) \quad = \quad \frac{\overline{\sigma^2}(d+1) + B}{\mathcal{N}} - \frac{B}{N} = \Delta$$

yielding the functional dependence $\mathcal{N}(\Delta, \sigma, N)$. ■

This result can immediately be generalized to the case where the learning model is linear in its parameter space. A similar technique can be used to derive a result on the expected mean squared residual itself which we will call $\mathcal{E}_r(\sigma)$.

Theorem 2.6.2 *Let $\mathcal{H}$, the learning model, be the set of linear functions $\mathbf{w} \cdot \mathbf{x} + w_0$ and let the learning algorithm be minimization of the squared error. Then*

$$\mathcal{E}_r(\sigma) = \mathcal{E}_r(0) + \overline{\sigma^2} - \frac{\overline{\sigma^2}(d+1)}{N} \tag{2.22}$$

$$\mathcal{E}_r(0) = E_0 - \frac{B}{N} + O(\frac{1}{N^{\frac{3}{2}}}) \tag{2.23}$$

where E_0 and B are the same constants appearing in theorem 2.6.1. Thus we find

$$\mathcal{E}_N(\sigma) - \mathcal{E}_r(\sigma) = \frac{2(\overline{\sigma^2}(d+1) + B)}{N} + o(\frac{1}{N^2}) \tag{2.24}$$

where, now, $\mathcal{E}_N(\sigma)$ is the noisy test error.

Proof: The residual error is given by

$$\begin{aligned}
\mathcal{E}_r(\sigma) &= \left\langle \frac{1}{N}(\hat{\mathbf{X}}^T\mathbf{w} - \mathbf{y})^2 \right\rangle = \left\langle \frac{1}{N}((\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X} - \mathbf{1})y)^2 \right\rangle \\
&= \frac{\langle \mathbf{f}^T\mathbf{f} \rangle - \langle f^T\mathbf{X}(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{f} \rangle + \langle \epsilon^T\epsilon \rangle - \langle \epsilon^T\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\epsilon \rangle}{N} \\
&= \underbrace{\langle f^2 \rangle - \frac{\langle \mathbf{f}^T X^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{f} \rangle}{N}}_{\mathcal{E}_r(0)} + \overline{\sigma^2} - \frac{\overline{\sigma^2}(d+1)}{N}
\end{aligned}$$

from which the first part of the theorem follows. Using the techniques of theorem 2.6.1 we find that

$$\begin{aligned}\mathcal{E}_r(0) &= \left\langle f^2 \right\rangle - \mathbf{q}^T\Sigma^{-1}\mathbf{q} \\ &\quad - \frac{\left\langle \mathbf{q}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q} + \mathbf{a}^T\Sigma^{-1}\mathbf{a} - 2\mathbf{a}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q} \right\rangle}{N} + O(\frac{1}{N^{\frac{3}{2}}})\end{aligned}$$

Comparing with theorem 2.6.1 we have the second part of the theorem. ∎

This result is similar to the results obtained by [Murata *et al.*, 1993], [Moody, 1991]. We now consider the case of a non-linear learning model. The following proposition shows that $\lim_{N\to\infty}\mathcal{A}(\mathcal{D}_N) = g_\infty$ is well defined point wise – i.e., $\forall\epsilon > 0$, $\exists M$ such that if $N > M$ then $\max_{\mathbf{x}} |g_\infty - \mathcal{A}(\mathcal{D}_N)| < \epsilon$. This can be skipped if this fact is self evident or if one wishes to assume convergence and one is merely interested in the rate. It is included here purely for technical completeness.

Proposition 2.6.3 *Let $\mathcal{L} \in \mathcal{CC}_2$. Then, the limit $\lim_{N\to\infty}\mathcal{A}(\mathcal{D}_N) = g_\infty$ for noiseless data sets is well defined point wise on sets of non-zero probability – i.e., $\forall\epsilon > 0$, $\exists M$ such that if $N > M$ then $\max_{\mathbf{x}} |g_\infty - \mathcal{A}(\mathcal{D}_N)| < \epsilon$.*

Proof: We will sketch the idea of the proof, the details can be filled in using exactly the same techniques as for the proof to theorem 2.6.6. First we show that for any two infinite data sets, the learned functions are essentially identical. For any infinite data set, as the input support is compact (closed and bounded), any infinitesimal volume of non zero probability has an infinite number of data points. Consider two such data sets. The means of the targets in this small volume will be equal (by the law of large numbers). Because the target function is continuous on this compact support, the means for the two data sets are arbitrarily close to the true values for each data set (this can be attained by letting the the size of the volume be arbitrarily small). By continuous compatibility, these two data sets must both be mapped arbitrarily close to the data set with the means as targets. Therefore they must be mapped arbitrarily close to each other. Thus, we see that $\left\langle (g_1 - g_2)^2 \right\rangle$ is less than ϵ for arbitrary small

ϵ, where the two different data sets drawn from the input distribution are mapped to g_i. So we conclude that $\langle (g_1 - g_2)^2 \rangle = 0$, therefore, $g_1 = g_2$ with probability 1. Thus, any two infinite noiseless data sets are mapped to the same function (as the functions are continuous), which we call g_∞.

Finally, consider a data set $\mathcal{D}_N$. For N large enough, this data set can be made arbitrarily close to an infinite data set using the argument above. Let $g_N = \mathcal{A}(\mathcal{D}_N)$. Therefore $\langle (g_N - g_\infty)^2 \rangle$ can be made arbitrarily small by choosing N large enough. In other words, $\lim_{N\to\infty} \langle (g_N - g_\infty)^2 \rangle = 0$, therefore g_N converges to g_∞ with probability 1. Further, because the functions are continuous and the support is compact, this convergence is uniform. ■

We have just shown that the limit $\mathcal{A}(\mathcal{D}_N)$ exists as $N \to \infty$. Thus, with noiseless data sets, we have convergence for stable learning systems. We now consider both the rate of convergence and what happens when noise is added.

Theorem 2.6.4 *Let $\mathcal{L}$ be stable. Let the target function f be continuous. Let the probability measure on the input space have compact support $\mathcal{X}$. Then $\forall \epsilon > 0$, $\exists C_1 > 0$ such that using $\mathcal{L}$, it is at least possible to attain a test error bounded by*

$$\mathcal{E}_N(\sigma) < \mathcal{E}_N(0) + \frac{\overline{\sigma^2} C_1}{N} + \epsilon + O\left(\frac{1}{N^2}\right) \tag{2.25}$$

PROOF: By rescaling, we can assume that the input space $\mathcal{X} \subseteq \mathcal{S} = [0,1]^d$. f is continuous, so it is uniformly continuous on the compact set $\mathcal{S}$. Therefore, $\exists \delta_1$ such that

$$\mid \mathbf{x} - \mathbf{x}' \mid < \delta_1 \Rightarrow \mid f(\mathbf{x}) - f(\mathbf{x}') \mid < \delta_2$$

Divide $[0,1]$ into intervals of size $\delta_1/\sqrt{d}$. Thus we divide $\mathcal{S}$ into $\left(\sqrt{d}/\delta_1\right)^d$ cubes. Let $C_{\mathbf{i}} \equiv C_{i_1,i_2\ldots i_d}$ define the cube with lowest coordinates $\mathbf{i}$. Let $N_{\mathbf{i}}$ be the number of data points in $C_{\mathbf{i}}$, and let $\mu_{\mathbf{i}} = \frac{1}{N_{\mathbf{i}}} \sum_{x_j \in C_{\mathbf{i}}} y_j$. Let $P_{\mathbf{i}} = Pr\{x \in C_{\mathbf{i}}\}$. We only need consider regions where $P_{\mathbf{i}} > 0$, as regions with $P_{\mathbf{i}} = 0$ are don't care regions. The following Lemma is easily obtained by noting that for $\mathbf{x}, \mathbf{x}' \in C_{\mathbf{i}}$, $|f(\mathbf{x}) - f(\mathbf{x}')| \leq \delta_2$.

Lemma 2.6.5 *Let* $\mathbf{x} \in C_{\mathbf{i}}$

$$\left| \frac{1}{N_{\mathbf{i}}} \sum_{\mathbf{x}_j \in C_{\mathbf{i}}} y_j - f(\mathbf{x}) \right| = |\ \mu_{\mathbf{i}} - f(\mathbf{x})\ | \leq \delta_2 + \frac{|\sum_{x_k \in C_i} \epsilon_k|}{N_{\mathbf{i}}}$$

Construct a new data set by replacing all the y's in $C_{\mathbf{i}}$ by $\mu_{\mathbf{i}}$. i.e., with no noise, the targets would be $f(\mathbf{x}_j)$ and with noise they are $\mu_{\mathbf{i}}$. $\forall \mathbf{x}_j \in C_{\mathbf{i}}$,

$$\mu_i = f(\mathbf{x}_j) + \underbrace{\sum_{\mathbf{x}_k \neq \mathbf{x}_j} \frac{f(\mathbf{x}_k) - f(\mathbf{x}_j)}{N_{\mathbf{i}}}}_{|\cdot| < \delta_2} + \frac{\sum_{x_k \in C_i} \epsilon_k}{N_{\mathbf{i}}} = f(\mathbf{x}_j) + \eta_j + \xi_j$$

where $\eta_j = \sum_{\mathbf{x}_k \neq \mathbf{x}_j} \frac{f(\mathbf{x}_k) - f(\mathbf{x}_j)}{N_{\mathbf{i}}}$ and $\xi_j = \sum_{x_k \in C_i} \epsilon_k / N_{\mathbf{i}}$. We have that $\langle \eta_j \rangle_{\mathcal{D}_N} = 0$ and $\langle \xi_j \rangle_\epsilon = 0$. Let $\mathcal{A}$ map the noiseless data set to $g_0 \in \mathcal{H}$ and this noisy version of the data set to $g = g_0 + \eta$. So for the test error we have

$$\mathcal{E}_N(\sigma) = \left\langle (f-g)^2 \right\rangle = \underbrace{\left\langle (f-g_0)^2 \right\rangle}_{\mathcal{E}_N(0)} + \underbrace{2\left\langle (f-g_0)(g_0-g) \right\rangle}_{T_1} + \underbrace{\left\langle (g_0-g)^2 \right\rangle}_{T_2}$$

where the expectations are with respect to the test point, training data points and the noise $(\mathcal{D}_N, \mathbf{x}, \epsilon)$. We now examine T_1 and T_2.

$$\begin{aligned}
|T_1| &= \left| \left\langle (f-g_0) \langle g_0 - g \rangle_\epsilon \right\rangle_{\mathcal{D}_N, \mathbf{x}} \right| \\
&\stackrel{(a)}{=} \left| \left\langle (f-g_0)(\mathcal{A}(\{\mathbf{x}_k, f(\mathbf{x}_k)\}) - \mathcal{A}(\{\mathbf{x}_k, f(\mathbf{x}_k) + \eta_k\})) \right\rangle_{\mathcal{D}_N, \mathbf{x}} \right| \\
&\leq \left| \left\langle f \left\langle (\mathcal{A}(\{\mathbf{x}_k, f(\mathbf{x}_k)\}) - \mathcal{A}(\{\mathbf{x}_k, f(\mathbf{x}_k) + \eta_k\})) \right\rangle_{\mathcal{D}_N} \right\rangle_{\mathbf{x}} + \right| \\
&\quad \left| \left\langle g_0 \left[\mathcal{A}(\{\mathbf{x}_k, f(\mathbf{x}_k)\}) - \mathcal{A}(\{\mathbf{x}_k, f(\mathbf{x}_k) + \eta_k\}) \right] \right\rangle_{\mathcal{D}_N, \mathbf{x}} \right| \\
&\stackrel{(b)}{\leq} \left\langle \max_{\mathbf{x}} |\ g_0| \left\langle |\mathcal{A}(\{\mathbf{x}_k, f(\mathbf{x}_k)\}) - \mathcal{A}(\{\mathbf{x}_k, f(\mathbf{x}_k) + \eta_k\})| \right\rangle_{\mathbf{x}} \right\rangle_{\mathcal{D}_N} \\
&\stackrel{(c)}{\leq} C\delta_2 \left\langle \max_{\mathbf{x}} |\ g_0| \right\rangle_{\mathcal{D}_N} \\
&\stackrel{(d)}{\leq} c_1 \delta_2
\end{aligned}$$

where (a) and (b) follow from the mean preserving assumption. (c) from continuous compatibility and (d) because the limit g_∞ exists point wise therefore the sequence $\langle \max_{\mathbf{x}} \mid g_0 \mid \rangle_{\mathcal{D}_N}$ is a bounded sequence. Similarly, for T_2 we get

$$
\begin{aligned}
|T_2| &= \left| \left\langle (\mathcal{A}(\{\mathbf{x}_k, f(\mathbf{x}_k)\}) - \mathcal{A}(\{\mathbf{x}_k, f(\mathbf{x}_k) + \eta_k + \xi_k\}))^2 \right\rangle_{\mathcal{D}_N,\mathbf{x},\epsilon} \right| \\
&\overset{(a)}{\leq} 2C^2 \left(\delta_2^2 + \left\langle \sum_{\mathbf{i}} \frac{\sum_{\mathbf{x}_j,\mathbf{x}_k \in C_{\mathbf{i}}} \epsilon_j \epsilon_k}{N_{\mathbf{i}}^2} \right\rangle_{\mathcal{D}_N,\epsilon} \right) \\
&= 2C^2 \left(\delta_2^2 + \left\langle \sum_{\mathbf{i}} \frac{\bar{\sigma}_i^2}{N_{\mathbf{i}}} \right\rangle_{\mathcal{D}_N,\epsilon} \right) \\
&\overset{(b)}{=} 2C^2 \left(\delta_2^2 + \overline{\sigma^2} \left\langle \sum_{\mathbf{i}} \frac{1}{N_{\mathbf{i}}} \right\rangle_{\mathcal{D}_N} \right) \\
&= 2C^2\delta_2^2 + 2C^2\overline{\sigma^2} \sum_{\mathbf{i}} \underbrace{\sum_{n=1}^{N} \frac{1}{n} \binom{N}{n} P_{\mathbf{i}}^n (1 - P_{\mathbf{i}})^{N-n}}_{\frac{1}{NP_{\mathbf{i}}} + O\left(\frac{1}{N^2}\right)} \\
&= 2C^2\delta_2^2 + \frac{\overline{\sigma^2}}{N} \underbrace{2C^2 \sum_{\mathbf{i}} \frac{1}{P_{\mathbf{i}}}}_{C_1} + O(\frac{1}{N^2})
\end{aligned}
$$

(a) follows from the continuous compatibility assumption. (b) follows because the noise is chosen independently of the inputs. Choosing δ_1 such that $c_1\delta_2 + 2C^2\delta_2^2 < \epsilon$ we have

$$
\mathcal{E}_N(\sigma) \leq \mathcal{E}_N(0) + \frac{C_1(\epsilon)\overline{\sigma^2}}{N} + \epsilon + O\left(\frac{1}{N^2}\right)
$$

■

We note that it is easy to extend these theorems to the case where the noise variances are drawn from some distribution. By taking the expectation over that distribution, the same result with $\overline{\sigma^2}$ being the expected value of the variance parameter is obtained. Note also that the preceding proof is by no means suggesting a method to calculate C_1. It is simply a means to show its existence. Often, especially when the input distribution is bounded, Corollary 2.3.3 will hold, and it might be possible to estimate these constants experimentally.

One might wonder what would happen if the mean preserving assumption is violated. We note that the only place where this is used is in the evaluation of T_1. Continuity could still be used however with the difference being that a term of order $\epsilon/\sqrt{N}$ would remain. in other words, one would have $\mathcal{E}_N(\sigma) \leq \mathcal{E}_N(0) + C''\sigma/\sqrt{N}$ + higher order. So if we do not have the mean preserving property then these methods do not guarantee $1/N$ convergence of the test error. Using identical methods, one can, however, get the following result using the continuity property alone: $\langle|f-g|\rangle \leq \langle|f-g_0|\rangle + \frac{C'''\sigma}{\sqrt{N}}$. This is very similar to theorem 2.6.4 where one measures test error by the expectation of the magnitude difference as opposed to the squared difference.

We now derive a theorem for the dependence of $\mathcal{E}_N(0)$ on N.

Theorem 2.6.6 *Let $\mathcal{L}$ be stable. Let the target function f be continuous. Let the probability measure on the input space have compact support. Then $\forall\epsilon > 0,\ \exists C_2 > 0$ such that using $\mathcal{L}$ it is at least possible to attain a noiseless test error bounded by*

$$\mathcal{E}_N(0) < E_0 + \frac{C_2}{N} + \epsilon + o\left(\frac{1}{N}\right) \tag{2.26}$$

Corollary 2.6.7 *If $C_2 \leq C_2'\ \forall\epsilon$ then $\mathcal{E}_N(0) \leq E_0 + \frac{C_2'}{N} + o\left(\frac{1}{N}\right)$ where $E_0 = \lim_{N\to\infty} \mathcal{E}_N(0)$*

Before we proceed to the proof of the theorem, the following lemma is needed.

Lemma 2.6.8 *Let N balls be independently be distributed into r cells according to the probabilities $p_1 \ldots p_r$. Then for every $m > 0, \exists A_m$ such that the probability, q, that at least one cell is empty is bounded by*

$$q \leq \frac{A_m}{N^m}$$

PROOF:

$$
\begin{aligned}
q &= Pr[\cup\, cell_i \ is \ empty] \leq \sum_i Pr[cell_i \ is \ empty] = \sum_i (1-p_i)^N \\
&\leq r(1-\min_i p_i)^N \leq \frac{A_m}{N^m}
\end{aligned}
$$

choosing $A_m \geq r(-m/ln(a))^m$. ■

PROOF OF THEOREM 2.6.6

Let $\mathcal{X}$, $\mathcal{S}$, δ_1, δ_2, $C_{\mathbf{i}}$, $P_{\mathbf{i}}$, $N_{\mathbf{i}}$ be as in the proof of theorem 2.6.1. We only consider those cubes with $P_{\mathbf{i}} > 0$.

Suppose that we have an infinite noiseless data set, $\mathcal{D}_\infty$. For all $\mathbf{i}$, let $\bar{y}_{\mathbf{i}} = \langle f(\mathbf{x})\rangle_{\mathbf{x}|\mathbf{x}\in C_{\mathbf{i}}}$ and let $\mu_{\mathbf{i}} = \frac{1}{N_{\mathbf{i}}}\sum_{\mathbf{x}_j\in C_{\mathbf{i}}} y_j$ if $C_{\mathbf{i}}$ is non-empty else, $\mu_{\mathbf{i}} = 0$. Construct two data sets from the infinite one, $\mathcal{D}_1$ and $\mathcal{D}_2$, by replacing all the y's in $C_{\mathbf{i}}$ by $\bar{y}_{\mathbf{i}}$, and $\mu_{\mathbf{i}}$ respectively. $\mathcal{D}_1$ does not depend on $\mathcal{D}_N$ and $\mathcal{D}_2$ can be obtained from $\mathcal{D}_N$. $\mathcal{D}_\infty$ and $\mathcal{D}_1$ are close data sets because for $\mathbf{x} \in C_{\mathbf{i}}$,

$$
\left| f(\mathbf{x}) - \langle f(\mathbf{y})\rangle_{\mathbf{y}|\mathbf{y}\in C_{\mathbf{i}}} \right| = \left| \langle f(\mathbf{x}) - f(\mathbf{y})\rangle_{\mathbf{y}|\mathbf{y}\in C_{\mathbf{i}}} \right| \leq \langle |f(\mathbf{x}) - f(\mathbf{y})|\rangle_{\mathbf{y}|\mathbf{y}\in C_{\mathbf{i}}} \leq \delta_2
$$

Therefore by continuous compatibility, $\langle (g_\infty - g_1)^2\rangle \leq C^2\delta_2^2$. Define $\epsilon_{\mathbf{i}}$ by $\mu_{\mathbf{i}} = \bar{y}_{\mathbf{i}} + \epsilon_{\mathbf{i}}$. Then $\langle \epsilon_{\mathbf{i}}\rangle_{\mathcal{D}_N} = 0$ for all non-empty $C_{\mathbf{i}}$. Let g_∞, g_1 and g_2 be $\mathcal{A}(\mathcal{D}_\infty)$, $\mathcal{A}(\mathcal{D}_1)$ and $\mathcal{A}(\mathcal{D}_2)$ respectively. Since we can construct $\mathcal{D}_2$, using $\mathcal{L}$ we can at least obtain a test error given by

$$
\begin{aligned}
\mathcal{E}_N(0) \ \leq \ & \overbrace{\left\langle (f-g_\infty)^2 \right\rangle}^{E_0} + \overbrace{\left\langle (g_\infty - g_1)^2 \right\rangle}^{\leq C^2\delta_2^2} + \left\langle (g_1-g_2)^2 \right\rangle \\
& +2\underbrace{|\,\langle (f-g_\infty)(g_\infty-g_1)\rangle\,|}_{\substack{\leq |f-g_\infty|_{max} C\delta_2 \\ \text{(by cont. comp.)}}} + 2|\,\langle (f-g_\infty)(g_1-g_2)\rangle\,| + 2|\,\langle (g_\infty-g_1)(g_1-g_2)\rangle\,|
\end{aligned}
$$

By the mean preserving property, $\langle (g_1-g_2)\rangle_{\mathcal{D}_N} = 0$. Therefore,

$$
|\,\langle (f-g_\infty)(g_1-g_2)\rangle\,| = |\,\left\langle (f-g_\infty)\,\langle (g_1-g_2)\rangle_{\mathcal{D}_N} \right\rangle_{\mathbf{x}}\,| = 0
$$

Similar reasoning shows that $2|\langle(g_\infty - g_1)(g_1 - g_2)\rangle| = 0$. Let Q be the probability that at least one cell is empty. For the final term we have

$$\begin{aligned}
\left\langle (g_1 - g_2)^2 \right\rangle &\overset{(a)}{\leq} (1-Q)C^2 \left\langle \sum_{\mathbf{i}} \epsilon_{\mathbf{i}}^2 \right\rangle_{\mathcal{D}_N | \forall \mathbf{i}, N_{\mathbf{i}}>0} + 4QC^2 |f|_{max}^2 \\
&\leq C^2 \left\langle \left\langle \sum_{\mathbf{i}} \epsilon_{\mathbf{i}}^2 \right\rangle_{y_{\mathbf{i}} | N_{\mathbf{i}}>0} \right\rangle_{N_{\mathbf{i}}>0} + c_1 Q \\
&\leq C^2 \left\langle \sum_{\mathbf{i}} \frac{\sigma_{\mathbf{i}}^2}{N_{\mathbf{i}}} \right\rangle_{N_{\mathbf{i}}>0} + c_1 Q \\
&\overset{(b)}{\leq} \frac{1}{N} \underbrace{C^2 \sum_{\mathbf{i}} \frac{\sigma_{\mathbf{i}}^2}{P_{\mathbf{i}}}}_{C_2} x + o(\frac{1}{N})
\end{aligned}$$

where $\sigma_{\mathbf{i}}^2 = Var(y_{\mathbf{i}} | y \in C_{\mathbf{i}})$. (a) follows by continuous compatibility because with probability $1 - Q$ the data sets are at most $\epsilon_{\mathbf{i}}$ apart and $\sum_{\mathbf{i}} \epsilon_{\mathbf{i}}^2 \geq \max_i \epsilon_{\mathbf{i}}^2$, and with probability Q they are at most $2|f|_{max}$ apart. (b) follows because $\langle 1/N_{\mathbf{i}} \rangle = 1/(NP_{\mathbf{i}}) + o(1/N)$ and Lemma 2.6.8 can be used to yield $Q = o(1/N)$. Finally we have

$$\mathcal{E}_N(0) \leq E_0 + C^2\delta_2^2 + 2|f - g_\infty|_{max}\delta_2 + \frac{C_2}{N} + o(\frac{1}{N})$$

Choosing δ_2 small enough, we have the theorem because $|f - g_\infty|$ is bounded on the compact support $\mathcal{X}$. ∎

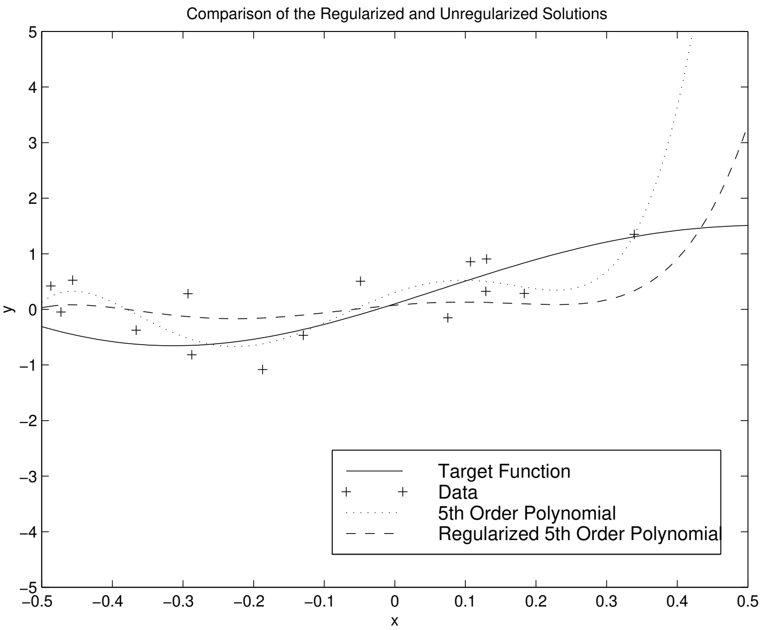
Chapter 3
Regularization in Learning Systems[1]

— A person is at his best when under just the right amount of pressure, not too much and not too little.

–Yerkes-Dodson/Amir Atiya.

— We are all regularized by our parents.

3.1 Introduction



15 noisy data were generated uniformly in $[-0.5, 0.5]$ *from the target function. Shown are the fits with and without regularization.*

Figure 3.1: An example of regularization using 5^{th} order polynomials.

[1]Parts of this chapter were done in collaboration with Xubo Song of the Learning Systems Group

Regularization techniques have been used to improve out of sample performance, typically by smoothing the fit to noisy data. Generally, they correspond to adding a penalty term to the empirical risk, that somehow penalizes the complexity of the learned function.

$$\hat{\mathcal{E}} = E_0 + \lambda\Omega \tag{3.1}$$

where E_0 is the empirical risk, Ω is the complexity regularizer and λ controls the relative weighting given to the complexity penalty. Well established techniques include Tikhonov type smoothing regularizers [Tikhonov and Arsenin, 1977] [Bishop, 1993] [Wu and Moody, 1996], heuristic weight decay regularizers [Plaut *et al.*, 1986], weight elimination [Scalettar and Zee, 1988], soft weight sharing [Nowlan and Hinton, 1992]. It is well established that such regularization techniques improve out of sample performance when one attempts to learn from a noisy training set. Figure 3.1 illustrates the potential benefits of regularization. In the case of noiseless data, it is unclear what kind of performance to expect when using regularization techniques.

Unfortunately, there is no obvious way in which to choose the regularization parameter. Possible techniques are cross validation eg. [Golub *et al.*, 1979] or some other principle such as structural risk minimization [Vapnik, 1995]. [Mackay, 1992a], [Mackay, 1992c] describes a method for choosing regularization parameters from the Bayesian point of view, given a prior distribution for the regularization parameter. Unfortunately, exactly what a prior on the regularization parameter would be and how one could obtain it are not well defined. It is shown in [Koistinen, 1997] that if $\lambda = o(n^{-1/2})$, then the gain in the test error is $o(1/n)$ therefore it is crucial to have a reliable method for choosing λ to take advantage of this small potential gain. Conversely, a poor choice for λ could easily have a negative effect.

By restricting the complexity of the learned function, regularization will help the out of sample performance in two distinct cases. When one has some *a priori* belief that the target function is "simple," it might be possible to incorporate this prior belief about the function into the regularization term and thus improve out of sample performance. When noise is present, attempting to fit the noise will produce a learned function that is more complicated than necessary, so restricting the complexity of the learned function should help the out of sample behavior. In this chapter we deal with the latter case – regularization in the presence of noise. We will address the connection between priors and regularization later, in chapter 9

By minimizing the out of sample error (test error), we will derive an analytic expression for the optimal weight decay parameter when the learning model is linear in the parameter space (general linear models). These results generalize the results of [Koistinen, 1997] to multidimensional input. We will show that for general linear models that are well specified, the regularization parameter is $O(\sigma^2/N)$ and the gain in test error is $O(1/N^2)$, analogous to the result in [Koistinen, 1997] for the one dimensional case. For more general target functions, the regularization parameter turns out to be $O(\sigma^2/N) + A_d$ where A_d is related to the approximation degree (how close the target function is to the learning model). A target function outside the learning model has an effect analogous to noise. Here too we will see that the test error gain is $O(1/N^2)$, and that regularization is considerably less useful when there is no noise.

For more general learning models and regularization functions, we develop a technique which we call EXPLOVA (Explanation of Variance)[2] in order to derive an expression for the regularization parameter. Using this technique, we also derive a data dependent condition on the learning model that we expect to be satisfied if a given regularization term is to be helpful, and give examples where we expect regularization to help. We will see that for general linear models, EXPLOVA produces results that are similar to the test error minimization technique. Using EXPLOVA, we find that in general, the regularization parameter is $O(\sigma^2/N)$. Our results prescribe weight decay when the data are noisy but not when the data are noiseless. We demonstrate this experimentally using simulations with neural networks.

[2]This name is partly due to Alexander Nicholson.

The goal of this chapter is two fold: to develop the form of the regularization parameter and to emphasize that its choice depends on the noise level and some prior knowledge about the target function.

This chapter is organized as follows. In section 3.2, we first derive analytic expressions for the weight decay parameter in the case of general linear models. We will then develop EXPLOVA which enables us to derive an expression for the regularization parameter and a condition necessary for regularization to be useful with general learning models. In section 3.3 we discuss the results from section 3.2 and show some experimental verification of them. We will end with some concluding remarks in section 3.4

3.2 The Optimal Regularization Parameter

In this section, we set up the learning problem and then consider two methods for finding the optimal regularization parameter. Assume the standard learning paradigm that has already been defined. A continuous target function $f : \mathbb{R}^d \to \mathbb{R}$ generates the training data set $\mathcal{D}_N$, which consists of N input–output pairs $\{\mathbf{x}_i, y_i\}_{i=1}^N$. Each $\mathbf{x}_i \in \mathbb{R}^d$ is drawn from some input probability measure $dF(\mathbf{x})$.

Additive noise is present in the training data, $y_i = f(\mathbf{x}_i) + \epsilon_i$. We further assume that the noise realizations are independent and zero mean, so

$$\langle \epsilon \mid \mathbf{x} \rangle_\epsilon = \mathbf{0}, \qquad \langle \epsilon\epsilon^T \mid \mathbf{x} \rangle_\epsilon = diag[\sigma_1^2, \sigma_2^2, \ldots, \sigma_N^2] \tag{3.2}$$

(we use $\langle \cdot \rangle$ to denote expectations, $\sigma = [\sigma_1 \sigma_2 \ldots \sigma_N]$, and $diag[\cdot]$ denotes a diagonal matrix).

The functions $g_\omega \in \mathcal{H}$ are parameterized by a weight vector ω. We use mean squared deviation as our error measure, thus, the **training error**, $\mathcal{E}_{tr}$ is defined by[3]

$$\mathcal{E}_{tr} = \frac{1}{N}\sum_{i=1}^{N}(g_\omega(\mathbf{x}_i) - y_i)^2 \tag{3.3}$$

and the **test error**, $\mathcal{E}_N(\sigma)$ is

$$\mathcal{E}_N(\sigma) = \left\langle (f(\mathbf{x}) - g_\omega(\mathbf{x}))^2 \right\rangle_{\epsilon, \mathcal{D}_N, \mathbf{x}} \tag{3.4}$$

where the expectation is over possible training data sets (noise and input points) and the test point. The ultimate goal is the minimization of the test error, $\mathcal{E}_N(\sigma)$. We consider a regularization term of the form $\lambda\Omega(\omega)$, where λ is the regularization parameter and Ω is the regularizing function. The learning algorithm will be to minimize the function

$$\hat{\mathcal{E}} = \mathcal{E}_{tr} + \lambda\Omega$$

When $\lambda = 0$ the algorithm becomes minimization of the training error. Intuitively, for non-zero λ, we minimize the training error but penalize complexity (due to attempting to fit the noise) with the regularization term.

We restrict ourselves to *stable learning systems.* Remember that stable learning systems possess the two properties "continuity" and "unbiasedness". Continuity ensures that "close" data sets are mapped to "close" functions. For two data sets differing only by the addition of zero-mean noise, unbiasedness requires that at every point, the average value (with respect to the noise) of the functions resulting from the noisy data set is equal to the value of the function resulting from the noiseless data set. (refer to chapter 2 definitions 2.2.1 and 2.2.3 for the formal definitions.)

---

[3]Our EXPLOVA analysis can be extended to other forms for the training error without much effort.

3.2.1 General Linear Learning Models

In this section, we will extend the results of [Koistinen, 1997] to weight decay for general linear learning models. We pick the regularization term so as to minimize the test error directly. Our goal is to analyze the (asymptotic) form of the optimal regularization parameter. We will see that for noisy data, the regularization parameter is $O(\sigma^2/N)$. For noiseless data, there is considerably less gain (and potential loss) in using a regularization term.

Define a set of expansion functions $\{\psi_i(\mathbf{x})\}_{i=1}^{S}$. Thus we map the d dimensional input space into an S dimensional space by $\mathbf{z}_k = \psi_k(\mathbf{x})$. We will assume the existence of all expectations necessary for the following analysis to proceed. Our learning model is linearly parameterized by the weight vector ω:

$$g_\omega(\mathbf{x}) = \sum_{i=1}^{S} w_i \psi_i(\mathbf{x}) = \omega \cdot \mathbf{z} \tag{3.5}$$

When $\psi_k(\mathbf{x}) = \mathbf{x}_k, k = 1 \ldots d$ and $\psi_{d+1} = 1$, we have standard linear regression. This framework is applicable to general polynomial regression, Fourier series methods, etc. The target vector is given by

$$\mathbf{y}^T = [y_1 \ldots y_N] = \mathbf{f}^T + \epsilon$$

where $\mathbf{f}^T = [f(\mathbf{x}_1) \ldots f(\mathbf{x}_N)]$. Let

$$\mathbf{Z} = [\mathbf{z}_1 \ldots \mathbf{z}_N]$$

be the transformed input matrix. Let $\hat{\Sigma}_\mathbf{z} = \mathbf{Z}\mathbf{Z}^T/N$ and $\hat{\mathbf{g}} = \sum_{i=1}^{N} \mathbf{z}_i f(\mathbf{z}_i)/N$. Let $\Sigma_\mathbf{z} = \langle \hat{\Sigma}_\mathbf{z} \rangle_\mathbf{x}$ and $\mathbf{g} = \langle \hat{\mathbf{g}} \rangle_\mathbf{x}$.

To make the analysis more presentable, we assume a regularization term of the form

$$\Omega(\omega) = \lambda \omega^T \hat{\Sigma}_\mathbf{z} \omega$$

Then, $\hat{\mathcal{E}}(w) = \mathcal{E}_{tr}(\omega) + \lambda\Omega(\omega)$ is quadratic in ω. By straight forward differentiation with respect to ω, we find that the weight vector minimizing $\hat{\mathcal{E}}$ is given by

$$\omega(\lambda) = \frac{(\mathbf{Z}\mathbf{Z}^T)^{-1}\mathbf{Z}\mathbf{y}}{1+\lambda} = \frac{\omega^*}{(1+\lambda)} \tag{3.6}$$

where $\omega^* = (\mathbf{Z}\mathbf{Z}^T)^{-1}\mathbf{Z}\mathbf{y} = \hat{\Sigma}_{\mathbf{z}}^{-1}(\hat{\mathbf{g}} + \mathbf{Z}\epsilon/N)$. We assume that $\mathbf{g} \neq 0$ else the optimal least squares solution is simply 0 and weight decay will always help. By the law of large numbers, we note that $\mathbf{Z}\mathbf{Z} \underset{N\to\infty}{\longrightarrow} N\Sigma_{\mathbf{z}}$ and $\mathbf{Z}\mathbf{f} \underset{N\to\infty}{\longrightarrow} N\mathbf{g}$, so we write

$$\mathbf{Z}\mathbf{Z}^T = N\Sigma_{\mathbf{z}} + \sqrt{N}\mathbf{V}(\mathbf{Z}), \qquad \mathbf{Z}\mathbf{f} = N\mathbf{g} + \sqrt{N}\mathbf{a}(\mathbf{Z}) \tag{3.7}$$

where $\langle\mathbf{V}\rangle_{\mathbf{Z}} = \langle\mathbf{a}\rangle_{\mathbf{Z}} = 0$ and $\mathrm{Var}(\mathbf{V})$ and $\mathrm{Var}(\mathbf{a})$ are $O(1)$, by the central limit theorem[4]. Using (3.7) and the identity $[1+\lambda\mathbf{A}]^{-1} = 1 - \lambda\mathbf{A} + \lambda^2\mathbf{A}^2 + O(\lambda^3)$, one finds that

$$\hat{\Sigma}_{\mathbf{z}}^{-1} = \Sigma_{\mathbf{z}}^{-1} - \frac{\Sigma_{\mathbf{z}}^{-1}\mathbf{V}\Sigma_{\mathbf{z}}^{-1}}{\sqrt{N}} + \frac{\Sigma_{\mathbf{z}}^{-1}\mathbf{V}\Sigma_{\mathbf{z}}^{-1}\mathbf{V}\Sigma_{\mathbf{z}}^{-1}}{N} + O\left(\frac{1}{N^{3/2}}\right) \tag{3.8}$$

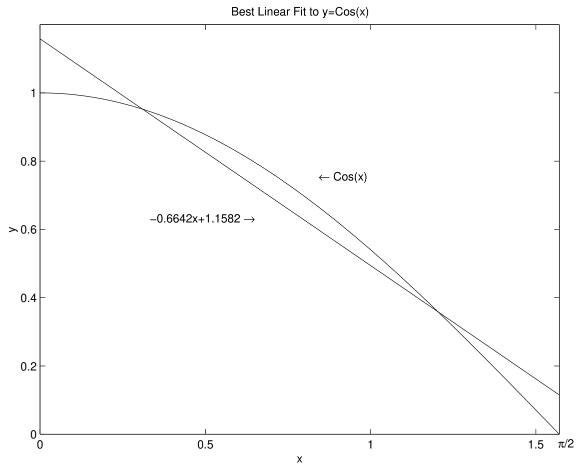
Using (3.5) in (3.4), and differentiating with respect to ω, we find that the best fit in our model is $w_0 = \Sigma\mathbf{g}$. Let the target function $f(\mathbf{z}) = \omega_0^T\mathbf{z} + \eta(\mathbf{z})$. The maximum value of η is a useful measure of how close the target function is to our learning model. Let us define this formally by

Definition 3.2.1 *The approximation degree (ν_d) of the general linear model with respect to the target function and input distribution is given by*

$$\nu_d = \sup_{\mathbf{z}}\{|\eta(\mathbf{z})|\} \tag{3.9}$$

For good learning models, ν_d will be close to zero. For noiseless data, $\langle\omega^*\rangle \approx \omega_0$. More precisely, $\langle\omega^*\rangle$ converges to ω_0 as $N \to \infty$. The rate of this convergence could also be a useful measure. Formally, we define it as follows.

[4]We assume the conditions for this to occur are satisfied.



The best fit to $y = \cos(x)$ for a uniform distribution on $[0, \pi/2]$. ν_d can be calculated to be approximately 0.1582

Figure 3.2: Example fit of a non-linear function by a general linear function.

Definition 3.2.2 *The* <u>*convergence degree* (c_d)</u> *of the general linear model with respect to the target function and input distribution is given by*

$$c_d = \sup_k \left\{ k : \langle \omega^* \rangle = \omega_0 + O\left(\frac{1}{N^k}\right) \right\} \tag{3.10}$$

Examples:

- For a well specified general linear model ($\eta(\mathbf{z}) = 0$), after we get S data points, $\omega^* = \omega_0$ therefore $c_d = \infty$.
- $c_d = 2$ implies that $\langle w^* \rangle = w_0 + O(1/N^2)$.
- The methods of chapter 2 show that in general, $c_d \geq 1$. A quick calculation shows that if $\left\langle \mathbf{V}\Sigma_{\mathbf{z}}^{-1}\mathbf{V} \right\rangle \Sigma_{\mathbf{z}}^{-1}\mathbf{g} - \Sigma_{\mathbf{z}}^{-1} \left\langle \mathbf{V}\Sigma_{\mathbf{z}}^{-1}\mathbf{a} \right\rangle = 0$, then $c_d \geq 3/2$.

Well Specified Learning Model

The target function is given by $f = \omega_0 \cdot \mathbf{z}$. Thus, $\mathbf{y} = \mathbf{Z}^T\omega_0 + \epsilon$, therefore, $\omega^* = \omega_0 + \hat{\Sigma}_{\mathbf{z}}^{-1}\mathbf{Z}\epsilon/N$. Using (3.4) and (3.6), we get for the test error

$$E_{test}(\lambda) = \frac{\lambda^2}{(1+\lambda)^2}\omega_0\Sigma_{\mathbf{z}}\omega_0 + \frac{\sigma^2\mathrm{Tr}[\left\langle \hat{\Sigma}_{\mathbf{z}}^{-1}\Sigma_{\mathbf{z}}\right\rangle]}{N(1+\lambda)^2} \tag{3.11}$$

where we used the facts $\left\langle \epsilon^T\mathbf{A}\epsilon\right\rangle_\epsilon = \sigma^2\mathrm{Tr}[\mathbf{A}]$ and $\left\langle \epsilon^T A\right\rangle = 0$ where A is independent of ϵ. When $\sigma^2 = 0$ it is no surprise that any $\lambda \neq 0$ will hurt. In fact it is easy to convince oneself that for any $\lambda > 0$, $\exists\sigma^2 > 0$ such that regularization with that value of λ will hurt. By setting $\partial_\lambda E_{test} = 0$, we find

$$\begin{aligned} \lambda_{optimal} &= \frac{\sigma^2\mathrm{Tr}[\left\langle \hat{\Sigma}_{\mathbf{z}}^{-1}\Sigma_{\mathbf{z}}\right\rangle]}{N\omega_0\Sigma_{\mathbf{z}}\omega_0} \\ &\overset{(a)}{=} \frac{\sigma^2 S}{N\omega_0\Sigma_{\mathbf{z}}\omega_0} + O\left(\frac{1}{N^2}\right) \\ &\overset{(b)}{\approx} \frac{\sigma^2 S}{N\omega^*\hat{\Sigma}_{\mathbf{z}}\omega^*} + o\left(\frac{1}{N}\right) \end{aligned} \tag{3.12}$$

where (a) follows because $\left\langle \hat{\Sigma}_{\mathbf{z}}^{-1}\Sigma_{\mathbf{z}}\right\rangle = \mathbf{I}_{(d+1)\times(d+1)} + \left\langle \Sigma_{\mathbf{z}}^{-1}\mathbf{V}\Sigma_{\mathbf{z}}^{-1}\mathbf{V}\right\rangle/N + o(1/N)$ and (b) because $\left\langle (\omega^*)^T\hat{\Sigma}_{\mathbf{z}}\omega^*\right\rangle = \omega_0^T\Sigma_{\mathbf{z}}\omega_0 + O(\sigma^2/N)$. Using (3.12) in (3.11) we find for the improvement in the test error, $\Delta E_{test} = E_{test}(\lambda) - E_{test}(0)$,

$$\Delta E_{test} = -\frac{3\sigma^4\mathrm{Tr}[\left\langle \hat{\Sigma}_{\mathbf{z}}^{-1}\Sigma_{\mathbf{z}}\right\rangle]^2}{N^2\omega_0\Sigma_{\mathbf{z}}\omega_0} + o\left(\frac{1}{N^2}\right) = O\left(\frac{\sigma^4}{N^2}\right) \tag{3.13}$$

Technical Note

For input dependent noise that is independent and has zero conditional mean, the test error can be calculated as

$$E_{test} = \frac{\lambda^2}{(1+\lambda)^2}\omega_0\Sigma_{\mathbf{z}}\omega_0 + \frac{\left\langle \sigma^2(\mathbf{z}_i)\mathbf{Z}_{ji}\mathbf{Z}_{il}[\hat{\Sigma}_{\mathbf{z}}^{-1}\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1}]_{lj}\right\rangle}{N(1+\lambda)^2} \tag{3.14}$$

where summation over repeated indices is understood; to $O(1/N)$ this expression becomes

$$E_{test} = \frac{\lambda^2}{(1+\lambda)^2}\omega_0\Sigma_{\mathbf{z}}\omega_0 + \frac{\mathrm{Tr}[\left\langle\sigma^2(\mathbf{z})\mathbf{z}\mathbf{z}^T\right\rangle\Sigma_{\mathbf{z}}^{-1}]}{N(1+\lambda)^2} \tag{3.15}$$

We can then proceed to obtain $\lambda_{optimal}$ in exactly the same way as has already been used above when the noise is input independent.

Misspecified Learning Model

In this case, the target function is not in the learning model. We can still make some progress because the learned hypothesis can be explicitly constructed. The results are not as clean as in section 3.2.1, however, they are strong enough to show that even in this fairly general case, it may not be a good idea to use a regularization term when the data are noiseless.

Using (3.4) and (3.6) the test error can be calculated as

$$E_{test} = \frac{\left\langle\hat{\mathbf{g}}^T\hat{\Sigma}_{\mathbf{z}}^{-1}\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1}\hat{\mathbf{g}} + \frac{\sigma^2}{N}\mathrm{Tr}[\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1}]\right\rangle}{(1+\lambda)^2} - \frac{2\left\langle\hat{\mathbf{g}}^T\hat{\Sigma}_{\mathbf{z}}^{-1}\mathbf{g}\right\rangle}{1+\lambda} + \left\langle f^2\right\rangle$$

We can write this in a form similar to (3.11) as follows

$$E_{test} = \frac{\lambda^2\left\langle\hat{\mathbf{g}}^T\hat{\Sigma}_{\mathbf{z}}^{-1}\mathbf{g}\right\rangle}{(1+\lambda)^2} + \frac{\frac{\sigma^2}{N}tr[\left\langle\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1}\right\rangle] + A_d}{(1+\lambda)^2} + \left\langle f^2\right\rangle - \left\langle\hat{\mathbf{g}}^T\hat{\Sigma}_{\mathbf{z}}^{-1}\mathbf{g}\right\rangle \tag{3.16}$$

where

$$A_d = \left\langle\hat{\mathbf{g}}^T\hat{\Sigma}_{\mathbf{z}}^{-1}\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1}\hat{\mathbf{g}} - \hat{\mathbf{g}}^T\hat{\Sigma}_{\mathbf{z}}^{-1}\mathbf{g}\right\rangle$$

A_d deviates from zero to the extent that $\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1}\hat{\mathbf{g}}$ deviates from $\mathbf{g}$, thus A_d is small and in fact decreases at least as fast as $1/N$. A more detailed analysis of A_d is given in the appendix, where it is shown that $A_d \approx \left\langle\mathrm{Tr}[\eta(\mathbf{z})\mathbf{z}\mathbf{z}^T\Sigma_{\mathbf{z}}^{-1}]\right\rangle$. Setting $\partial_\lambda E_{test}$=0 we solve for $\lambda_{optimal}$.

$$\lambda_{optimal} = \frac{\frac{\sigma^2}{N}\left\langle \mathrm{Tr}[\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1}]\right\rangle + A_d}{\hat{\mathbf{g}}^T\Sigma_{\mathbf{z}}^{-1}\mathbf{g}} \tag{3.17}$$

$$\stackrel{(a)}{\approx} \frac{\sigma^2\left\langle \mathrm{Tr}[\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1}]\right\rangle + \left\langle \mathrm{Tr}[\eta(\mathbf{z})^2\mathbf{z}\mathbf{z}^T\Sigma_{\mathbf{z}}^{-1}]\right\rangle}{N\hat{\mathbf{g}}^T\Sigma_{\mathbf{z}}^{-1}\mathbf{g}} \tag{3.18}$$

where (a) follows from (3.46) in appendix 3.5. Note that $\hat{\Sigma}_{\mathbf{z}}^{-1}\hat{\mathbf{g}} = \omega^*$ so as $\hat{\mathbf{g}}^T\Sigma_{\mathbf{z}}^{-1}\mathbf{g} = w^{*T}\hat{\Sigma}_{\mathbf{z}}\omega^* + O(1/\sqrt{N})$. Notice that the extra term induced by the misspecification is exactly what would arise if instead we added noise to the data with a variance equal to $\eta^2(\mathbf{z})$, dependent on $\mathbf{z}$ (3.15). Using our expression for $\lambda_{optimal}$ in (3.16), we see that the gain in test error performance is

$$\Delta E_{test} = -\frac{3(\frac{\sigma^2}{N}\mathrm{Tr}[\left\langle \hat{\Sigma}_{\mathbf{z}}^{-1}\Sigma_{\mathbf{z}}\right\rangle] + A_d)^2}{\hat{\mathbf{g}}^T\Sigma_{\mathbf{z}}^{-1}\mathbf{g}} + o\left(\frac{1}{N^2}\right) \tag{3.19}$$

which is $O(1/N^2)$. If $c_d > 1$ then it can be shown that $A_d = o(1/N)$ and so can be ignored (in the asymptotic limit) in the expression for $\lambda_{optimal}$. Otherwise A_d can be bounded in terms of ν_d, therefore, for good learning models, A_d will be small.

To summarize the results of this section, we see that in the presence of noise, the optimal weight decay parameter is $O(\sigma^2/N)$. In fact, any $\lambda \leq \lambda_{optimal}$ will decrease the expected test error. Further, when there is no noise, we expect that regularization is ineffective.

3.2.2 Explanation of Variance (EXPLOVA)

The regularization term decreases the complexity of the learned hypothesis. The test error usually improves because the variance of the new classifier decreases. This compensates for the extra variance "learned" due to the noise in the data. We try to exploit the fact that with noise, the learned function has higher variance in order to deduce some properties that the regularizer should have.

We suppose that $\mathcal{E}_{tr}$ is minimized at ω^*. Due to the addition of the regularization term, the minimum of $\hat{\mathcal{E}}$ will be shifted from ω^* by an amount $\Delta\omega$. This produces a change in the variance of the output for the learned function. The presence of noise adds variance to the targets. This implies that the variance of the learned function will be higher than it should be, in proportion to the level of noise. The presence of the regularization term should offset the variance gained due to the noise, and this change should be related to the noise level. We now quantify this as follows.

Initially the learned function was trained to "explain" as much variance in the data as it could. It is well understood, that the presence of noise increases the test error, [Barron, 1984],[Eubank, 1988],[Akaike, 1970]. For linear models, and the input distribution being delta functions at the observed data points,

$$\mathcal{E}_N(\sigma) = \mathcal{E}_{tr} + \frac{2\overline{\sigma^2}(d+1)}{N} \tag{3.20}$$

where $\overline{\sigma^2} = \sum_i \sigma_i^2/N$. [Moody, 1991] derives a similar result for more general learning models with $(d+1)$ replaced by p_{eff}, an effective number of parameters for the learning system. [Amari and Murata, 1991] shows that a similar result holds when descending using backpropagation on the training error.

Due to the presence of noise, we suppose that the test error increases by an amount $\Delta E(\sigma, N)$. Expanding the expression for the test error we have

$$\begin{aligned}
\mathcal{E}_N(\sigma) &= \left\langle (f-g)^2 \right\rangle_{\epsilon,\mathcal{D}_N,\mathbf{x}} \\
&= \left\langle f^2 \right\rangle_x - 2\left\langle f \left\langle g \right\rangle_\epsilon \right\rangle_{\mathcal{D}_N,\mathbf{x}} + \left\langle g^2 \right\rangle_{\epsilon,\mathcal{D}_N,\mathbf{x}}
\end{aligned}$$

The first term is independent of noise, and the second term is equal to the term that would replace it if there were no noise (by the unbiasedness assumption). Therefore $\langle g^2 \rangle$ must increase by $\Delta E(\sigma, N)$. The presence of noise increases the complexity of the learned function by increasing $\langle g^2 \rangle_{\epsilon,\mathcal{D}_N,\mathbf{x}}$.

In the previous chapter, we saw that for stable learning systems (theorem 2.3.2)

$$\Delta E(\sigma, N) \leq \frac{\mathcal{C}\overline{\sigma^2}}{N} + O(\frac{1}{N^2}) \tag{3.21}$$

where $\mathcal{C}$ is a constant that depends on the target function, input distribution and learning system. For linear models[5], $\mathcal{C} = (d+1)$. We would like regularization to enable us to perform as well as we would were there no noise present in the data. Unfortunately all we know is that the noise induces an increase in $\langle g^2 \rangle$. Thus we know that to be performing as well as in the noiseless case, $\langle g^2 \rangle$ should decrease by $\Delta E(\sigma, N)$. This is a necessary requirement, but it is not sufficient. In a sense, we are using the fact that $\langle g^2 \rangle$ should decrease as a hint [Abu-Mostafa, 1995]. We enforce this "hint" not because it guarantees getting closer to the noiseless performance, but because if we are getting closer to the noiseless performance, then the hint will be obeyed. Furthermore, $\langle g \rangle$ should remain unchanged due to the regularizer, because the addition of noise did not change $\langle g \rangle$ in the first place.

The general approach is as follows. The training error will be minimized at some weight vector ω^*. If the regularization parameter is small, then the regularization term will be small. The introduction of the regularization term will cause the minimum to shift to $\omega^* + \Delta\omega$, where $\Delta\omega$ will be small. This change in ω will lead to a change in $\langle g_\omega^2 \rangle$ which we will equate to $\Delta\mathcal{E}(\sigma, N) \approx \mathcal{C}\overline{\sigma^2}/N$.[6]

By taking the Taylor expansion of $\hat{\mathcal{E}}(\omega)$ about ω^* up to second order in $\Delta\omega = \omega - \omega^*$, and using the fact that $\nabla_{\omega^*}\mathcal{E}_{tr} = 0$, we find that

$$\hat{\mathcal{E}}(\omega) = \mathcal{E}_{tr}(\omega^*) + \lambda\Omega(\omega^*) + \frac{1}{2}\Delta\omega^T[H_{\omega^*}(\mathcal{E}_{tr}) + \lambda H_{\omega^*}(\Omega)]\Delta\omega + \lambda\Delta\omega^T\nabla_{\omega^*}\Omega \tag{3.22}$$

where the Hessian $[H_{\omega^*}(\cdot)]_{ij} = \partial_{\omega_i}\partial_{\omega_j}(\cdot)\mid_{\omega^*}$ is the matrix of second derivatives. $H_{\omega^*}(\mathcal{E}_{tr})$ is positive definite and can be calculated from the data given the learning model and ω^* (which can be obtained by optimizing the training error). Setting $\nabla_{\Delta\omega}\hat{\mathcal{E}}(\omega) = 0$

[5] see theorem 2.3.1 in the appendix

[6] The analysis can also be carried out for other error measures if one can evaluate $\Delta\mathcal{E}(\sigma, N)$

we find for the change in the weight vector induced by the regularization term

$$\Delta\omega \approx -\lambda \left[H_{\omega^*}(\mathcal{E}_{tr}) + \lambda H_{\omega^*}(\Omega)\right]^{-1} \nabla_{\omega^*}\Omega \tag{3.23}$$

$$= -\lambda H_{\omega^*}^{-1} \nabla_{\omega^*}\Omega + O(\lambda^2) \tag{3.24}$$

where $\nabla_{\omega^*}\Omega \neq 0$.

We will ignore the $O(\lambda^2)$ term as λ will be small, and write H_{ω^*} for $H_{\omega^*}(\mathcal{E}_{tr})$. Let

$$V(g_{\omega^*}) \equiv \frac{1}{N}\sum_{i=1}^{N} g_{\omega^*}(\mathbf{x}_i)^2, \qquad M(g_{\omega^*}) \equiv \frac{1}{N}\sum_{i=1}^{N} g_{\omega^*}(\mathbf{x}_i) \tag{3.25}$$

be sample estimates of $\langle g_{\omega^*}^2 \rangle$ and $\langle g_{\omega^*} \rangle$. Equating the change in $V(g_{\omega^*})$ to $-\mathcal{C}\sigma^2/N$ we can deduce what λ should be. The change in $V(g_{\omega^*})$ is approximately given by $\Delta V(g_{\omega^*}) \approx \Delta\omega \cdot \nabla_{\omega^*} V(g_{\omega^*})$. Using (3.24) and differentiating the expression for $V(g_{\omega^*})$, we have for the change in $V(g_{\omega^*})$

$$\Delta V(g_{\omega^*}) = -\frac{2\lambda}{N}\left(H_{\omega^*}^{-1}\nabla_{\omega^*}\Omega\right) \cdot \left(\sum_{m=1}^{N} g(\mathbf{x}_m)\nabla_{\omega^*} g(\mathbf{x}_m)\right) + O(\lambda^2) \tag{3.26}$$

Similarly, one can evaluate $\Delta \left\langle g^2 \right\rangle_{\epsilon,\mathcal{D}_N,\mathbf{x}}$

$$\Delta \left\langle g^2 \right\rangle_{\epsilon,\mathcal{D}_N,\mathbf{x}} = -2\lambda \left(H_{\omega^*}^{-1}\nabla_{\omega^*}\Omega\right) \cdot \left\langle g \nabla_{\omega^*} g \right\rangle_{\epsilon,\mathcal{D}_N,\mathbf{x}} + O(\lambda^2) \tag{3.27}$$

Equating this change to $-\mathcal{C}\overline{\sigma^2}/N$ we have

$$\lambda = \frac{\mathcal{C}\overline{\sigma^2}/N}{2\left(H_{\omega^*}^{-1}\nabla_{\omega^*}\Omega\right) \cdot \left\langle g\nabla_{\omega^*} g\right\rangle_{\epsilon,\mathcal{D}_N,\mathbf{x}}} + O(\frac{1}{N^2}) \tag{3.28}$$

$$\approx \frac{\mathcal{C}\overline{\sigma^2}/N}{\frac{2}{N}\left(H_{\omega^*}^{-1}\nabla_{\omega^*}\Omega\right) \cdot \left(\sum_{m=1}^{N} g(\mathbf{x}_m)\nabla_{\omega^*} g(\mathbf{x}_m)\right)} + O(\frac{1}{N^2}) \tag{3.29}$$

Equating the change in $M(g_{\omega^*})$ to zero (due to unbiasedness), the following condition must be satisfied

$$\left(H_{\omega^*}^{-1}\nabla_{\omega^*}\Omega\right)\cdot\left(\sum_{m=1}^{N}\nabla_{\omega^*}g(\mathbf{x}_m)\right)+O(\lambda^2)=0 \tag{3.30}$$

This condition can usually be satisfied by a linear transformation of the inputs. This will become clearer when we consider linear models as an example.

We conclude from (3.29) that the regularization term is $O(\frac{\sigma^2}{N})$. For a positive regularization parameter to be consistent with a decrease in variance, the necessary condition is that

$$\left(H_{\omega^*}^{-1}\nabla_{\omega^*}\Omega\right)\cdot\langle g\nabla_{\omega^*}g\rangle_{\epsilon,\mathcal{D}_N,\mathbf{x}}\geq 0 \tag{3.31}$$

hold at $\omega = \omega^*$. Since the regularization term should be applicable to any target function (so, for all ω^*), it is necessary that this inequality be true for all ω. This is a condition that has to be satisfied by a given regularization term and is not independent of the learning model – a regularization term will be useful for a certain learning model, but perhaps not for another – for example, consider two identical learning models, one parameterized by ω, the other parameterized by $1/\omega$; if weight decay helps in one case, it will not help in the other.

Given only the data, (3.31) is not verifiable, however the sample version

$$\left(H_{\omega^*}^{-1}\nabla_{\omega^*}\Omega\right)\cdot\left(\sum_{m=1}^{N}g(\mathbf{x}_m)\nabla_{\omega^*}g(\mathbf{x}_m)\right)\geq 0 \tag{3.32}$$

can be directly verified at ω^*, as ω^* can be obtained by optimizing the training error. We note that using $V(g_{\omega^*})$ itself as the regularization term is guaranteed to satisfy the consistency condition.

The expression for the regularization parameter λ (3.29) is calculated as follows. Optimize the training error to obtain ω^*. Given ω^*, the term (3.32) can be calculated using the data points. This can then be used to evaluate (3.29) provided that estimates for $\overline{\sigma^2}$ and $\mathcal{C}$ are available. Descending on $\hat{\mathcal{E}}$, starting from ω^*, we then arrive at the final regularized solution.

EXPLOVA is summarized as follows. Noise is present in the data set, and the noise level is known. Further, an effective model complexity or an effective number of parameters is known. This knowledge is summarized in the constant $\mathcal{C}$, which in general may depend on the target function, input distribution and the learning model. We then calculate the value of the regularization parameter by equating the the variance gain in the function due to the presence of noise and the variance loss due to the presence of the regularization term. EXPLOVA is more a philosophy than a specific method, applicable to other techniques. We stress that the noise level needs to be known, because, for given a regularization term, one can find a noise level small enough such that regularization will hurt.

We conclude this section by specializing the preceding analysis to the case of weight decay. For simplicity, we present the analysis for the weight decay term

$$\Omega(\omega) = \frac{1}{2}\omega^T H_{\omega^*}\omega$$

where H_{ω^*} can be computed by first optimizing the training error. In this case, $\nabla_{\omega^*}\Omega(\omega^*) = H_{\omega^*}\omega^*$, therefore using (3.24)

$$\Delta\omega \approx -\lambda\omega^*$$

Using (3.29), we find for the regularization parameter

$$\lambda \approx \frac{\mathcal{C}\sigma^2/N}{\frac{2}{N}\omega^* \cdot \left(\sum_{m=1}^{N} g(\mathbf{x}_m)\nabla_{\omega^*} g(\mathbf{x}_m)\right)} \tag{3.33}$$

so, for a positive decay parameter to generate a decrease in variance, the necessary condition is that

$$\omega^* \cdot \left(\sum_{m=1}^{N} g(\mathbf{x}_m)\nabla_{\omega^*} g(\mathbf{x}_m)\right) \geq 0 \tag{3.34}$$

and (3.30) becomes $\omega^* \cdot \sum_{m=1}^{N} \nabla_{\omega^*} g(\mathbf{x}_m) = 0$. As already pointed out, this can usually be achieved by transforming the input space. (3.34) should hold for all ω^*. This is a strong condition on the learning model. In expectation this becomes $\omega^* \cdot \langle g \nabla_{\omega^*} g \rangle \geq 0$ for all ω^*.

Examples

- *General Linear Learning Model*

  For general linear learning models, the training error is given by

$$\mathcal{E}_{tr} = \frac{1}{N} \sum_{i=1}^{N} (\omega^T \mathbf{z}_i - y_i)^2$$

  from which we see that $H_{\omega^*} = 2\hat{\Sigma}$ where $\hat{\Sigma}_{\mathbf{z}} = \mathbf{Z}\mathbf{Z}^T/N$ and we use the definition $\mathbf{Z}\mathbf{Z}^T = \sum_{m=1}^{N} \mathbf{z}_m \mathbf{z}_m^T / N$. Let $\Sigma_{\mathbf{z}} = \left\langle \hat{\Sigma}_{\mathbf{z}} \right\rangle$. The consistency condition (3.34) and its expectation become

$$(\omega^*)^T \hat{\Sigma}_{\mathbf{z}} \omega^* \geq 0 \qquad (\omega^*)^T \Sigma_{\mathbf{z}} \omega^* \geq 0$$

  which are both satisfied for all ω^* because $\hat{\Sigma}_{\mathbf{z}}$ and $\Sigma_{\mathbf{z}}$ are positive semi-definite matrices. The mean preserving condition (3.30) becomes $\omega^* \cdot \sum_{i=1}^{N} \mathbf{z}_i = 0$. This can be achieved by an initial translation of variables such that $\sum_{i=1}^{N} \mathbf{z}_i = 0$. (Note that this will change $\hat{\Sigma}_{\mathbf{z}}$). The regularization term becomes

$$\frac{1}{2} \frac{\sigma^2 S}{N \omega^{*T} \hat{\Sigma}_{\mathbf{z}} \omega^*} w^T \hat{\Sigma}_{\mathbf{z}} \omega \tag{3.35}$$

- $g_\omega(\mathbf{x}) = \phi(\omega \cdot \mathbf{x})$

  If $\phi' \geq 0$, and $x\phi(x) \geq 0$, then

$$\omega \cdot g \nabla_\omega g = \omega \cdot \mathbf{x} \phi(\omega \cdot \mathbf{x}) \phi'(\omega \cdot \mathbf{x}) \geq 0$$

  and so the consistency condition is satisfied. eg. $\phi(t) = \tanh(t)$

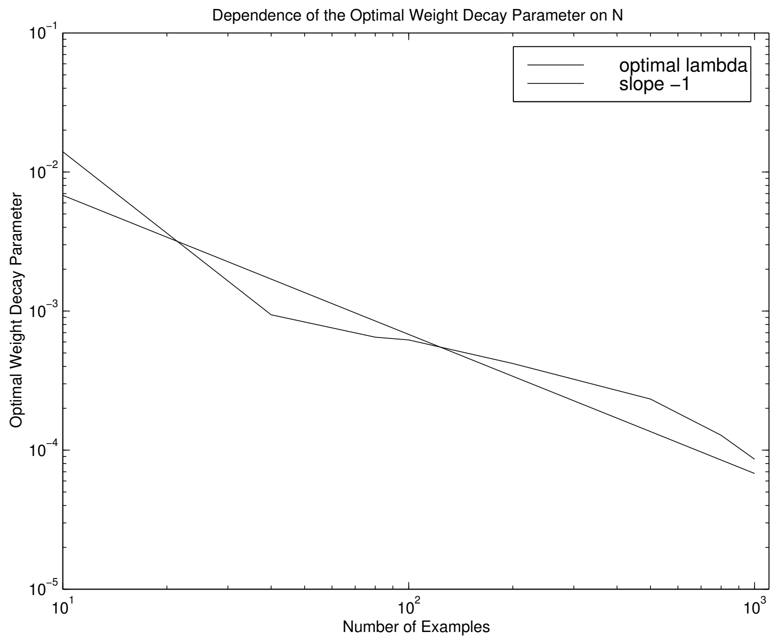
3.2.3 Summary of Theoretical Results

We have demonstrated that weight decay in the presence of noise can yield improved results. In the noiseless case, we do not expect weight decay be of significant help. Further, weight decay may hurt, especially if the target function is 'almost' in learning model (section 3.2.1).

The optimal regularization term for linear models (calculated by directly minimizing the test error) is similar to the EXPLOVA result. EXPLOVA can be applied to more general learning models and more general regularization terms. The algorithm would be to first optimize the training error to get ω^*. Using ω^* one can check (for the learning model at hand) whether the regularization term is expected to help (3.34). If so, with an estimate for the noise level, one can construct the regularization term. We emphasize that without an estimate of the noise level it is not possible to systematically choose a regularization parameter in the sense that for any regularization parameter, if the noise is small enough, regularization will hurt. Further, one also needs an estimate for the parameter $\mathcal{C}$, an effective model complexity.

3.3 Experiments

Using a model that is non-linear in the parameter space, we illustrate the dependence of $\lambda_{optimal}$ in the noisy case (Figure 3.3). The dependence is consistent with the predicted $\frac{1}{N}$ behavior, which is readily derivable from the methods of section 3.2.2 but not evident using the techniques of section 3.2.1 . Figure 3.4 shows the test error profile for the noiseless and noisy cases. Note that the behavior in the noiseless case does not clearly imply the existence of an optimal λ, where as the noisy case does. The simulations indicate that when $\sigma^2 > 0$, there exists a regularization term that will improve the test error, but conversely, given any regularization term, there is a scenario in which the test error will be increased by the regularizer as is evidenced by the fact that for no noise the test error is an increasing function of λ. Thus in order to have effective regularization, one needs some priors on the noise level.



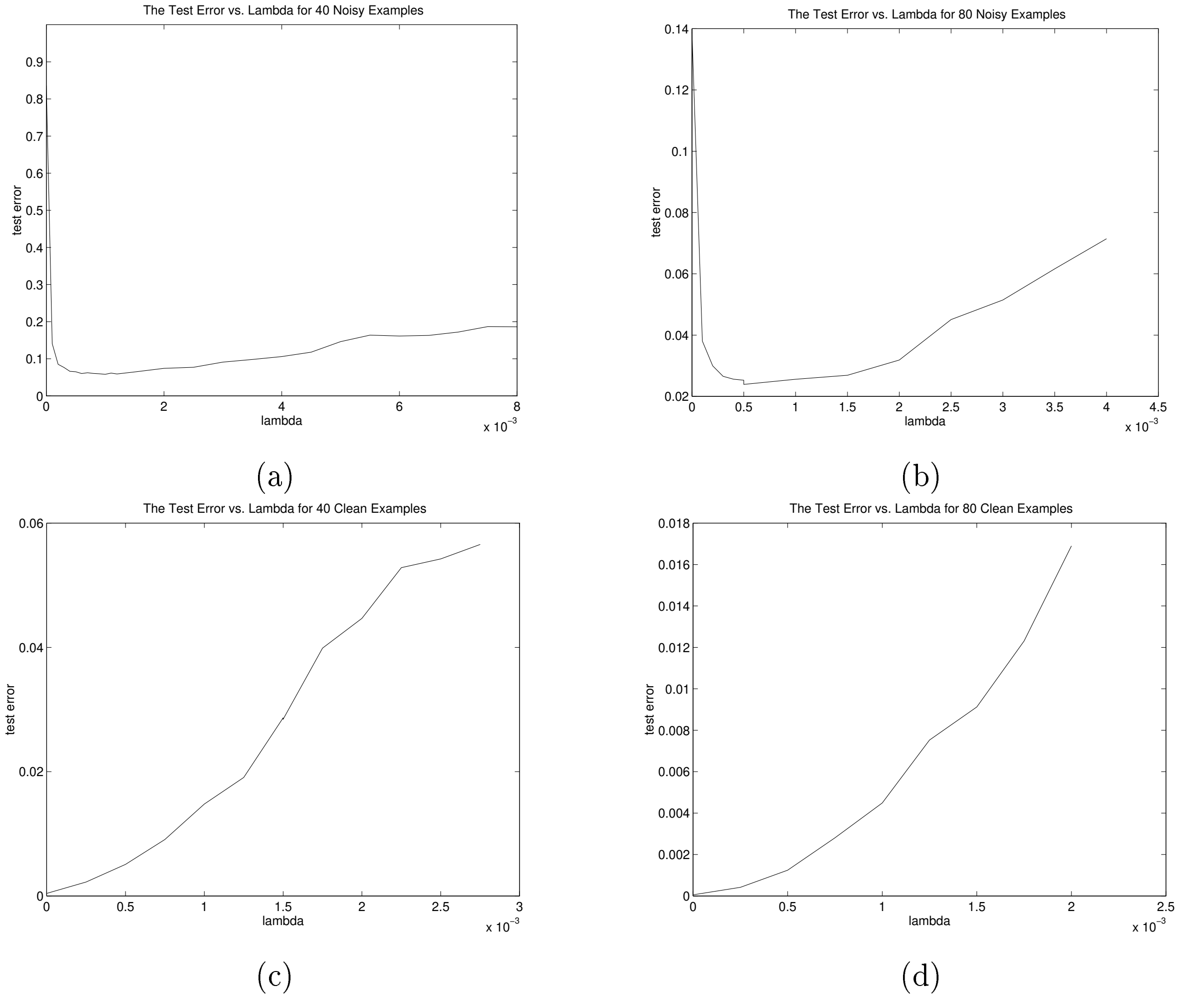
The experiments were run for a teacher 2-5-1 neural network and student 2-10-1 with σ^2=0.3^2. All hidden units were tanh. The expected test error for λ was calculated over 1000 runs and then for each N the lambda minimizing this is used. A slope -1 line is shown for comparison with the expected $\frac{1}{N}$ decrease which should have slope -1 on this LogLog plot.

Figure 3.3: Illustration of the behavior of $\lambda_{optimal}$ on N.

3.4 Comments

We analyzed regularization (specifically weight decay) for general linear models (G.L.M's). We demonstrated that when the data is noiseless, regularization is of limited use (especially for $c_d > 1$ or $\nu_d \ll 1$), contrary to the idea that regularization will help by curtailing the model complexity. If there is nothing to over fit, then the model will not over fit. Our experimental verification in the more general case was also consistent with this prediction – regularization is of limited help when no noise is present.

For G.L.M.'s, we analyzed the form of the regularization parameter and found an explicit formula for $\lambda_{optimal} = O(\frac{\sigma^2+n^2}{N})$, where the n^2 arises from the component of the target function outside the learning model when the G.L.M. is misspecified – the component of the target function outside the learning model has an effect equivalent



The top two graphs ((a) and (b)) show the dependence of the test error in the presence of noise ($\sigma^2 = 0.3^2$) when training a 2-10-1 network on data produced by a 2-5-1 network. All hidden units were tanh. We see the presence of a best $\lambda > 0$. The lower two figures ((c) and (d)) show the analogous results when the data are noiseless. The results are strikingly consistent with the conclusions of section 3.2.1 even though the learning model is non linear.

Figure 3.4: Test error dependence on λ with and without noise.

to input dependent noise[7].

We deduced a technique (EXPLOVA) for testing the effectiveness of regularization, which relied upon using the 'hint' that the target function is simpler than indicated by the variance of the targets . This technique offers hope for being able to choose a regularization parameter in some systematic way, given an estimate for the noise level. In the presence of noise, EXPLOVA suggests a regularization parameter of the form

$$\lambda \approx \frac{\mathcal{C}\overline{\sigma^2}/N}{\frac{2}{N}\left(H_{\omega^*}^{-1}\nabla_{\omega^*}\Omega\right)\cdot\left(\sum\limits_{m=1}^{N} g(\mathbf{x}_m)\nabla_{\omega^*}g(\mathbf{x}_m)\right)} \tag{3.36}$$

which is applicable to general learning models and regularization terms. Further, EXPLOVA furnishes a method for evaluating whether or not a given regularization term is useful for a particular learning model. We believe that the philosophy embedded in EXPLOVA bears promise – the knowledge of the noise level indicates that the targets have a higher variance than the true function, therefore, we should incorporate this knowledge by requiring that our learned function have smaller variance. Further theoretical and experimental investigation along these lines seems promising.

We emphasize the need for a prior belief on the noise level by noting that there is no universally good regularizing scheme – for any given regularizer, there is a level of noise small enough (where now we treat a target function outside the learning model as a noisy target function in the learning model) such that the regularization will hurt. The need for an estimate of the noise and some prior information becomes apparent by taking the Bayesian approach. Suppose that we wish to maximize the posterior probability that the hypothesis, (g), is the target function, (f), given the data set.

$$P(g=f|\mathcal{D}_N) \propto P(\mathcal{D}_N|g=f)P(g=f) \tag{3.37}$$

[7]Using the methods in chapter 2, it is possible to show that the component of the target function outside the linear learning model has an effect on convergence similar to input dependent noise as well, thus we see that in many respects, a target function outside the learning model behaves very much like a target function inside the learning model plus some additive noise.

Assume a Gaussian noise model. Then maximizing the logarithm of (3.37) leads to the equivalent optimization problem

$$\min_g \left\{ \sum_i (g(\mathbf{x}_i) - y_i)^2 - \sigma^2 \log(P(g = f)) \right\}$$

We have the familiar form of an empirical risk plus a regularization term $-\sigma^2 \log(P(g = f))$. We see that the size of the regularization term depends on the estimate σ^2. We also see that the relative weight of the regularization term to the empirical risk is decreasing$\sim 1/N$, in agreement with our results for more general noise. We have looked at a specific prior, a delta function at the target function. Notice that our results depended on parameters related to the target function. In general, priors will have to be incorporated into the learning process. We consider this issue in chapter 9.

The gains due to regularization are $O(1/N^2)$ for the *optimal* regularization parameter, so it is important to have an accurate estimate for $\lambda_{optimal}$ in order to avoid potential disaster. To this end, we need σ^2 and A_d (for G.L.M.'s) or σ^2 and $\mathcal{C}$ (for EXPLOVA). These quantities are priors, thus, **effective regularization requires the use of some prior information**.

3.5 Appendix

We use the notation

$$\mathbf{Z} = [\, \mathbf{z}_1 \quad \mathbf{z}_2 \quad \ldots \quad \mathbf{z}_N \,], \qquad \mathbf{y} = \mathbf{f} + \epsilon$$

$$\Sigma_{\mathbf{z}} \equiv \left\langle \mathbf{z}\mathbf{z}^T \right\rangle_{\mathbf{z}}, \qquad \mathbf{g} = \langle \mathbf{z} f(\mathbf{z}) \rangle_{\mathbf{z}}$$

The law of large numbers gives us that $\mathbf{Z}\mathbf{Z}^T \underset{N\to\infty}{\longrightarrow} N\Sigma_{\mathbf{z}}$ and $\hat{\mathbf{g}} = \mathbf{Z}\mathbf{f} \underset{N\to\infty}{\longrightarrow} N \langle \mathbf{z} f(\mathbf{z}) \rangle_{\mathbf{z}} = \mathbf{g}$. where we assume that the conditions for this to happen are satisfied. $g \in \mathcal{H} \Leftrightarrow g(\mathbf{z}) = \mathbf{z}^T\omega$. The Least Squares estimate of ω is given by

$$\hat{\omega}^* = (\mathbf{Z}\mathbf{Z}^T)^{-1}\mathbf{Z}\mathbf{y} \tag{3.38}$$

We assume the conditions for the central limit theorem to apply are satisfied, so we write

$$\mathbf{Z}\mathbf{Z}^T = N\Sigma_{\mathbf{z}} + \sqrt{N}\mathbf{V}(\mathbf{Z}), \qquad \mathbf{X}\mathbf{f} = N\mathbf{g} + \sqrt{N}\mathbf{a}(\mathbf{Z}) \tag{3.39}$$

where $\langle \mathbf{V} \rangle_{\mathbf{Z}} = \langle \mathbf{a} \rangle_{\mathbf{Z}} = 0$ and $\mathrm{Var}(\mathbf{V})$ and $\mathrm{Var}(\mathbf{a})$ are $O(1)$ by the central limit theorem. Using (3.7) and the identity $[1 + \lambda\mathbf{A}]^{-1} = 1 - \lambda\mathbf{A} + \lambda^2\mathbf{A}^2 + O(\lambda^3)$ one finds that

$$\hat{\Sigma}_{\mathbf{z}}^{-1} = \Sigma_{\mathbf{z}}^{-1} - \frac{\Sigma_{\mathbf{z}}^{-1}\mathbf{V}\Sigma_{\mathbf{z}}^{-1}}{\sqrt{N}} + \frac{\Sigma_{\mathbf{z}}^{-1}\mathbf{V}\Sigma_{\mathbf{z}}^{-1}\mathbf{V}\Sigma_{\mathbf{z}}^{-1}}{N} \tag{3.40}$$

3.5.1 Properties of A_d

Here, we will present a detailed analysis of A_d, given by

$$A_d = \left\langle \hat{\mathbf{g}}^T\hat{\Sigma}_{\mathbf{z}}^{-1}\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1}\hat{\mathbf{g}} - \hat{\mathbf{g}}^T\hat{\Sigma}_{\mathbf{z}}^{-1}\mathbf{g} \right\rangle \tag{3.41}$$

Using (3.40) and (3.39), we find that

$$A_d = \frac{\left\langle 2\mathbf{g}^T\Sigma_{\mathbf{z}}^{-1}\mathbf{V}\Sigma_{\mathbf{z}}^{-1}\mathbf{V}\Sigma_{\mathbf{z}}^{-1}\mathbf{g} + \mathbf{a}^T\Sigma_{\mathbf{z}}^{-1}\mathbf{a} - 3\mathbf{g}^T\Sigma_{\mathbf{z}}^{-1}\mathbf{V}\Sigma_{\mathbf{z}}^{-1}\mathbf{a} \right\rangle}{N} + O\left(\frac{1}{N^{3/2}}\right) \tag{3.42}$$

We immediately see that A_d is $O(1/N)$. We can develop a more intuitive understanding of A as follows. Let $f(\mathbf{z}) = \mathbf{z}^T\omega_0 + \eta(\mathbf{z})$, where the best fit ω_0 is given by $\omega_0 = \Sigma_{\mathbf{z}}^{-1}\mathbf{g} = \Sigma_{\mathbf{z}}^{-1}\langle \mathbf{z}f(\mathbf{z})\rangle$. Then consistency requires that $\Sigma_{\mathbf{z}}^{-1}\langle \mathbf{z}\eta(\mathbf{z})\rangle = 0$, so $\langle \mathbf{z}\eta(\mathbf{z})\rangle = 0$. Let

$$\hat{\mathbf{g}} = \frac{\mathbf{Zy}}{N} = \hat{\Sigma}_{\mathbf{z}}\omega_0 + \xi \tag{3.43}$$

where we have defined $\xi = \mathbf{Z}\eta/N = \sum_i \mathbf{z}_i\eta(\mathbf{z}_i)$ is an $S \times 1$ vector. $\langle\xi\rangle = 0$ and the covariance matrix for ξ can be calculated as

$$\left\langle \xi\xi^T \right\rangle = \frac{\left\langle \mathbf{z}\mathbf{z}^T\eta^2(\mathbf{z})\right\rangle_{\mathbf{z}}}{N} \tag{3.44}$$

The expression for A_d can now be reduced to

$$A_d = \left\langle \xi^T\hat{\Sigma}_{\mathbf{z}}^{-1}\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1}\xi \right\rangle + \left\langle \omega_0\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1}\xi \right\rangle \tag{3.45}$$

We ignore the second term because it only involves ξ. Also by construction, ω_0 is approximately orthogonal to ξ so if $\Sigma_{\mathbf{z}}\hat{\Sigma}_{\mathbf{z}}^{-1} \approx \mathbf{I}$ then this is justified[8]. The first term can be simplified by taking the trace. To order $1/N$, we then find

$$A_d = \frac{\mathrm{Tr}[\left\langle \eta(\mathbf{z})^2\mathbf{z}\mathbf{z}^T\right\rangle \Sigma_{\mathbf{z}}^{-1}]}{N} \tag{3.46}$$

Notice that this is equivalent to the term that would arise were there input dependent noise with noise variance $\eta(\mathbf{z})^2$.

[8]Technically, one can show that the $O(1/N)$ coefficient is $\omega_0^T\left\langle \mathbf{V}\Sigma_{\mathbf{z}}^{-1}\alpha\right\rangle$ where alpha is a random variable with the same statistics as $\sqrt{N}\xi$. However, we expect that $\mathbf{V}$ should be uncorrelated with α, so we can ignore this term.

Chapter 4

The Maximum Likelihood Method and the Prediction of Variance in Time Series

— *The objective world simply* is; *it does not happen.*

–*Herman Weyl*

The previous two chapters illustrate that the noise variance plays an important role in the behavior of learning systems. A natural next step would be to consider the estimation of the noise variance. Specifically we consider the general problem of predicting variance in time series and the use of maximum likelihood as a tool for choosing between any two variance predictors.

What can one say about the prediction of variance? In general nothing (NNFL, chapter 8), unless we make some assumptions about the underlying deterministic behavior. We will assume that the mean is known for theoretical purposes. Practically, this requires that the mean can be learned efficiently.

4.1 Introduction

Consider the time series depicted in figure 4.1. Each point $x(t)$ is drawn from a distribution whose mean is $\mu(t)$ and variance is $\sigma^2(t)$. While we can only observe $x(t)$, we wish to learn about the mean $\mu(t)$ and the variance $\sigma^2(t)$. An accurate prediction of the mean tells us about the expected behavior of the time series. An accurate prediction of the variance is also important, for example, in financial time series, the volatility (a normalized version of $\sigma(t)$) is a tradeable quantity. Typically, the variance serves to place an error bar on the predicted value.

The question arises as to how one can judge various models that are predicting

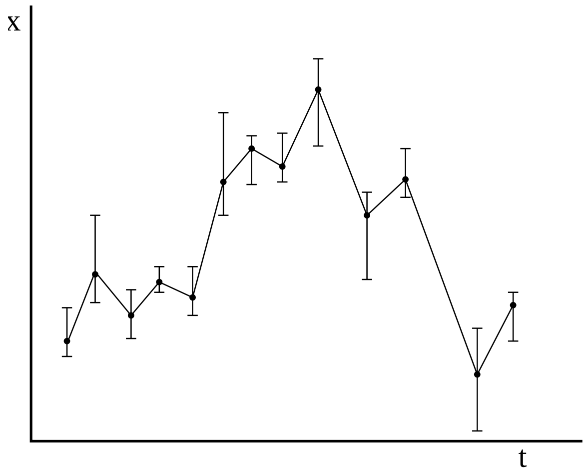


Figure 4.1: A sample time series

a non-explicit parameter like the variance. The variance falls into the class of non-explicit parameters because, on drawing a random variable from its distribution, no direct information on the variance is conveyed. Depending on the error measure one wishes to use, the value of the random variable gives direct information on the 'central' value - for example it is the best estimate of the mean using squared error, or the best measure of the median using absolute error. For this reason it is possible for a model to "learn" the mean or median by "training" on the actual value of the random variable using one of these error functions. If we have more than one drawing from the distribution then some direct information does exist on the variance but we would like to consider time series with time-varying distributions, so we only get one drawing from each distribution. As a result, one has to somehow infer information on the variance, and here lies the difficulty with variance prediction. The task of mean and variance prediction now becomes a learning problem. We discuss how this can be done using maximum likelihood, and we take the special case of Gaussian noise to illustrate our points.

The most striking result we will demonstrate is that, for any finite number of data points, it is more likely than not that we will pick the worse of two specific models if we use the likelihood function to compare them. It turns out that maximum likelihood will lead to an *underestimation* of $\sigma(t)$, even when the mean is predicted perfectly. This naturally leads to the question "can we correct for the systematic underestimation?". Such a correction factor would allow us to choose from a class of models using maximum likelihood and then correct for the bias in the method of selection. We will derive the probability distribution for this under-prediction factor and discuss various correction factors. We consider models that

$$I_{t-\Delta t} \longrightarrow \boxed{\begin{array}{c}\text{black box}\\ \text{predictor}\end{array}} \longrightarrow \hat{\mu}, \hat{\sigma}$$

Figure 4.2: A learning system

predict the mean and variance at time t as in figure 4.2. A variety of such techniques exist ([Hamilton, 1994], [Weigend and Gershenfeld, 1993], [Montgomery *et al.*, 1990], [Engle, 1982], [Bollerslev, 1986]. One would like to have a reliable method for choosing between different models that are predicting $\sigma(t)$. Our goal is the selection of the optimal model given a number of models. Training can be viewed as a generalization of this where one chooses from a *class* of models (e.g., neural networks with a given architecture). We will evaluate maximum likelihood as a selection criterion between models.

We describe the time series by some underlying time varying distribution, $f(x \mid I_{t-\Delta t})$, which is the conditional probability density for obtaining a value x at time t given the information available at $t - \Delta t$. We are interested in $\langle x \rangle_f$, a prediction for the outcome and $\sigma^2 = \left\langle (x - \langle x \rangle_f)^2 \right\rangle_f$, the variance. $\langle \cdot \rangle_f$ denotes expectation with respect to $f(x \mid I_{t-\Delta t})$.

This chapter is organized as follows. Next, we will briefly discuss some financial motivation before moving on to an analysis of the maximum likelihood method where we will discuss some of its problems. We will then consider the possibility of compensating for the maximum likelihood problems.

4.1.1 Financial Motivation

Volatility factors into a number of equations in finance. [Black and Scholes, 1973] derived option pricing models for which the expected future volatility is an important input. [Kat, 1993] has shown that more accurate volatility prediction will improve the replication efficiency of delta hedging strategies using Black–Scholes hedge ratios, even if the volatility is not constant. [Crouhy and Galai, 1995] shows that for path independent options, though the option value depends only on the average volatility, the hedge ratio itself depends on the path of future volatility. It would therefore be financially useful to have an accurate prediction of the volatility.

A variety of techniques exist for predicting volatility. One can use the option prices to compute 'implied' volatilities, such as those derived from the Black-Scholes pricing equations. Another alternative is a multifactor model. Usually combinations of such models work best, but these models all include some constancy in the volatility. [Schwert, 1989] tests for the constancy of the volatility and strongly rejects this hypothesis. Thus one would like to have models that reasonably account for changing volatility. Autoregressive Conditional Heteroscedasticity (ARCH) type models introduced by [Engle, 1982] stipulate the conditional variance as a function of past innovations. Generalizations of this are GARCH [Bollerslev, 1986], and EGARCH [Nelson, 1991], where the conditional variance is a function of past innovations and variances. [Hull and White, 1987] has tried models that have stochastic parts to them.

Variability versus Volatility: Financial market indices seem "too volatile" in that movements seem to not be attributable to any new objective information. It is important to distinguish how "jagged" or "choppy" a time series is from the volatility. One might more accurately refer to the former as the variability. At any given time, if the mean value is known, the tendency of the actual value at that time to wander about this mean value is related to the volatility. In the case where the mean and the variance are constant, the variability will reflect the volatility. We are more concerned with time varying variance.

Correspondence with the Black-Scholes Volatility Models

In modeling the time variation of a stock price, Black-Scholes assume an Ito Process [Ito, 1951] of the form

$$\mathrm{d}S = \tilde{\mu} S \,\mathrm{d}\,t + \tilde{\sigma} S \epsilon \sqrt{\mathrm{d}\,t} \tag{4.1}$$

$\epsilon \sim N(0,1)$, S is the instrument price. $\tilde{\sigma}$ is defined as the volatility and is the standard deviation of the proportional change in the stock price in unit time. It is this $\tilde{\sigma}$ that enters into the calculation of the hedge ratios, option prices, etc. The discrete analog of (4.1) is

$$\frac{\Delta S}{S} \sim N(\tilde{\mu}\Delta t, \tilde{\sigma}^2 \Delta t) \tag{4.2}$$

or $\Delta S \sim N(S\tilde{\mu}\Delta t, S^2\tilde{\sigma}^2\Delta t)$. In our general framework, $f(S \mid I_{t-\Delta t})$ will be the probability density for S at time t. In particular, $I_{t-\Delta t}$ includes the information $S_{t-\Delta t}$. Hence, the Black-Scholes model fits into this framework. The variance σ^2 is related to the volatility $\tilde{\sigma}$ by

$$\tilde{\sigma} = \frac{\sigma}{S_{t-\Delta t}\sqrt{\Delta t}} \tag{4.3}$$

So to calculate option prices, hedge ratios, etc., according to the Black-Scholes prescription, it suffices to know $\sigma(t)$. With this consideration in mind, we now restrict our analysis to the prediction of variance.

Technical Aside

In the Ito interpretation, (4.1) can be used to show that [Gardiner, 1997, pg 95]

$$d\log(S) = (\tilde{\mu} - \frac{1}{2}\tilde{\sigma}^2)dt + \tilde{\sigma}\epsilon\sqrt{dt} \tag{4.4}$$

Thus $\log(S)$ follows a trully Gaussian process, with the variance equal to the volatility. Thus, in the non-discrete case, S is actually log-normal, and it is only $\log(S)$ that is normal.

4.2 Estimating $\sigma(t)$

4.2.1 Basic Setup

We start by introducing some notation that will be used throughout this discussion. We will use μ, σ for the actual mean and standard devition, and $\hat{\mu}$, $\hat{\sigma}$ for predicted values. We will usually denote a data point drawn from the true distribution by d. For a random variable with a Gaussian distribution with mean μ and variance σ^2, we will write $x \sim N(\mu, \sigma^2)$.

We will consider the case of noisy time series. Given the history of information (including the full history of values of the time series), there exists some conditional probability distribution for the next value. We label the time series variable x, and let $f(x_t \mid I_{t-\Delta t})$ be the probability density function for x_t, the value of the series at time t, where $I_{t-\Delta t}$ is the information available at time $t-\Delta t$. Usually, $I_{t-\Delta t}$ is taken as the past few values of the variable x. Ideally, we would like to know what f is, but we will focus on the first two moments of f. We are interested in $\mu(I_{t-\Delta t})$ and $\sigma(I_{t-\Delta t})$, the mean and variance at time t. A model consists of a "black box" that takes as input $I_{t-\Delta t}$ and outputs $\{\hat{\mu}^t, \hat{\sigma}^t\}$, predictions of the mean and variance for time t (figure 4.2). A collection of models consists of a set of such pairs of functions $\{\hat{\mu}_i^t, \hat{\sigma}_i^t\}_{i=1}^{M}$. We will drop the t superscript when the context is clear. The index i refers to which model we are talking about. Here, we will not be concerned with exactly what goes into the black box of figure 4.2. All we know is that we are given a set of models (e.g., GARCH, Neural Networks,...) that take the input $I_{t-\Delta t}$ and provide estimates of the mean and variance as output.

4.2.2 Choosing Between the Models

In order to choose between the models, one requires some 'validation' data. Since our goal is to predict the mean μ and the variance σ^2. We would ideally be given n data points which consist of a series of inputs and the actual values of the mean and variance corresponding to those inputs, $\{I^{\alpha}_{t-\Delta t}, \mu^{\alpha}, \sigma^{\alpha}\}^{n}_{\alpha=1}$. For each model i, one then compares its predictions $\{\hat{\mu}^{\alpha}_{i}, \hat{\sigma}^{\alpha}_{i}\}$ with the actual values and then chooses that model that performed 'best' on the validation data. More formally, one constructs an error measure for model i

$$E_i = \sum_{\alpha=1}^{n} \mathcal{E}(I^{\alpha}_{t-\Delta t}, \hat{\mu}^{\alpha}_{i}, \hat{\sigma}^{\alpha}_{i}, \mu^{\alpha}, \sigma^{\alpha}) \tag{4.5}$$

and chooses the model yielding the lowest value for the error. $\mathcal{E}$ is a measure of how bad a prediction was on a given data point. The most familiar error measure is probably the squared difference error, $E_i = \sum_{\alpha=1}^{n}[(\hat{\mu}^{\alpha}_{i} - \mu^{\alpha})^2 + (\hat{\sigma}^{\alpha}_{i} - \sigma^{\alpha})^2]$. Unfortunately one does not usually have access to the actual values of the mean and variance. All one usually has are data points $(\{d_{\alpha}\}^{n}_{\alpha=1})$ drawn from the distributions $f(x_t \mid I^{\alpha}_{t-\Delta t})$. Based on the data, one has to construct an error measure

$$E_i = \sum_{\alpha=1}^{n} \mathcal{E}(I^{\alpha}_{t-\Delta t}, \hat{\mu}^{\alpha}_{i}, \hat{\sigma}^{\alpha}_{i}, d_{\alpha}) \tag{4.6}$$

that evaluates the estimates $\hat{\mu}$ and $\hat{\sigma}$ without knowing the actual μ and σ. One evaluates this error measure for the different models and picks the model that minimizes the error.

4.2.3 Maximum Likelihood to Choose Between Models

If we know the functional dependence of $f(x_t \mid I_{t-\Delta t})$ on the parameters that we are trying to predict, then after observing the data we can evaluate the likelihood that the data occurred under the assumption that model i is correct. Given the likelihoods of the different models, we will choose that model that has the highest likelihood[1]

$$l(\vec{d} \mid \text{model } i) = \prod_{\alpha=1}^{n} f(d_{\alpha} \mid I^{\alpha}_{t-\Delta t}) \tag{4.7}$$

[1]in the evidence framework, we are picking the model for which the evidence is highest.

where we have assumed that the data have been drawn independently from their respective distributions. The likelihood is the physically important quantity because it relates to a probability. A more mathematically useful quantity is the log (likelihood) because it yields easily to the law of large numbers for many data points, as it converts the product into a sum. In order to evaluate the right hand side of (4.7) for a given model, we need to make some assumption about f. For the analysis that follows we will assume that f is Gaussian. A similar analysis could be done for any other assumption on f. Thus,

$$f(d \mid I_{t-\Delta t}) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[-\frac{(d-\mu)^2}{2\sigma^2}\right] \tag{4.8}$$

where μ and σ depend on $I_{t-\Delta t}$. An equivalent way to formulate this model (say) for the price of a stock is

$$S_t = \mu + \sigma N(0,1) \tag{4.9}$$

To reproduce the Black-Scholes model, one would choose $\mu = S_{t-\Delta t}(1 + \tilde{\mu}\Delta t)$ and $\sigma = \tilde{\sigma} S_{t-\Delta t}\sqrt{\Delta t}$. In particular, we see that Black-Scholes models satisfy our Gaussian assumption on f. The likelihood that the data occurred under a particular model i is given by

$$l_i = l(\vec{d} \mid \text{model } i) = \prod_{\alpha=1}^{n} \frac{1}{\sqrt{2\pi\hat{\sigma}_\alpha^{i\;2}}} exp\left[-\frac{(d_\alpha - \hat{\mu}_\alpha^i)^2}{2\hat{\sigma}_\alpha^{i\;2}}\right] \tag{4.10}$$

Maximizing the likelihood, or equivalently maximizing the log(likelihood) is the criterion that we will use to differentiate between the models. As an example, consider the training of Neural Networks to predict the variance using likelihood as the objective function to optimize [Weigend and Nix, 1995].

4.3 Problems with Maximum Likelihood

4.3.1 Expected Value of the Likelihood and Log(likelihood)

Maximum likelihood is widely used for parameter estimation [Valavanis, 1959]. Here we will analyse maximum likelihood through its expectation. It is well known that for a sample of data drawn from the same distribution, the maximum likelihood predictors are $\hat{\mu} = \bar{d}$, the sample mean and $\hat{\sigma}^2 = \overline{d^2} - \overline{d}^2$, the sample variance. According to the law of large numbers, these estimates approach the true values with probability 1. Suppose we do not have many data points. We can look at the expectation of the likelihood and the log(likelihood) instead.

Consider the likelihood function as a function of the data d_α. Thus it is itself a random variable, for which the distribution is known given the distribution of the data point d_α. One can calculate the expectation of the likelihood and log(likelihood) for model i. For simplicity in notation, we will drop the α index.

$$\log\langle l_i\rangle = -\frac{1}{2}\left[\frac{(\mu-\hat{\mu}_i)^2}{\sigma^2+\hat{\sigma}_i^2} + \log(\sigma^2+\hat{\sigma}_i^2) + \log 2\pi\right] \tag{4.11}$$

$$\langle\log l_i\rangle = -\frac{1}{2}\left[\frac{(\mu-\hat{\mu}_i)^2+\sigma^2}{\hat{\sigma}_i^2} + \log(\hat{\sigma}_i^2) + \log 2\pi\right] \tag{4.12}$$

The first expression tends toward the second in the limit $\sigma^2 \to 0$, as should be expected. Both the likelihood and log(likelihood) are convex independently in the variables $\hat{\mu}_i$ and $\hat{\sigma}_i^2$. However, as functions of two variables, they are not convex. The expectation of the log(likelihood) is maximized when the predicted mean and variance are equal to the true mean and variance[2] .

$$\max_{\hat{\mu}_i,\hat{\sigma}_i^2}[\langle\log(l_i)\rangle] \quad \Rightarrow \quad \hat{\mu}_i \to \mu,\ \hat{\sigma}_i^2 \to \sigma^2 \tag{4.13}$$

If we expect the observed value of the log(likelihood) to be close to its expected value,

[2] $\langle$log(likelihood) $\rangle$ is a cross entropy between f_i and f, so this is a special case of Jensen's inequality.

which will be true with enough data points, then it seems reasonable to maximize the log(likelihood) in order to predict $\{\hat{\mu}, \hat{\sigma}\}$. For this reason we use maximizing the log(likelihood) as our criterion for choosing between models. The expectation of the likelihood itself is not maximized at the correct values. Its maximum is when $\hat{\sigma} \to 0$.

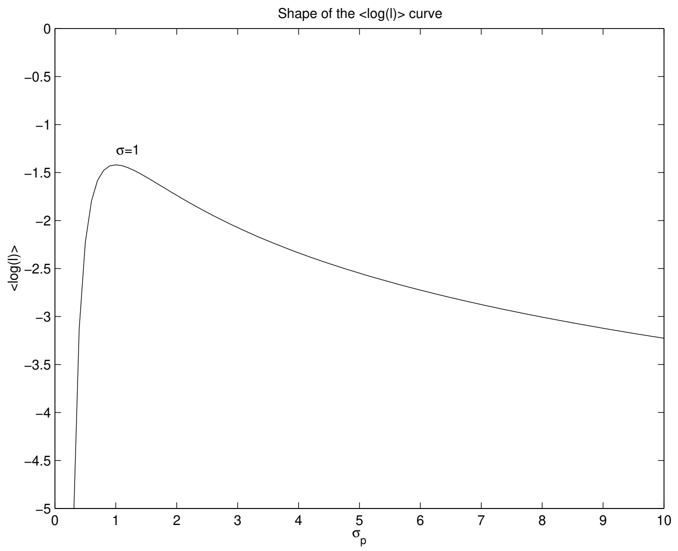
$$\max_{\hat{\mu}_i, \hat{\sigma}_i^2} [\log\langle l_i \rangle] \qquad \Rightarrow \qquad \hat{\mu}_i \to \mu,\ \hat{\sigma}_i^2 \to 0 \tag{4.14}$$

This rules out likelihood itself as a comparator because its expected behavior is not desirable.

We now look at the expected log(likelihood) more closely. We will say that model 1 strictly dominates model 2 if $|\hat{\mu}_1 - \mu| > |\hat{\mu}_2 - \mu|$ and $|\hat{\sigma}_1 - \sigma| > |\hat{\sigma}_2 - \sigma|$ and one of the inequalities is strict[3]. One expects that if model 1 strictly dominates model 2, then its expected log(likelihood) should be higher. Unfortunately one can find common situations where this is not the case. This becomes evident from figure 4.3. As $\hat{\sigma}_i^2 \to 0$, $\langle \log l_i \rangle \to -\infty$ so we can make model 2's variance large while sending model 1's variance to zero. Then, model 1 will strictly dominate model 2 despite having a lower log(likelihood) . Thus, we see that even log(likelihood) will lead to an expected worse choice in such situations. However we note that training (say neural networks) by maximizing log(likelihood) [Weigend and Nix, 1995] using small perturbations will select 'better' models on the average because the maximum is unique. Note that by initially choosing a model with lower log(likelihood), training could be faster owing to the asymmetry of the curve about the actual variance. If the neural network cannot implement the maximum of the curve then training may stop at a worse value than is necessary, owing to this asymmetry.

The previous analysis might suggest that a lower variance model with higher likelihood is the best of both worlds. Always pick the lower variance model in this case. What about the possibility of a "worse" model predicting lower variance and having a higher likelihood. This will be the subject of the next section. In particular

[3]we use the absolute error criterion here, but almost any will do just as well.



The curve is not symmetric about the true variance. Two models with equal likelihoods can have different errors.

Figure 4.3: Plot of $\langle \log l_i \rangle$ as a function of $\hat{\sigma}_i^2$, keeping $\hat{\mu}$ fixed

we will compare the ground truth model to a 'worse' model predicting a lower variance.

4.3.2 Probability of Choosing the Wrong Model

Suppose that you have two models, model 1 dominating model 2 as described above. One expects that model 1 will be picked more often than model 2. This does not follow from the previous analysis of the expected value because it is possible for model 1's expected log(likelihood) to be greater without its likelihood being higher most of the time.

- Is is possible for a model to strictly dominate another model, yet on the basis of likelihood be chosen less that half the time?

We will not only answer this question with an affirmative, but with an affirmative even when the better model is the *ground truth model.* Set up the problem in the following way. We have n data points $\{I_{t-\Delta t}, d_\alpha\}$ where each of the d_α have been drawn from a (possibly *different*) distribution $\sim N(\mu_\alpha, \sigma_\alpha^2)$. We will consider the case where $\hat{\mu}^i_\alpha = \mu_\alpha$ (perfect prediction of the mean), while the variances predicted by the models are given by

$$\hat{\sigma}^i_\alpha = \lambda^i_\alpha \sigma_\alpha, \quad \lambda^i_\alpha \geq 0 \tag{4.15}$$

thus the models are parametrized by λ^i_α,$\alpha = 1, \ldots, n$. The likelihood of model i with parameters $\vec{\lambda}^i$ is

$$l_i = \prod_\alpha N_{d_\alpha}(\mu_\alpha, (\lambda^i_\alpha \sigma_\alpha)^2) \tag{4.16}$$

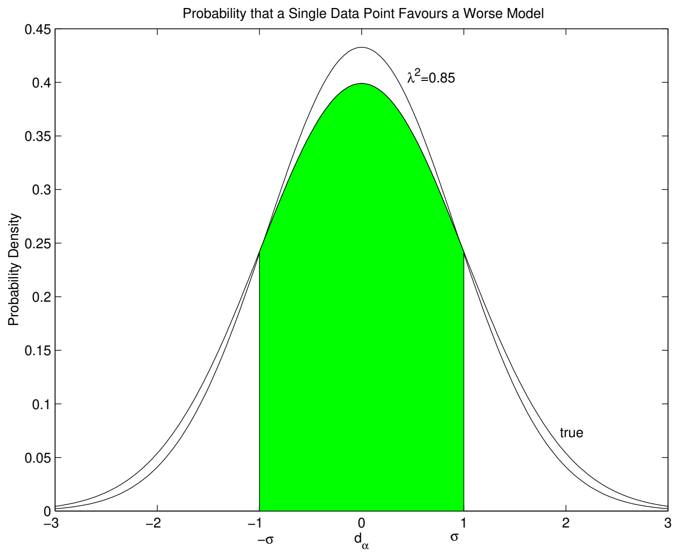
We are going to compare the two models $\{\lambda_\alpha = 1\}$, the perfect model, and $\{\lambda_\alpha\}$. We seek the probability $(P_{\vec{\lambda}})$ that $l_{\vec{\lambda}} \geq l_{\vec{1}}$ where $l_{\vec{1}}$ is the likelihood of the ground truth model and $l_{\vec{\lambda}}$ is the likelihood of model $\vec{\lambda}$.

Treat the data outcome $\vec{d}$ as a random variable $\vec{x}$. Then this likelihood itself is a random variable, depending on the parameter $\vec{\lambda}$, so we can ask the question: what is the probability that this random variable with parameter $\vec{\lambda}$ is greater than the random variable with parameter $\vec{\lambda} = \vec{1}$, i.e., we are asking for the frequency with which the 'worse' model is chosen over the *actual* model.

To get the ball rolling consider the simplest case – one data point. Assume that both models have mean 0, the true model has variance 1 and the worse model has variance λ^2, slightly less than 1. The situation is depicted in figure 4.4. The probability we require is the area under the true curve for the region where the worse model is above the true model. This area is approximately 0.68. Therefore, close to 70% of the time, this worse model will be chosen *over the true model.* Generalizing to n data points, let $P_{\vec{\lambda}}$ be the probability that the worse model is picked over the true model. It is given by

$$P_{\vec{\lambda}} = \mathrm{Prob}[l_{\vec{\lambda}} \geq l_{\vec{1}}] = \int_{\{l_{\vec{\lambda}} \geq l_{\vec{1}}\}} \mathrm{d}^n\mathrm{x}\, l(\vec{x} \mid \vec{\lambda} = \vec{1}) \tag{4.17}$$

In appendix 4.6, it is shown that $P_{\vec{\lambda}}$ is symmetric in α (remember that α indexes the data points), so the condition for $P_{\vec{\lambda}}$ to be maximum will be symmetric in λ_α. With this in mind, we therefore consider the case $\vec{\lambda} = \lambda\vec{1}$. $P_{\vec{\lambda}}$ is calculated in appendix 4.6 along with various asymptotic properties. Figure 4.5 summarizes the result in a



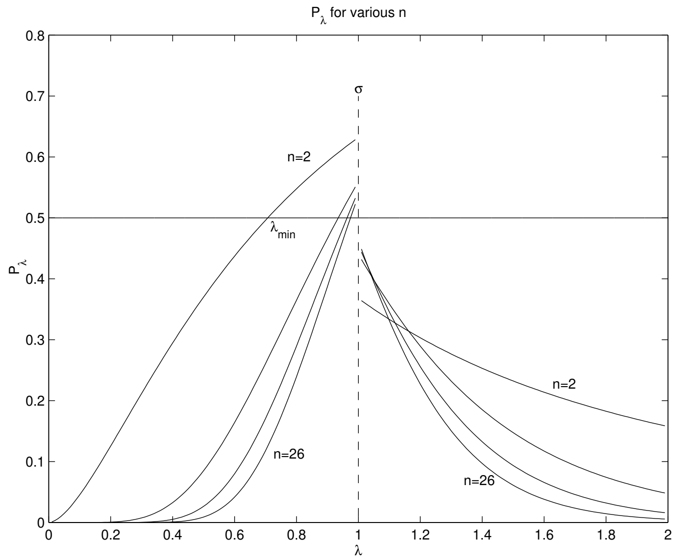
The probability that a worse model predicting a slightly smaller variance is picked over the true model is the probability that the data point lands in the one sigma interval. Ths probability is therefore given by the shaded area on the graph.

Figure 4.4: Probability that a single data point favors a worse model.

plot of P_λ versus λ. For λ slightly less than 1, P_λ is greater than $\frac{1}{2}$. This means that a worse model will be chosen over the actual model more than $\frac{1}{2}$ the time. *This holds for any number of data points.* The smallest value that λ can be with $P_\lambda \geq \frac{1}{2}$ ($\lambda_{min}(n)$) is tabulated for various n in table 4.1. Referring to table 4.1, we note that for n=100, a model with a 0.7% error in the prediction of σ will be chosen more often than the actual model. This is a fairly significant error when dealing with a tradable quantity in financial markets.

4.4 Correcting the Maximum Likelihood Errors

Up to now we have diagnosed some problems with the maximum likelihood scheme. We now try to tackle the problem of compensating for this error using our understanding of how it fails. The models that are more probable than the actual model are models which underestimate the variance (assuming the mean is predicted perfectly).



For n even we get the result $P_\lambda = 1 - e^{-\beta n} \sum_{k=0}^{\frac{n}{2}-1} \frac{(\beta n)^k}{k!}$, $\beta = \frac{\lambda^2 \log \lambda}{\lambda^2 - 1}$. *The result for n odd can be obtained numerically. Regions where the curve exceeds* $\frac{1}{2}$ *are regions where the worse model is picked more often than the actual one.*

Figure 4.5: Plot of P_λ as a function of λ.

n	$\lambda_{min}(n)$	$P_{max}(n)$	$\langle\frac{1}{\lambda}\rangle = \alpha_{correc}$
1	0.4937	0.6831	∞
2	0.7072	0.6321	1.7725
5	0.8729	0.5842	1.1894
10	0.9349	0.5594	1.0837
20	0.9670	0.5422	1.0397
50	0.9866	0.5268	1.0153
100	0.9934	0.5186	1.0076
500	0.9986	0.5088	1.0015
1000	0.9993	0.5059	1.0008

For $\lambda_{min}(n) \le \lambda \le 1$, $P_\lambda \ge \frac{1}{2}$. $\langle\frac{1}{\lambda}\rangle$ *is the expected correction factor* α_{correc} *for the new prediction given the maximum likelihood model prediction* ($\hat{\sigma} \rightarrow \alpha_{correc}\hat{\sigma}$). $P_{max}(n)$ *is the maximum value that* P_λ *can take and occurs for* $\lambda \rightarrow 1^-$

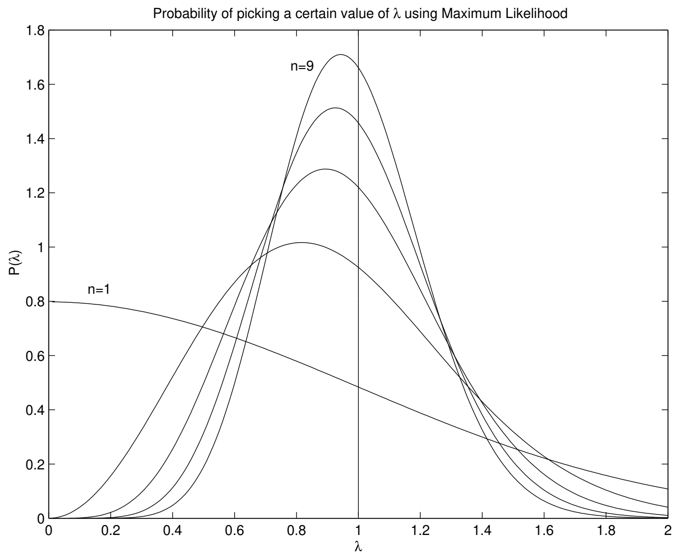
Table 4.1: Probability of picking a worse model.

One expects that the model chosen using maximum likelihood will systematically have a λ that is less than 1. Suppose that we have a model that is predicting $\hat{\sigma} = \lambda\sigma$ on the average. We would like to correct the prediction arrived at using maximum likelihood methods by multiplying our prediction by some correction factor to get a better prediction.

$$\hat{\sigma} \rightarrow \alpha_{correc}\hat{\sigma} \tag{4.18}$$

In order to calculate α_{correc}, we need the probability distribution for obtaining a particular model with parameter λ.

4.4.1 Probability Distribution Over λ



As $n \rightarrow \infty$ $P(\lambda)$ tends to a Gaussian centered at 1. When n=1, $P(\lambda)$ is finite at the origin.

Figure 4.6: Plot of $P(\lambda)$ for various values of n.

Consider the following idealized training dynamics. We have the class of models that are parametrised by λ. Training proceeds in the following way. Start with a random λ and perturb λ a little toward the higher range and toward the lower range. We are now only dealing with the three λ's $\left\{\lambda - \frac{d\lambda}{2}, \lambda, \lambda + \frac{d\lambda}{2}\right\}$. Using the data we compare the likelihood of these three models and choose the model that produces the highest likelihood. We continue the process until no change in λ results. One can now decrease the step size to attain the necessary accuracy. In this way, training of the model proceeds in such a way that the model ends up picking the λ that maximized the likelihood function on the data set. In actuality the learning proceeds not by varying λ but by varying the parameters of the model (in the case of neural networks, these are the weights). If the learning model is sufficiently general, then because we expect the likelihood to be maximized at all λ_α equal, we restrict ourselves to that space for the computation of $P(\lambda)$. The detailed derivation of this probability density is in appendix 4.7. The final result we get for this probability density is

$$P(\lambda)d\lambda = \frac{n^{\frac{n}{2}} e^{-n\frac{\lambda^2}{2}}}{\Gamma(\frac{n}{2})} \left(\frac{\lambda^2}{2}\right)^{\frac{n}{2}-1} \lambda d\lambda \tag{4.19}$$

We show a plot of $P(\lambda)$ for various n in figure 4.6.

Correction Factors

Given the data set, we settle on a model $\hat{\sigma} = \lambda\sigma$ that maximizes the likelihood. The probability distribution for λ is given by (4.19). Knowing $P(\lambda)$, can we change the prediction in some deterministic way in order to improve our expected performance. More quantitatively, one has in mind some penalty function $\mathcal{E}(\sigma_p, \sigma)$, where σ_p is the final prediction. We assume $\sigma_p = \alpha\hat{\sigma}$. Noting that by definition, $\sigma = \hat{\sigma}/\lambda$, we would like to solve the following problem

$$\alpha_{correc} = \underset{\alpha}{\operatorname{argmin}} \left\langle \mathcal{E}(\alpha\hat{\sigma}, \frac{\hat{\sigma}}{\lambda}) \right\rangle_\lambda \tag{4.20}$$

Table 4.2 shows the correction factors for some penalty terms. We will focus on $\alpha_{correc} = \frac{1}{\lambda}$, which has an interpretation as an expected correction factor to bring $\hat{\sigma}$ up to σ. One might also be interested in knowing $1/\langle\lambda\rangle$, the correction factor to make σ_p unbiased. Note that $\left\langle\frac{1}{\lambda}\right\rangle \geq \frac{1}{\langle\lambda\rangle}$. These quantities are easily calculated from

$\mathcal{E}(\sigma_p, \sigma)$	α_{correc}
$(\sigma_p - \sigma)^2$	$\langle \frac{1}{\lambda} \rangle$
$\left(\frac{\sigma}{\sigma_p} - 1\right)^2$	$\langle \frac{1}{\lambda^2} \rangle / \langle \frac{1}{\lambda} \rangle$
$\left(\log \frac{\sigma}{\sigma_p}\right)^2$	$exp[\langle \log(\lambda) \rangle]$

Table 4.2: α_{correc} for different penalty functions.

P(λ) using the identity (4.49) in appendic 4.7

$$\alpha_{correc} = \left\langle \frac{1}{\lambda} \right\rangle = \sqrt{\left(\frac{n}{2}\right)} \frac{\Gamma(\frac{n-1}{2})}{\Gamma(\frac{n}{2})} \tag{4.21}$$

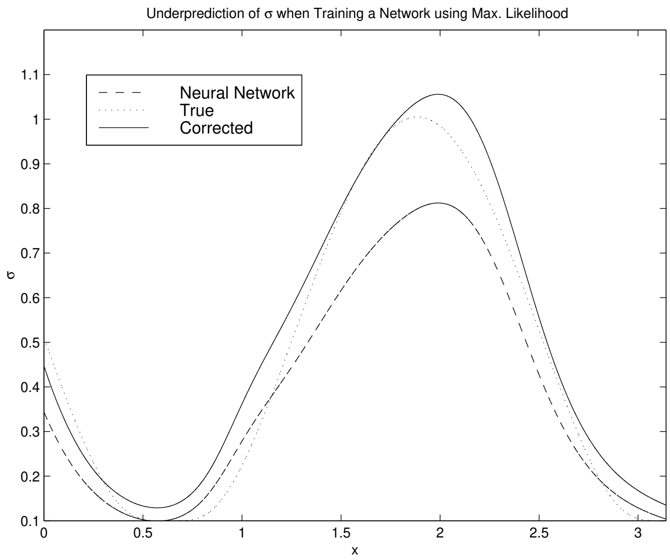
$$\frac{1}{\langle \lambda \rangle} = \sqrt{\left(\frac{n}{2}\right)} \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n+1}{2})} \tag{4.22}$$

Values of the correction factor for various values of n are given in table 4.1. Note that for $n = 100$ the correction is about 0.8% which is not trivial.

We summarize the correction method. Assuming that the models are predicting the mean well, train the models using maximum likelihood and arrive at a model to be used for future predictions. Now, given a new test input, obtain this model's prediction and correct by the correction factor described above, depending on the penalty function that you wish to use.

4.4.2 Example: Training Neural Networks Using maximum Likelihood

In this example, the model is a neural network with 131 weights. The weights are altered by maximizing the likelihood [Weigend and Nix, 1995]. The input variable which we have called $I_{t-\Delta t}$ is $x \in [0, \pi]$. The mean $\mu(x)$ and variance $\sigma^2(x)$ are



We show the neural network prediction, $\hat{\sigma}(t)$, the true $\sigma(t)$, and the corrected version using the correction factor given for 200 data points in table 4.3

Figure 4.7: Plot of the neural network learned σ.

functions of x

$$\mu(x) = \sin(1.5x)\sin(2.5x), \qquad \sigma^2(x) = 0.01 + 0.25 \times [1 - \sin(2.5x)]^2 \tag{4.23}$$

and the model is a mapping $x \to \begin{bmatrix} \hat{\mu}(x) \\ \hat{\sigma}^2(x) \end{bmatrix}$. The network was trained on n examples by altering the weights in the direction that increased the likelihood that the data occurred under the model. The final network arrived at is used as the final model. It is to this network that we wish to apply the correction factor. A question arises as to what the effective number of data points (n_{eff}) is. Each parameter can be regarded as being trained on some of the data points. So n_{eff} should be approximately proportional to the number of examples, $n_{eff} \sim \gamma n$. Using this relationship we can get a theoretical prediction to compare with the experimental value. The results are summarized in table 4.3 [4].

[4] We thank Zehra Cataltepe of the Learning Systems Group at Caltech for use of the results from her experiments in verifying the method of [Weigend and Nix, 1995]. $\langle \alpha_{correc} \rangle$ was computed from the results only where σ was larger than a threshold because where σ is small, the behaviour

n	$\langle \frac{1}{\lambda} \rangle$	
	expt.	theory
100	1.25 ± 0.18	2.03
200	1.24 ± 0.03	1.30
300	1.22 ± 0.04	1.19
600	1.07 ± 0.09	1.08
1000	1.04 ± 0.03	1.05

The theoretical predictions were obtained by assuming a proportional relationship $n_{eff} = \gamma n$, fitting γ to the data using the expected theoretical form. The theory fit is for $\gamma = 0.012$. It is interesting to compare γ to 1/# weights (= 0.008), then n_{eff} would have an interpretation as the # of examples per weight.

Table 4.3: Theoretical fit to experiments using Neural Networks

The agreement between the theoretical values and the experimental values seems convincing especially as the number of data points increases. Note that the variance in the experimental values is relatively small implying that the network has indeed settled on a model with almost a constant parameter for λ. A sample curve is shown for $n = 200$ in figure 4.7. How one would calculate n_{eff} for a general class of models with a given learning algorithm is not yet obvious. In our discussion we have assumed that the class of models is good enough to implement the various models with parameter λ. Exactly how we search through this space and how the models are parametrized are expected to affect n_{eff}.

4.5 Comments

It seems appropriate to summarize the path that we have followed in this chapter. We started out by setting up a framework for comparing between models. In this framework, we used maximum likelihood to compare between models and we found that this can lead to choosing the wrong model. These results are summarized by the

is erratic. The mean as expected was learned well, so we can apply the analysis above where we assumed the mean was being predicted exactly.

picture in figure 4.8.

	IN SAMPLE	OUT OF SAMPLE
FINITE DATA	Choosing the model that maximizes the likelihood will yield a model that systematically predicts lower variance, even if the mean is predicted well.	Probability of choosing the wrong model is $> \frac{1}{2}$ for some 'worse' models.
EXPECTED VALUE	NOT APPLICABLE	Possible to find models 1,2 with model 1 worse than 2 but $\langle \log l_1 \rangle > \langle \log l_2 \rangle$.

Figure 4.8: Diagram depicting the problems with the maximum likelihood method.

We find a systematic underestimation of the variance in time series analogous to that of a sample variance. We expect the mean to be predicted well. When the mean is predicted well, we derive a probability distribution for the underprediction factor. Given a penalty function, we can derive a correction factor, α_{correc}. This correction factor can be large. In this way we are able to choose a model from a class of models using maximum likelihood, and then improve that model using the correction factor. What about when the mean is not being predicted well? One would like to have a way to predict the variance without having to predict the mean. No noisy free lunch (chapter 8) partially answers this question. It is necessary to have some prior on $\mu(t)$ to make any prediction, and in this case we have a "good" prior.

4.6 Appendix A

In this appendix we derive the technical results used in section 4.3. The a priori model that will be assumed is Gaussian. Suppose we have a data point (d_α) drawn from the actual distribution. Given a model $\{\hat{\mu}_\alpha, \hat{\sigma}_\alpha\}$, the likelihood of the data point is

$$l(d_\alpha \mid \hat{\mu}_\alpha, \hat{\sigma}_\alpha) \equiv N_{d_\alpha}(\hat{\mu}_\alpha, \hat{\sigma}^2_\alpha) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2_\alpha}} exp\left[-\frac{(d_\alpha - \hat{\mu}_\alpha)^2}{2\hat{\sigma}^2_\alpha}\right] \tag{4.24}$$

n data points d_α are drawn independently from distributions with (possibly varying) actual means (μ_α) and (possibly varying) actual variances (σ^2_α). Assume that the model is predicting the mean correctly ($\hat{\mu}_\alpha = \mu_\alpha$), while the variances predicted by the model are given by

$$\hat{\sigma}_\alpha = \lambda_\alpha \sigma_\alpha, \quad \lambda_\alpha \geq 0 \tag{4.25}$$

The likelihood for the model with parameters $\vec{\lambda}$ is

$$l(\vec{d} \mid \vec{\lambda}) = \prod_\alpha N_{d_\alpha}(\mu_\alpha, (\lambda_\alpha \sigma_\alpha)^2) \tag{4.26}$$

Treat the data outcome $\vec{d}$ as a random variable $\vec{x}$. Then this likelihood is a random variable, depending on the parameter $\vec{\lambda}$. We seek the probability that this random variable with parameter $\vec{\lambda}$ is greater than the random variable with parameter $\vec{\lambda} = \vec{1}$, i.e., we are asking for the frequency with which the 'worse' model is chosen over the *actual* model. This probability is given by

$$P_{\vec{\lambda}} = \mathrm{Prob}[l_{\vec{\lambda}} \geq l_{\vec{1}}] = \int_{\{l_{\vec{\lambda}} \geq l_{\vec{1}}\}} \mathrm{d}^n\mathrm{x}\, l(\vec{x} \mid \vec{\lambda} = \vec{1}) \tag{4.27}$$

The boundary condition for the integral is non trivial. The condition $l_{\vec{\lambda}} \geq l_{\vec{1}}$ implies $\log(l_{\vec{\lambda}}) \geq \log(l_{\vec{1}})$ due to the monotonicity of the *log* function. Thus, the boundary condition reduces to

$$\sum_{\alpha} \frac{(x_\alpha - \mu_\alpha)^2}{2\sigma_\alpha^2}[1 - \frac{1}{\lambda_\alpha^2}] \geq \sum_{\alpha} \log(\lambda_\alpha) \tag{4.28}$$

Making a change of variables to $u_\alpha = \frac{(x_\alpha - \mu_\alpha)}{\sigma_\alpha}$, the expression for $P_{\vec{\lambda}}$ becomes

$$\int\limits_{\frac{1}{2}\sum\limits_{\alpha} u_\alpha^2 \left[1 - \frac{1}{\lambda_\alpha^2}\right] \geq \sum\limits_{\alpha} \log(\lambda_\alpha)} d^n u_\alpha \; e^{-\frac{u_\alpha^2}{2}} \tag{4.29}$$

The boundary condition is symmetric in α, the constraint on the λ_α's ($\lambda_\alpha \geq 0$) is symmetric in α and the integrand is also symmetric in α so the condition for $P_{\vec{\lambda}}$ to be maximized will be symmetric in α. With this as motivation, we consider the class of models for which $\vec{\lambda} = \lambda\vec{1}$, i.e., all λ_α's constant. The boundary condition now reduces to

$$\begin{cases} \sum\limits_{\alpha} \frac{(x_\alpha - \mu_\alpha)^2}{2\sigma_\alpha^2} \leq \beta n, & 0 < \lambda < 1 \\ \sum\limits_{\alpha} \frac{(x_\alpha - \mu_\alpha)^2}{2\sigma_\alpha^2} \geq \beta n, & 1 < \lambda \end{cases} \qquad \beta = \frac{\lambda^2 \log \lambda}{\lambda^2 - 1}, \quad \beta > 0 \tag{4.30}$$

$$0 < \lambda < 1 \Rightarrow 0 < \beta < \frac{1}{2}, \qquad \lambda \to 0, 1 \Rightarrow \beta \to 0, \frac{1}{2} \tag{4.31}$$

The integral can be reduced to one dimensional form by transforming to spherical coordinates with the aid of the following result derivable by standard methods [Huang, 1987, pg 138-139]

$$\int_{r<R} \mathrm{d}^n \mathrm{x}\, f(r) = \frac{2\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2})} \int_0^R \mathrm{dr}\; r^{n-1} f(r) \tag{4.32}$$

Thus with some manipulation, one finally gets

$$P_{\vec{\lambda}} = P_\lambda = \begin{cases} \frac{1}{\Gamma(\frac{n}{2})} \int_0^{\beta n} \mathrm{dy} \;\; y^{\frac{n}{2}-1} e^{-y} & 0 < \beta < \frac{1}{2} \\ \\ \frac{1}{\Gamma(\frac{n}{2})} \int_{\beta n}^{\infty} \mathrm{dy} \;\; y^{\frac{n}{2}-1} e^{-y} & \frac{1}{2} < \beta \end{cases} \tag{4.33}$$

When n is even we can get a closed form answer. When n is odd, and large, we can get an answer in terms of the asymptotic form for the error function. We pursue the case n even, leaving the rest to numerical analysis. We also restrict ourselves to the interesting domain $0 < \lambda < 1$. The case $1 < \lambda$ follows an identical analysis. We rewrite (4.33) as

$$\frac{(-1)^{m-1}}{\Gamma(m)} \left[\left(\frac{\partial}{\partial q} \right)^{m-1} \int_0^{2\beta m} dy \ e^{-qy} \right]_{q=1} \tag{4.34}$$

where $n = 2m$. Doing the integral followed by the derivatives and noting that $\Gamma(m) = (m-1)!$ for positive integer m yields after some algebra

$$P_\lambda = \begin{cases} 1 - e^{-\beta n} \sum_{k=0}^{\frac{n}{2}-1} \frac{(\beta n)^k}{k!} & 0 < 1 < \lambda \\ e^{-\beta n} \sum_{k=0}^{\frac{n}{2}-1} \frac{(\beta n)^k}{k!} & 1 < \lambda \end{cases} \tag{4.35}$$

We use the following lemma to discover an asymptotic form for this as $n \to \infty$

Lemma 4.6.1

$$\lim_{n\to\infty} \left[e^{-n} \sum_{k=0}^{[f(n)]} \frac{n^k}{k!} \right] = \int_{-\infty}^{\alpha} du \, \frac{e^{-\frac{u^2}{2}}}{\sqrt{2\pi}} \qquad \alpha = \lim_{n\to\infty} \left[\sqrt{n} \left(\frac{f(n)}{n} - 1 \right) \right] \tag{4.36}$$

Proof: Note that a Poisson distribution with parameter n has $P(k) = e^{-n} n^k / k!$. As $n \to \infty$, this tends to a Gaussian. More precisely [Feller, 1968a, pg 194, Problem 9]

$$\sum_{n+\beta\sqrt{n}<k<n+\alpha\sqrt{n}} P(k) \to \mathcal{N}(\alpha) - \mathcal{N}(\beta) = \int_\beta^\alpha du \, \frac{e^{-\frac{u^2}{2}}}{\sqrt{2\pi}} \tag{4.37}$$

In our case, $\beta = -n$ and $\alpha = \frac{f(n)-n}{\sqrt{n}}$. The lemma now follows. ■

The asymptotic form of P_λ has $\alpha = \lim_{n\to\infty} \left[\sqrt{\frac{n}{\beta}} \left(\left(\frac{1}{2} - \beta\right) - \frac{1}{n} \right) \right]$. Thus for $\alpha < 0$ ($\Rightarrow \mathrm{P}_\lambda > \frac{1}{2}$), $\beta(n)$ must approach $\frac{1}{2}$ faster than $\frac{1}{n}$. We can make this more precise. Suppose that $\lambda = 1 - g(n)$ and $\beta = \frac{1}{2} - \eta(n)$. Then expanding to second order using the definition of $\beta(\lambda)$ one finds

$$g(n) = 2\eta(n) - \frac{2}{3}\eta^2(n) + \vartheta(\eta^3) \qquad \eta(n) = \frac{g(n)}{2} + \frac{g^2(n)}{12} + \vartheta(g^3) \tag{4.38}$$

Expanding the expression for P_λ in $\eta(n)$ we can get an asymptotic form as $\beta \to \frac{1}{2}$. We can then answer the question: what are the values of β that give $\mathrm{P}_\lambda > \frac{1}{2}$. Tedious but elementary algebra yields

$$P_\lambda \overset{\eta\to 0}{\sim} 1 - e^{-\frac{n}{2}} \sum_{k=0}^{\frac{n}{2}-1} \frac{1}{k!}(\frac{n}{2})^k - \epsilon(n)[\eta(n) + \eta^2(n) + \vartheta(n\eta^3)] \qquad \epsilon(n) = \frac{ne^{-\frac{n}{2}}(\frac{n}{2})^{\frac{n}{2}-1}}{(\frac{n}{2}-1)!} \tag{4.39}$$

Using Stirling's approximation for the factorial function, we can now get an expression for that $\lambda_{min}(n)$ for which $\mathrm{P}_\lambda = \frac{1}{2}$. Thus $\mathrm{P}_\lambda > \frac{1}{2}$ for $\lambda_{min}(n) < \lambda < 1$

$$\lambda_{min}(n) \overset{n\to\infty}{\sim} 1 - \frac{1}{n}\left[0.6 - \frac{0.3}{n}\right] \tag{4.40}$$

As $\lambda \to 1^-$, $\mathrm{P}_\lambda(n)$ tends toward its maximum value ($\mathrm{P}_{max}(n)$) for given n

$$P_\lambda \xrightarrow{\lambda\to 1^-} P_{max}(n) = \frac{1}{2} + \frac{1}{\sqrt{12\pi n}}\left[1 - \frac{1}{36n}\right] \tag{4.41}$$

Note the fact that for n fairly large, $\lambda_{min}(n)$ is significantly less than 1. All models that lie in the region between $\lambda_{min}(n)$ and 1 are more likely to be chosen than the actual model, so the question arises as to whether one can compensate for this systematic underestimation of the variance when the maximum likelihood scheme is used. This is the subject of Appendix 4.7.

4.7 Appendix B

In this appendix we derive the probability distribution used in the development of the λ correction factor. A similar analysis to that which led up to (4.30) yields the following conditions

$$l_{\lambda_1} \leq l_\lambda \Rightarrow \begin{cases} \sum_i \frac{(x_i-\mu_i)^2}{2\sigma_i^2} \leq n\beta(\lambda,\lambda_1), & \lambda < \lambda_1 \\ \sum_i \frac{(x_i-\mu_i)^2}{2\sigma_i^2} \geq n\beta(\lambda,\lambda_1), & \lambda > \lambda_1 \end{cases} \qquad \beta(\lambda,\lambda_1) = \frac{\lambda^2 \log\left(\frac{\lambda}{\lambda_1}\right)}{\left[\frac{\lambda^2}{\lambda_1^2} - 1\right]}, \quad \beta > 0 \tag{4.42}$$

What is needed is that $l_{\lambda+\frac{d\lambda}{2}} \leq l_\lambda$ and that $l_{\lambda-\frac{d\lambda}{2}} \leq l_\lambda$. From (4.42) this is equivalent to the condition that the data satisfy

$$n\beta(\lambda,\lambda-\tfrac{d\lambda}{2}) \leq \sum_\alpha \frac{(x_\alpha-\mu_\alpha)^2}{2\sigma_\alpha^2} \leq n\beta(\lambda,\lambda+\tfrac{d\lambda}{2}) \tag{4.43}$$

So to get the probability that this is true we simply need to integrate the probability density for the x_i's over the region where this condition is true. Thus we want

$$\int_{n\beta(\lambda,\lambda-\frac{d\lambda}{2}) \leq \sum_\alpha \frac{(x_\alpha-\mu_\alpha)^2}{2\sigma_\alpha^2} \leq n\beta(\lambda,\lambda+\frac{d\lambda}{2})} \mathrm{d}^n\mathrm{x} \; \prod_\alpha N_{x_\alpha}(\mu_\alpha,\sigma_\alpha^2) \tag{4.44}$$

This is the probability that the data fall within the range required to ensure that $l_{\lambda+\frac{d\lambda}{2}} \leq l_\lambda$ and that $l_{\lambda-\frac{d\lambda}{2}} \leq l_\lambda$. But this precisely the probability that our algorithm stops at λ. Using (4.32) we can reduce this integral into one dimensional form as was done to get (4.33). Thus the probability that $\lambda' \in [\lambda, \lambda+\frac{d\lambda}{2}]$ is given by

$$P(\lambda)\frac{d\lambda}{2} = \frac{1}{\Gamma(\frac{n}{2})}\int_{n\beta(\lambda,\lambda-\frac{d\lambda}{2})}^{n\beta(\lambda,\lambda+\frac{d\lambda}{2})} \mathrm{dy}\; y^{\frac{n}{2}-1}e^{-y} \tag{4.45}$$

The idea now is to expand the limits in $\mathrm{d}\lambda$. To this end we require the expansion $\beta(\lambda,\lambda+\delta) = \frac{\lambda^2}{2}\left[1+\frac{\delta}{\lambda}\right] + \vartheta(\delta^2)$. We use this to expand the limits of integration and then using the substitution $nx = y - n\frac{\lambda^2}{2}$ the integral reduces to

$$P(\lambda)d\lambda = \frac{2n^{\frac{n}{2}}e^{-n\frac{\lambda^2}{2}}}{\Gamma(\frac{n}{2})}\left(\frac{\lambda^2}{2}\right)^{\frac{n}{2}-1}\int_{-\frac{\lambda}{4}d\lambda}^{\frac{\lambda}{4}d\lambda} dx \left[1+\frac{2x}{\lambda^2}\right]^{\frac{n}{2}-1} e^{-nx} \tag{4.46}$$

Expanding the integrand as a Taylor series in x and doing the integral gives a result as a Taylor series in $d\lambda$. Taking the limit as $d\lambda \to 0$ one finally gets

$$P(\lambda)d\lambda = \frac{n^{\frac{n}{2}}e^{-n\frac{\lambda^2}{2}}}{\Gamma(\frac{n}{2})}\left(\frac{\lambda^2}{2}\right)^{\frac{n}{2}-1} \lambda d\lambda \tag{4.47}$$

Thus we have found, using this method of model selection, the probability that a model with parameter λ is obtained. We are interested in quantities like the expected correction factor

$$\alpha_{correc} = \frac{1}{\lambda} \tag{4.48}$$

These quantities are easily calculated from P(λ) using the identity

$$\langle\lambda^m\rangle = \int_0^\infty d\lambda\, \lambda^m \left\{\frac{n^{\frac{n}{2}}e^{-n\frac{\lambda^2}{2}}}{\Gamma(\frac{n}{2})}\left(\frac{\lambda^2}{2}\right)^{\frac{n}{2}-1}\lambda\right\} = \left(\frac{2}{n}\right)^{\frac{m}{2}}\frac{\Gamma\left(\frac{n+m}{2}\right)}{\Gamma\left(\frac{n}{2}\right)} \tag{4.49}$$

One finds by substituting the value $m = \pm 1$ that

$$\alpha_{correc} = \left\langle\frac{1}{\lambda}\right\rangle = \sqrt{\left(\frac{n}{2}\right)}\frac{\Gamma(\frac{n-1}{2})}{\Gamma(\frac{n}{2})} \tag{4.50}$$

$$\frac{1}{\langle\lambda\rangle} = \sqrt{\left(\frac{n}{2}\right)}\frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n+1}{2})} \tag{4.51}$$

Part II

Density Estimation

Chapter 5

Introduction and Background[1]

— *When solving a problem, try to avoid solving a more general problem as an intermediate step.*

– *Vladimir Vapnik*

A majority of problems in science and engineering have to be modeled in a probabilistic manner. Even if the underlying phenomena are inherently deterministic, the complexity of these phenomena often makes a probabilistic formulation the only feasible approach from the computational point of view. Consequently, tools from probability theory have become very valuable in the characterization, analysis, and solution of many such problems. Although quantities such as the mean, the variance, and possibly higher order moments of a random variable have often been sufficient to characterize a particular problem, the quest for higher modeling accuracy, and for more realistic assumptions drives us towards modeling the available random variables using their probability density. This of course leads us to the problem of density estimation (see [Silverman, 1993]). Density estimation has been a very active research topic in the past three decades. There has always been competition among researchers to invent more accurate density estimation techniques, since this will significantly impact the applications that rely on density estimation as an essential component. Examples of applications that need a density estimation step include the following: the application of the optimal pattern classification procedure, the Bayes procedure (see [Bishop, 1995a], and [Duda and Hart, 1973]), the determination of the optimal detection threshold for the signal detection problem [van Trees, 1968], time series prediction applications where the density for a future data point is required rather

[1] I would like to acknowledge that this work on density estimation was done in collaboration with Dr. Amir Atiya while he was a Visiting Associate at the Caltech Learning Systems Group.

than a single forecast for the value (see [Weigend and Srivastava, 1995]), clustering for unsupervised classifier design [Duda and Hart, 1973], the design of optimal scalar quantizers for the signal encoding problem [Gersho and Gray, 1992], and finding estimates of confidence intervals and quantile levels for the parameter estimation and regression problems [Larsen and Marx, 1986].

5.1 Density Estimation is Ill-Posed

Non parametric statistics developed with the need for solutions to the density estimation and regression problem that were both unbiased and consistent. The density, if it exists, is the solution to the integral equation[2]

$$\int_{-\infty}^{x} dt\; p(t) = F(x) \tag{5.1}$$

where $F(x)$ is the probability distribution function. The density estimation problem can be stated as follows.

> From a set of functions $\{p(t)\}$, solve the integral equation (5.1) in the case where the distribution function $F(x)$ is unknown but a data set $\{x_1, x_2, \ldots, x_N\}$ drawn *i.i.d.* according to this distribution function is given.

Solving this problem for a wide class of functions $\{p(t)\}$ is an ill posed problem[3]. As an example consider the estimator that is a sum of delta functions at the data points. A small change in the data points results in a large change in the estimate. Therefore the need for some form of regularization arises.

Using the data, one can construct the sample distribution function $F_s(x)$ given by

$$F_s(x) = \frac{1}{N} \sum_{i=1}^{N} \Theta(x - x_i) \tag{5.2}$$

[2]We restrict ourselves to one dimension for most of our discussion of density estimation.
[3]A more detailed discussion of ill-posed problems can be found in part I

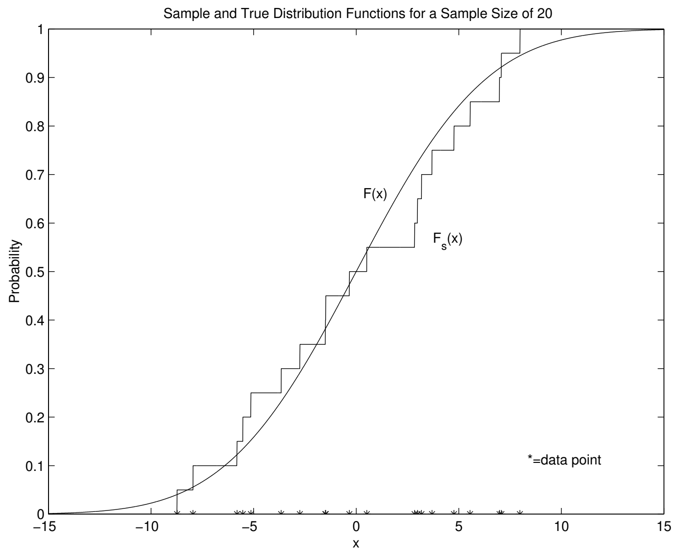


Figure 5.1: The sample distribution function.

where $\Theta(x) = 1$ if $x \geq 0$ and $\Theta(x) = 0$ otherwise. A sample distribution function is illustrated in figure 5.1. The uniform convergence [Serfling, 1980, pg 62]

$$\sup_x |F(x) - F_s(x)| \xrightarrow{N\to\infty} 0 \tag{5.3}$$

suggests that one can use $F_s(x)$ as an approximation to $F(x)$. Therefore, solving (5.1) with $F(x)$ replaced by $F_s(x)$ is the approach that we will take. Unfortunately, this alone will not suffice because as already mentioned, the density estimation problem is ill-posed. We need to restrict our space $\{p(t)\}$ by using regularization. In order to do this we have to make some assumptions on $F(x)$.

Naturally, one asks whether such assumptions can be avoided. Unfortunately the answer to this question is no. It can be shown that estimation of a probability density implies strong mode [4] estimation of the probability distribution [Vapnik, 1995,

[4] We say that $\hat{P}(A) = \hat{P}(x_1, \ldots, x_N)$ estimates the probability measure P in the strong mode if

$$\sup_{A\in\mathcal{F}} \left|P(A) - \hat{P}(A)\right| \xrightarrow{P} 0$$

as $N \to \infty$ where the *sup* is taken over all subsets of the sigma algebra of interest, $\mathcal{F}$.

pr 61]. It can also been shown (eg. [Barron *et al.*, 1992], [Devroye and Györfi, 1990]) that no estimator can estimate an arbitrary probability measure in the strong mode. Therefore it is *necessary* to make some assumptions on the probability measure in order to estimate the density. As a simple example, the estimator consisting of delta functions at each data point (appropriately normalized) is a legitimate density estimator, but this does not converge to the true density unless the true density happens to be a set of delta functions. Conversely, continuous density estimators will not (in general) converge to the true density if the true density happens to be a set of delta functions. Lacking any prior information, there is no reason to favor one of these methods over the other. We will impose smoothness constraints on the true density by bounding its derivatives.

This theme regarding the necessity of prior information has already appeared in the context of regularization and will appear again when we discuss natural priors. We will assume that the measure we are attempting to estimate has bounded derivatives.

5.2 Existing Density Estimation Techniques

How can one estimate the form of the density using a number of data points drawn from that density? Traditional density estimation methods can be grouped into two broad categories (see [Silverman, 1993], [Bishop, 1995a], and [Duda and Hart, 1973]). The first of these is the parametric approach. It assumes that the density has a specific functional form, such as a Gaussian or a mixture of Gaussians. The unknown density is estimated by using the data to obtain estimates for the parameters of the functional form. Typically, the parameters are estimated using maximum likelihood or Bayesian techniques. The drawback of the parametric approach is that the functional form of the density is rarely known beforehand, and the commonly assumed Gaussian or mixture of Gaussian models rarely fit densities that are encountered in practice.

The more common approach for density estimation is the non parametric approach, where no functional form for the underlying density is assumed. Rather than expressing the density as a specific parametric form, it is determined according to a formula involving the data points available. Non parametric density estimation belongs to the class of ill-posed problems in the sense that small changes in the data can lead to large changes in the estimated density. Therefore it is important to have methods that are robust to slight changes in the data. For this reason some amount of regularization is needed [Tikhonov and Arsenin, 1977], [Tikhonov, 1963]. The most common non parametric method is the kernel density estimator, also known as the Parzen window estimator [Parzen, 1962]. The density is estimated by summing "kernel functions" centered at each data point.

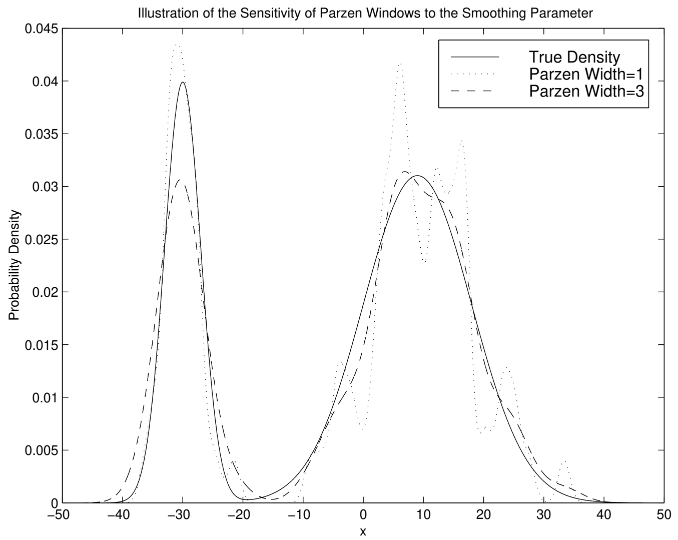
$$p(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\gamma(N)}K(\frac{x-x_i}{\gamma(N)}) \tag{5.4}$$

$\gamma(N)$ is the kernel width. Embedded in the choice of $\gamma(N)$ is the amount of regularization or smoothing. A typical kernel function is the "Gaussian bump."

$$K(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} \tag{5.5}$$

The problem with the kernel method is its extreme sensitivity to the choice of the kernel width ($\gamma(N)$). A wrong choice can lead to either under smoothing or over smoothing. Methods to estimate the kernel width are either asymptotic and given in terms of the unknown density, thus impractical, or are based on cross-validation, thus prone to statistical error. Another drawback of the kernel regression method is that it has the tendency to exhibit bumpy behavior at the tails. If the bumps are smoothed out by increasing the kernel width, then essential detail in the main part of the density is masked out or the width of the main part of the density will be exaggerated. We illustrate this behavior in figure 5.2.

The other common non parametric approach is the k-nearest neighbor technique [Fukunaga and Hostetler, 1973]. In this approach, the density at a particular point x is estimated as inversely proportional to V_k, the volume of the hypersphere centered



The Parzen estimate from 100 data points sampled from a mixture of two Gaussians. Illustrated here is the sensitivity to the smoothing parameter. With a small parameter (width=1), the detail is picked up but the function becomes extremely bumpy. A smoother function can be obtained at the expense of less detail by using a larger smoothing parameter (width=3). Looking at the curves, we see that the choice of the smoothness parameter can be critical.

Figure 5.2: Illustration of the sensitivity of the Parzen estimator to the smoothing parameter

at x, with radius equal to the Euclidean distance to the k^{th} nearest neighbor.

$$p(x) = \frac{1}{N}\frac{k-1}{V_k} \tag{5.6}$$

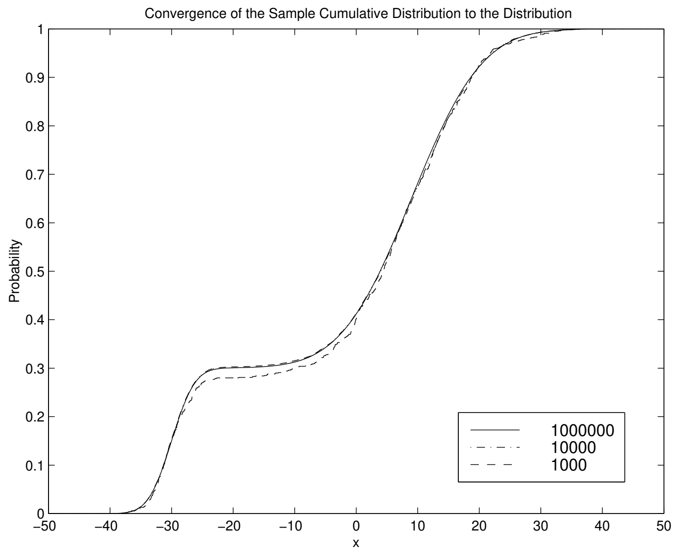
The k-nearest neighbor approach shares some of the drawbacks of the kernel density estimator, such as the difficulty of obtaining the best smoothing parameter k. In addition, the density estimate is not a smooth function, has a very heavy tail, and integrates to infinity on non-compact sets.

In spite of the importance of the density estimation problem, proposed methods using neural networks have been very sporadic in the literature. Many of them are similar to existing conventional methods, but developed in a neural network framework. Traven's approach [Traven, 1991] and Cwik and Koronacki's method

[Cwik and Koronacki, 1996] are based on mixture of Gaussian density estimation method. Schioler and Kulczyki's method [Schioler and Kulczyki, 1997] is based on the kernel estimation method. Roth and Baram [Baram and Roth, 1996] developed an elegant method using a one-layer network of sigmoidal nodes, and training it by maximizing the entropy of the outputs. Van Hulle [van Hulle, 1996] developed an interesting technique based on a self organizing approach, whereby the algorithm converges to a solution, such that at any point, the density of the weight vectors is an estimate of the unknown density. Modha and Fainman [Modha and Fainman, 1994] proposed a multilayer network, that is trained by maximizing the log-likelihood of the data. [Weigend and Srivastava, 1995] develop a fractional binning approach, developed in the context of time series prediction, that is based on partitioning the input space into bins, assigning a softmax output neuron for every bin, and training the network on the fraction of data falling in each respective bin. Smyth and Wolpert [Smyth and Wolpert, 1998] suggested a method for combining the estimates of several density estimators (such as kernel estimators or mixture of Gaussian estimators). The common factor in all these methods is their focus on estimating the density itself.

5.3 New Density Estimation Techniques

First we will propose two new methods for density estimation which can be implemented using multilayer networks. The approaches can thus be considered as semi-parametric models. The reason is that the model is described in terms of a number of parameters (the weights of the network), rather than the actual data points, but at the same time the number of parameters can be increased in a way that would ultimately achieve an all-powerful model capable of approximating any (continuous) density function. One important advantage of multilayer networks over local methods such as the kernel estimator is that they have been shown in theory and in practical problems to be superior for high-dimensional problems, for example approximation ability does not decay with increasing input dimension [Barron, 1993].

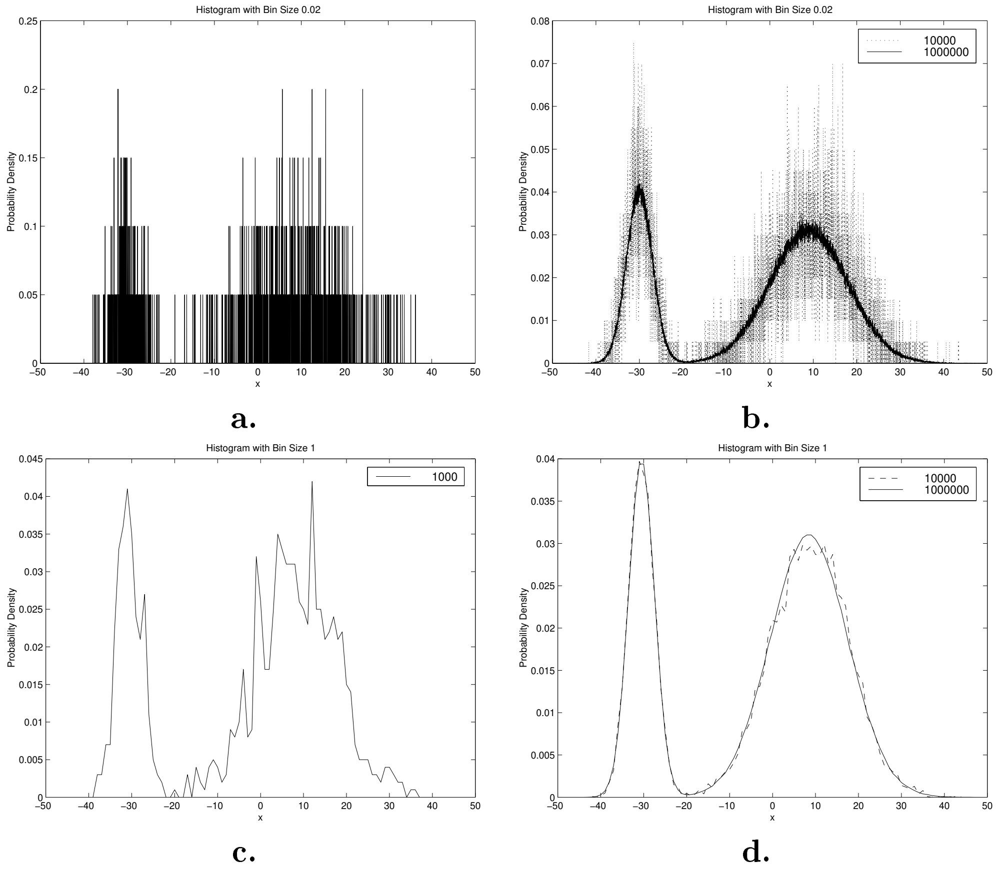


The sample distribution function for a mixture of two Gaussians is shown for 1000,10000 and 1000000 data points. Notice that the curve for 10000 data points is essentially overlapping the 1000000 data points curve.

Figure 5.3: The convergence of the sample distribution function.

Further, multilayer networks give us the flexibility to choose an error function to suit our application. The methods developed here are based on approximating the distribution function, in contrast to most previous works which focus on approximating the density itself. Straightforward differentiation then gives us the estimate of the density function. The distribution function is often useful in its own right - one can directly evaluate quantiles or the probability that the random variable occurs in a particular interval.

Figures 5.3 and 5.4 indicate the essential intuition behind why one might focus attention towards learning from the distribution function rather than from the density. One can see the higher sensitivity of the density estimator to finite sample size and the bin width (smoothness parameter). Too small a bin width increases the sensitivity to the 'noise' due to the small sample size. A larger bin width will smooth out this noise at the expense of a loss of information. The distribution estimator, however, does not appear as sensitive to the small sample size, and no smoothness parameter need be chosen. The regularization that is controlled by the bin width (or kernel width) is



The histograms of data drawn from data drawn from a mixture of two Gaussians. (a) and (b) are for a bin width of 0.02, and (c) and (d) are for a bin width of 1. The histograms for 1000, 10000, 1000000 points are shown. Notice the extreme sensitivity to the bin width.

Figure 5.4: Convergence of the histogram estimator.

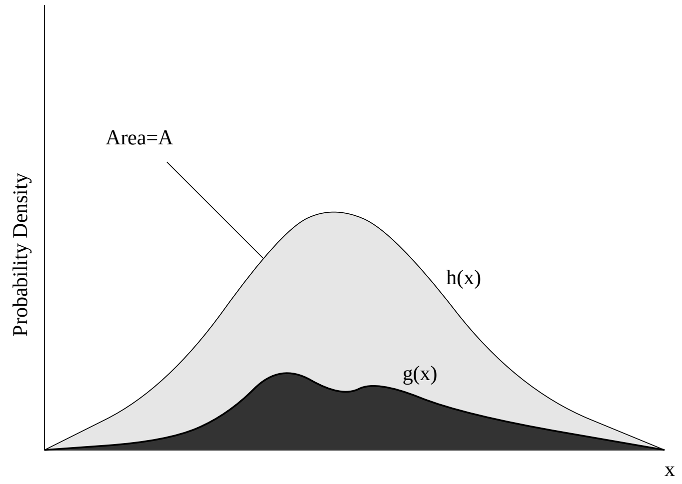
embedded in the integral operator that is used in passing from the histogram to the distribution.

We will develop two methods for the estimation of densities by first learning the cumulative and then taking the derivative of the resulting function. One of the techniques is based on a stochastic algorithm (SLC), and the second is based on a deterministic technique based on learning the cumulative (SIC). We will show that these two techniques yield equivalent results. The stochastic technique will generally be smoother on smaller numbers of data points, however, the deterministic technique is faster and applies to more that one dimension.

We will analyze the consistency and the convergence rate of the estimation error for our methods in the univariate case. When the unknown density is bounded and has bounded derivatives up to order K, we prove that the estimation error is $O((\log\log(N)/N)^{-(1-\frac{1}{K})})$, where N is the number of data points. As a comparison, for the kernel density estimator (with non-negative kernels) and the k^{th} nearest neighbor density estimator, the estimation error is $O(N^{-4/5})$, under the assumptions that the unknown density has a square integrable second derivative (see [Silverman, 1993]), and that the *optimal* kernel width is used, which is not possible in practice because computing the optimal kernel width requires knowledge of the true density. One can see that for smooth density functions with bounded derivatives, our methods achieve an error rate that approaches $O(N^{-1})$. We have found in the literature only the following methods that achieve a comparable error rate. The first is a kernel estimator with the specific choice of kernel that possesses zero l^{th} order moments for $l = 1, ..., L$, L being even. For such kernels, the convergence rate is $O(N^{-2L/(2L+1)})$. The problem, however, with this method is that the kernel has negative parts, and hence the resulting density estimate can be negative at some points and could therefore be an illegitimate density function. Because of this problem, this method is impractical and is not commonly used. The other method is the adaptive kernel estimator. The error rate is theoretically $O(N^{-8/9})$ for this method. However, the error estimate for this method is based on using the optimum smoothing function, which is given in terms of the unknown density and its derivatives. Even if one uses pilot estimates of these unknown characteristics of the true density to determine the smoothing function, the resulting statistical error will leave a residual $O(N^{-4/5})$ term that cannot entirely cancel out. The error estimate of our method, in contrast to these methods, does not rely on optimizing parameters based on the unknown density.

5.4 Random Variate Generation

The topic of random variate generation has been a vast research topic in the past three decades. In many scientific problems in areas such as physics, biology, engineering, and economics it is now more the rule than the exception to be confronted with problems where analytic solutions are not possible and simulating the given problem would be the only feasible approach. Problems which possess some random element will necessitate an accurate and fast random variable generation procedure for the Monte Carlo simulation to be efficiently performed. Examples of applications include the simulation of a biological system, such as how nerve cells interact together, the simulation of a communication network where the packets are assumed to arrive according to a particular density function, the simulation of a control system where the plant disturbance is a random variable, and the simulation of the stock market in an attempt to estimate a valuation for some derivative instruments. In the past, the assumption of standard density functions such as Gaussian or exponential densities for these random variables often allowed analytical solutions for these problems. But now, with more accuracy needed, the next step has been to assume more realistic density functions for the random variables (possibly estimated from the data). Unfortunately, there are very few techniques that can generate a random variable possessing an arbitrary density function (see [Press *et al.*, 1988] for a review of the approaches). One well known method is the transformation method: a uniformly distributed random number x is generated, and this number is passed through a non-linear transformation $y = G^{-1}(x)$, where $G(x)$ is the given (cumulative) distribution function. The problem with this method is that in the majority of cases $G^{-1}(x)$ (or even $G(x)$) is not known analytically. Solving for $G^{-1}(x)$ numerically is not practical since an essential feature of random number generation is speed. In Monte Carlo simulations, random variables are typically generated millions of times, in order to obtain statistically reliable results. There are many variations of the transformation method, such as obtaining transformations of several random variables (see [Knuth, 1981a]



The integrable function $h(x)$ dominates $g(x)$. A point is generated uniformly in the area A and then is accepted or rejected depending on whether it falls in the area under $g(x)$.

Figure 5.5: The rejection method for random variate generation.

and [Johnson, 1994]). These methods are largely ad hoc, and can be used to generate only some of the well-known density functions such as Gaussian, Gamma, Beta, etc.

The other method for random variable generation is the rejection method. It is a more general method than the transformation method. To use the rejection method, a comparison function $h(x)$ that dominates $g(x)$ everywhere is required (see figure 5.5). A point ($p = [p_x, p_y]$) is generated uniformly in the two-dimensional region bounded by the comparison function and the x-axis. For this purpose, H^{-1} is needed where $H(x) = \int_{-\infty}^{x} dt\ h(t)$. The point ($p$) will be accepted or rejected depending upon the realization of a second uniform $[0, 1]$ random variable (u) as follows. u is compared to $g(p_x)/h(p_x)$ and the point p is accepted if $u < g(p_x)/h(p_x)$. The problem with the rejection method is that for some large-tail densities it is not possible to find a comparison function with an analytically invertible distribution function (a necessary condition for the method) that is everywhere larger than the given density. Therefore, it cannot guaranteed that this method is capable of generating random variates from an arbitrary density. The acceptance rate is $1/A$ where A is the area under $h(x)$.

The acceptance rate can be small ($\tilde{0}.01$), thus slowing down the random variate generation process. Further, if the functional form of the given density is not known and it is characterized only by a finite set of data points (or say by a kernel density formulation), it becomes a computationally extensive procedure (requiring that we pass through all the points in the data set) to generate a single point.

Our goal is to develop several new random variable generating techniques which can be implemented using multilayer networks. The first method is based on viewing the random number generation as the inverse process of the density estimation. This method becomes the only computationally efficient method available for generating a random variable given a set of data points rather than a functional formulation. Two variants of this method are also proposed. We also propose a method that formulates the problem within a "control" framework. A cascade structure is proposed, that consists of a density shaping network (acting as the "controller"), and a density estimation network (acting as the "plant"). For the methods that we propose, the bulk of the time is spent in "learning" to generate the random variates. Once this learning is done, the actual generation of these random numbers is fast and can thus be used for Monte Carlo simulations.

The next chapter will develop the two new density estimation techniques, where we will also discuss the new, related, random number generating techniques. We will illustrate the techniques using neural networks. Then, in chapter 7 we will analyze the convergence characteristics of the two methods. We will show that the two methods are essentially equivalent, and then we will analyze convergence.

Chapter 6

New Density Estimation Techniques with Application to Random Variate Generation

— Traveller, there are no paths. Paths are made by walking.
– Antonio Machado

In this chapter we will propose two new methods for density estimation. For both methods, one can use a standard multilayer network, such as a one-hidden-layer or a two-hiden-layer network. We will use neural networks throughout for illustrative purposes, but stress that any sufficiently general learning model will do just as well. The network's output will represent an estimate of the distribution function, and its derivative will be an estimate of the density. We have already discussed the intuitive advantages of learning from the sample distribution function. We will now proceed to a description of the two methods.

6.1 SLC (Stochastic Learning of the Cumulative)

Let $x_n \in \mathbf{R},\ n = 1, ..., N$ be the given data points. Let the underlying density be $g(x)$ and its distribution function $G(x) = \int_{-\infty}^{x} g(t)dt$. We assume the density to be continuous and have continuous derivatives of all orders. Let the neural network output be $H(x, w)$, where w represents the set of weights of the network. Ideally, after training the neural network, we would like to have

$$H(x, w) = G(x). \tag{6.1}$$

Learning requires a set of targets. To determine a target for the network training, we make the following observation. The density of the variable $y \equiv G(x)$ (x being generated according to $g(x)$) is uniform in $[0,1]$. The reason is as follows. Let $z(y)$ represent the density of y. Using the well known formula for transformation of random variables,

$$z(y) = \frac{g(x)}{|\frac{dy}{dx}|} = \frac{g(x)}{|\frac{dG(x)}{dx}|} = 1 \qquad \text{for } 0 \leq y \leq 1, \tag{6.2}$$

and $z(y) = 0$ for $y < 0$ or $y > 1$. We used the fact that $g(x) = dG(x)/dx$. Thus, if $H(x, w)$ is to be as close as possible to $G(x)$, then the network output should have a density as close as possible to uniform in $[0,1]$. This is what our goal will be. We will attempt to train the network such that its output density is uniform. Having achieved this goal, the network mapping should represent the distribution function $G(x)$[1].

The basic idea behind the proposed algorithm is to use the N data points drawn from the unknown density as inputs to the network. For every training cycle we generate a different set of N network targets randomly from a uniform distribution in $[0,1]$, and adjust the weights to map the data points (sorted in ascending order) to these generated targets (also sorted in ascending order). Thus we are training the network to map the data to a uniform distribution.

Before describing the steps of the algorithm, we note that the resulting network has to represent a monotonically nondecreasing mapping, otherwise it will not represent a legitimate distribution function. Let $M = \{w | \partial H(x,w)/\partial x \geq 0, \ \forall x\}$. Then we wish to pick $w \in M$, such that $H(x,w)$ is as close as possible to $G(x)$. In practice, enforcing monotonicity could be done by using a class of "monotonic networks" [Sill, 1998b], or else by using *hint* penalty terms [Abu-Mostafa, 1995], [Abu-Mostafa, 1990]. Hints are auxiliary information or constraints on the target function, that are known *a priori* (independent of the training examples). By using hints in the form of penalty terms,

[1] A similar philosophy is used in [Baram and Roth, 1996], where they show that their entropy maximization method implies transformation to a uniform density (and also implies the necessity to maximize the expected logarithm of the determinant of the transformation Jacobian).

we can guide the learning process, and obtain a network that satisfies the hints. In our simulations, we used the latter approach of adding a term, that penalizes non-monotone mappings to the error function. The proposed algorithm is as follows.

1. Let $x_1, x_2, ..., x_N$ be the points drawn from the unknown density. Without loss of generality assume the points are sorted in ascending order: $x_1 \leq x_2 \leq \cdots \leq x_N$.

2. Set $t = 1$, where t is the training cycle number. Initialize the weights to $w(1)$. This is usually done randomly.

3. Generate randomly from a uniform distribution in $[0, 1]$ N points $u_1', u_2', ..., u_N'$. These are the network targets for this cycle.

4. Sort the targets in ascending order, i.e. $u_1 \leq u_2 \leq ... \leq u_N$ (where we have renamed the ordered targets u_n). Then, the point u_n is the target output for x_n.

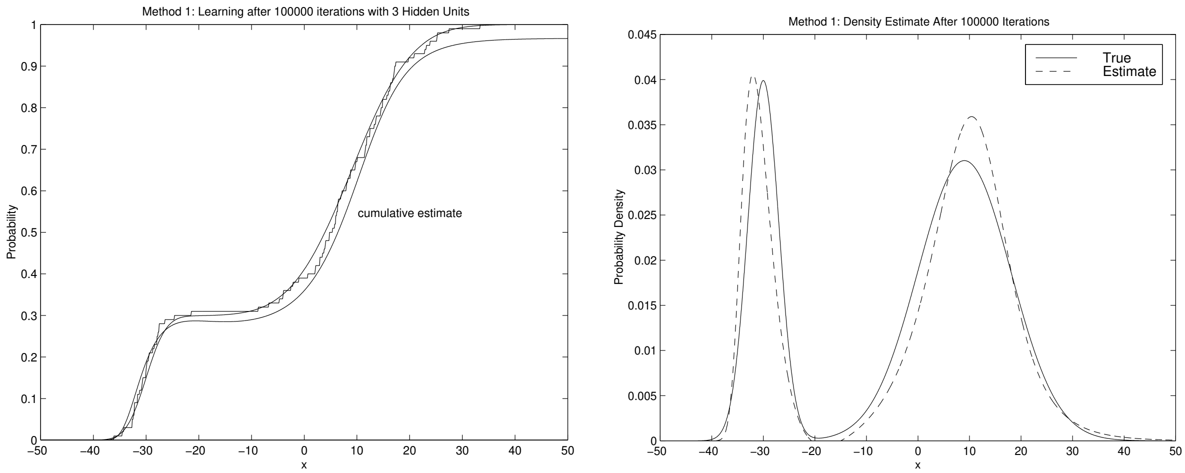
5. Adjust the network weights according to the backpropagation scheme:

$$w(t+1) = w(t) - \eta(t)\frac{\partial \mathcal{E}}{\partial w} \tag{6.3}$$

where $\mathcal{E}$ is the objective function that includes the error term and the monotonicity hint penalty term:

$$\mathcal{E} = \sum_{n=1}^{N}\Big[H(x_n) - u_n\Big]^2 + \lambda\sum_{k=1}^{N_h}\Theta\Big(H(y_k) - H(y_k+\Delta)\Big)\Big[H(y_k) - H(y_k+\Delta)\Big]^2 \tag{6.4}$$

where we have suppressed the w dependence. The second term is the monotonicity penalty term, λ is a positive weighting constant, Δ is a small positive number, $\Theta(x)$ is the familiar unit step function, and the y_k's are any set of points where we wish to enforce the monotonicity. Because the hint is known to be true, λ can be chosen very large.



(a) **(b)**

The cumulative estimate after 100000 iterations of SLC is shown in (a) along with the sample and true distribution functions. The implied density is shown in (b). The data set consisted of 100 points sampled i.i.d. from the mixture of two Gaussians shown in (b).

Figure 6.1: Stochastic learning of the probability density function.

6. Set $t = t + 1$, and go to step 3 to perform another cycle until the error is small enough.

7. Upon convergence, the density estimate becomes

$$\hat{g}(x) = \frac{\partial H(x, w)}{\partial x} \tag{6.5}$$

Note that as presented, the randomly generated targets are different for every cycle, which will have a smoothing effect that will allow convergence to a truly uniform distribution. One other version, that we have implemented in our simulation studies, is to generate new targets after every fixed number L of cycles, rather than every cycle. This will generally improve the speed of convergence as there is more "continuity" in the learning process. Also note that it is preferable to choose the activation function for the output node to be in the range of 0 to 1, to ensure that the estimate of the distribution function is in this range. A sample run of how SLC performs on 100 data points drawn from a mixture of two Gaussians is shown in figure 6.1.

SLC is only applicable to estimating univariate densities. The reason is that for the multivariate case, the nonlinear mapping $y = G(x)$ will not necessarily result in a uniformly distributed output y. Fortunately, many, if not the majority of problems encountered in practice are univariate. This is because multivariate problems, with even a modest number of dimensions, need a huge amount of data to obtain statistically accurate results. Our second method, described next, is applicable to the multivariate case as well.

6.2 SIC (Smooth Interpolation of the Cumulative)

Again, we have a multilayer network, to which we input the point $\mathbf{x}$, and the network outputs the estimate of the distribution function. Let $g(\mathbf{x})$ be the true density function, and let $G(\mathbf{x})$ be the corresponding distribution function. Let $\mathbf{x} = (x^1, ..., x^d)^T$. The distribution function is given by

$$G(\mathbf{x}) = \int_{-\infty}^{x^1} \cdots \int_{-\infty}^{x^d} g(\mathbf{x}) dx^1 \cdots x^d, \tag{6.6}$$

a straightforward estimate of $G(\mathbf{x})$ could be the fraction of data points falling in the area of integration:

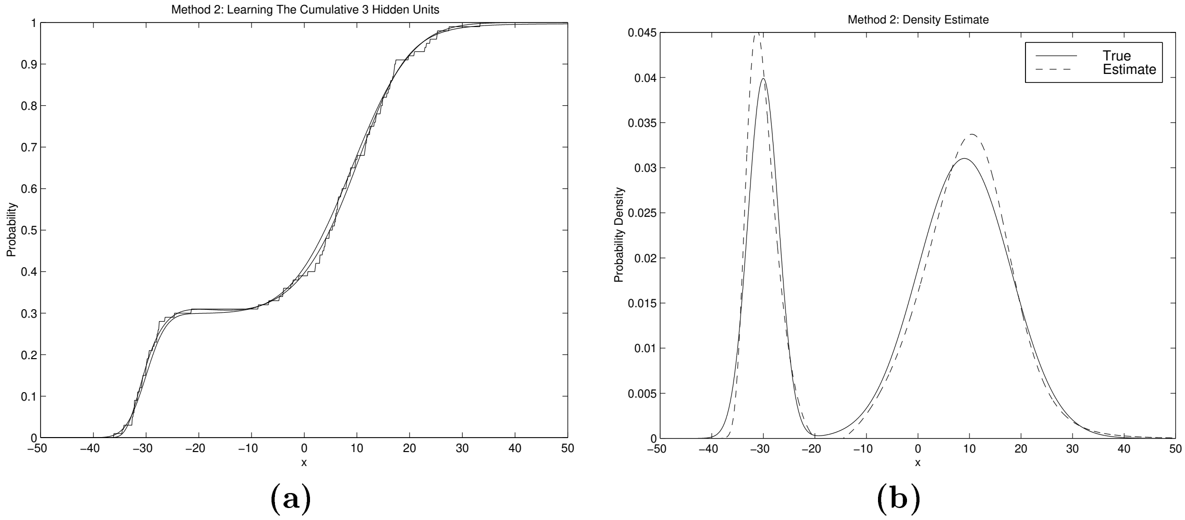
$$\hat{G}(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} \Theta(\mathbf{x} - \mathbf{x}_n), \tag{6.7}$$

where Θ is defined as

$$\Theta(\mathbf{x}) = \begin{cases} 1 & \text{if } x^i \geq 0 \text{ for all } i = 1, \ldots, d, \\ 0 & \text{otherwise.} \end{cases}$$

The statistical properties of the estimate (6.7) have been widely studied ([Serfling, 1980], [Finkelstein, 1971]). For the method we propose, we will use such an estimate for the target outputs of the neural network.

The estimate given by (6.7) has a staircase-like shape if plotted against $\mathbf{x}$, and thus



The cumulative estimate after 10000 iterations of conjugate gradient is shown in (a) along with the sample and true distribution functions. The implied density is shown in (b). The data set consisted of 100 points sampled i.i.d. from the mixture of two Gaussians shown in (b).

Figure 6.2: Smooth interpolation by a neural network to estimate a probability density function.

is discontinuous (see for example figure 5.1). The neural network method developed here provides a smooth, and hence more realistic estimate of the distribution function. Further, the density can be obtained by differentiating the output of the network with respect to its inputs. To our knowledge, there is no density estimator in the literature that is based on interpolating estimates of the distribution function, and taking the derivative to obtain the density.

For the low-dimensional case, we can uniformly sample (6.7) using a grid, to obtain the examples for the network. Beyond two or three dimensions, this is not feasible of course because of computational considerations. One idea is to sample the input space randomly (using say a uniform distribution over the approximate range of $\mathbf{x}_n$'s), and for every point determine the network target according to (6.7). Another option is to use the data points themselves as examples. The target for a point $\mathbf{x}_m$ would then be

$$\hat{G}(x_m) = \frac{1}{N-1} \sum_{n=1,\ n \neq m}^{N} \Theta(x_m - x_n). \tag{6.8}$$

We also use monotonicity as a hint to guide the training. Once training is performed, and $H(\mathbf{x}, w)$ approximates $G(\mathbf{x})$, the density estimate can be obtained as

$$\hat{g}(\mathbf{x}) = \frac{\partial^d H(\mathbf{x}, w)}{\partial x^1 \cdots \partial x^d}. \tag{6.9}$$

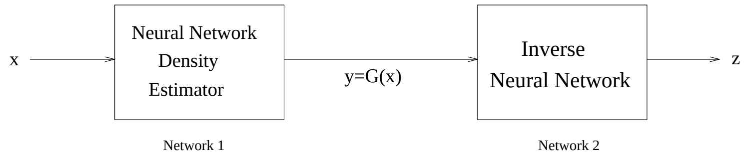
We note that for a few dimensions, a derivation of the derivatives in (6.9) will be straightforward. For larger dimensions a numerical differentiation scheme such as the simple differencing method is more feasible. A sample run of how SIC performs on the same 100 data points used to run SLC is shown in figure 6.2.

6.3 Application to Random Variate Generation

6.3.1 Learning $G^{-1}(x)$ (SLCI and SICI)

Suppose that we wish to generate random variates from a specified univariate density $g(x)$. It could also be that $g(x)$ is represented only by a number of data points drawn from it, rather than a functional form. Typical approaches to such a problem would estimate the density from the given data, and apply some of the well-known methods such as the transformation method or the rejection method (which would be computationally extensive, since a pass through all the given data points has to be performed to generate a single point).

We propose a method implementable by multilayer networks, that is inspired by the transformation method. But, unlike the transformation method, we do not assume that $G^{-1}(x)$ is known analytically. The network learns to implement the transformation $G^{-1}(x)$. The basic structure of the method is illustrated in figure 6.3. It consists of two cascaded multilayer networks. The first network is trained to estimate the distribution function $G(x)$ from the data using one of the techniques



The first network learns $G(x)$. This can be done using the data available or if one has $g(x)$, one generates according to $g(x)$ using (say) the rejection method. Thus generating the data for this learning may be slow, but once the learning has taken place, Network 1 implements $G(x)$. Now for an arbitrary set of inputs, we take Network 1's outputs as the inputs to Network 2 and the target are the inputs that went into Network 1. Thus we train Network 2 to implement the inverse of Network 1 which should be G^{-1}.

Figure 6.3: Training a network to implement $G^{-1}(x)$.

described in the previous sections. Once the first network is trained, the density of its output is uniform in $[0, 1]$ as discussed in sections 6.1 and 6.2. Then, we train the second network to invert the mapping produced by first network. This inversion can be accomplished to an arbitrary precision by using as large a network and as many data points as we want. Learning the inverse proceeds as follows. Let $H_1(x, w)$ and $H_2(x, v)$ be the functions implemented by Network 1 and Network 2 respectively. Let $\{x_i\}$ be an arbitrary set of inputs. Then, the input/output examples for Network 2 will be $\{H_1(x_i, w), x_i\}$. Once the entire training process is complete, random variate generation according to $g(x)$ is a done by passing a uniform deviate in $[0, 1]$ (y) through Network 2, i.e., $H_2(y, v) \sim g(x)$. To see why this is so, observe that for the cascade structure of the two networks in figure 6.3, the density of the output z is equal to density of the input x because Network 2 is the inverse of Network 1. Since the density of the output y of Network 1 is uniform, inputing a uniform deviate (y) into Network 2 should produce a variable (z) having a density equal to that of x. More formally, assume that Network 1 was implementing $G(x)$. Then, Network 2 is implementing $G^{-1}(x)$. Therefore, $z = G^{-1}(y)$ where y has a uniform distribution. Using the formula for transformations of random variables, the density g_Z of the output z is evaluated as

$$g_Z(z) = \frac{g_Y(y)}{|\frac{dG^{-1}(y)}{dy}|} = \frac{1}{|1/g(z)|} = g(z). \tag{6.10}$$

This method applies only for the one-dimensional case because the mapping $G(x)$ transforms x into a uniform density only for the univariate case. A second method based on a control formulation of the problem, described later, is more general and applies to the multi-dimensional case as well.

It might seem wasteful to first learn $G(x)$,and then invert it. Perhaps a direct method for learning $G^{-1}(x)$ would be more efficient. Along these lines, we propose two other approaches which are inspired by the density estimation techniques themselves. They are variants of the method just described in that they build upon the idea that a network mapping a uniform distribution to the true distribution must be implementing $G^{-1}(x)$.

SLCI (Stochastic Learning of G Inverse)

This technique is very similar to the stochastic method for estimating $G(x)$ and can be viewed as the inverse of Method 1. Once again, G^{-1} is a monotonic function so a monotonicity hint should be used here as well. The algorithm is as follows.

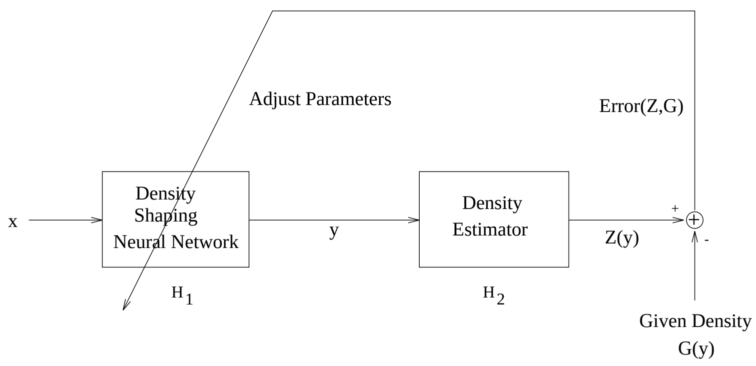
1. Sort the data points $x_1 \leq x_2 \leq \cdots \leq x_N$
2. Set $t = 1$ and initialize the weights of the network to $w(1)$.
3. Generate N numbers $(\{u_i\}_{i=1}^N)$ from a uniform density in $[0, 1]$ and sort them so that $u_1 \leq u_2 \leq \cdots \leq u_N$.
4. Train the network to map input u_n to output target x_n exactly as in steps 5 and 6 of the stochastic density estimation technique. Every cycle or every L cycles generate new u_n's.

5. After training is complete, input to the network a uniformly generated number. The output is distributed according to $G(x)$.

SICI (Smooth Interpolation of G Inverse)

This approach is analogous to the smooth interpolation approach for estimating the density. It is identical to SLCI, except that the input examples are $(i-1)/(N-1)$ (corresponding to output example x_i), instead of the uniform deviates.

6.3.2 Control Formulation of Random Variate Generation



The first network shapes the density of x. The transformed density is estimated, and then compared with the true density. The parameters of the first network are altered until the transformed density matches the required density.

Figure 6.4: Control formulation for random variate generation.

We propose a novel formulation of random number generation within a "control framework". Figure 6.4 illustrates the basic idea. As with the previous method, the structure consists of two cascaded units. A random variable x is generated according to any standard density function, for example Gaussian or uniform. In principle, an unlimited number of x's can be generated. This random variable is the input to the composite structure. The first network acts as a nonlinear transformation that shapes the density of x. Let the density of the transformed variable be $Z(y)$ where y is the transformed variable. The second unit serves to estimate the distribution function, for example SLC or SIC (sections 6.1 and 6.2) could be used. The output of the second unit, is the estimate of the distribution function $Z(y)$ of Network 1's output. Let the desired distribution function be $G(y)$. An error signal, $\mathcal{E}(G(y), Z_{w_1}(y))$ can now be generated where we have explicitly shown that the distribution Z is a function of w_1, the weights of the first network. This error signal can be backpropagated to adjust the weights of Network 1 so as to minimize $\mathcal{E}(G(y), Z_{w_1}(y))$. Such a training process would change Network 1's mapping in such a way that would shape the distribution function of y to get it as close as possible to the desired distribution function. Once training is complete, we can generate numbers according to the density $G(x)$, by generating x according to the standard density, and then passing it through Network 1.

Note 1: This algorithm is not restricted to the case of scalar random variable, and so can be used to generate multi-dimensional variables.

Note 2: To generate from a density represented only by a set of data points, we use the same method above but replace $G(y)$ by $\hat{G}(y)$, an estimate of $G(y)$ using the data points. This estimate can be obtained using one of the methods described in sections 6.1 and 6.2.

Note 3: The entire learning process can be an involved process. However, once the learning has been done, the generation of random variates is fast.

The analogy with the control problem is as follows. The second unit represents the "plant," and Network 1 is the "controller." The controller is trained, such that the plant produces the desired output[2]. The details of the algorithm are as follows:

I: Initialization:

[2]This technique has a similar form to a neural network control structure (see [Narendra and Parthasarathy, 1990]) where, typically, a neural network is trained to identify the plant (estimate the plant output), analogous in our set up to the density estimation unit. Then, a neural network controller is trained to control the plant – i.e., the identification network estimates the plant output and the controller network is altered until this output is as desired.

1. Let the desired density be $g(y)$ and the corresponding distribution function $G(y)$. Generate N samples $x_1, \cdots, x_N \in \mathbf{R}^d$ according to any standard density such as a Gaussian density. The larger N is, the more accurate the final result will be.

2. Let $H_1(x, w)$ and $H_2(x, v)$ be the functions implemented by Network 1 and Network 2 (where we have assumed that the density estimator is a neural network as in the previous sections). Network 1 has d outputs, and Network 2 has one output. Initialize the weights of the networks.

3. Calculate Network 1's output: $y_n(w) = H_1(x_n, w),\ n = 1, \cdots, N$.

4. Train Network 2 using SLC or SIC in order to estimate the distribution of the y_n's. Thus, $H_2(y, v(w))$ is implementing the distribution function of y, where we have explicitly shown the w dependence.

II: Training Iterations:

5. Suppose the error function we wish to minimize is $\mathcal{E}(H_2(w), G)$. Possible error functions are an L_1 or L_2 norm, or a cross entropy (Kulback-Leibler distance). For the case of squared error, let $\{z_n\}_{n=1}^{M}$ be any set of points for calculating the error function ($\mathcal{E}$),

$$\mathcal{E} = \sum_{n=1}^{M} \Big[H_2(z_n, v(w)) - G(z_n) \Big]^2 \tag{6.11}$$

6. One alters the weights (w) of the first network in the direction of the negative gradient:

$$w(t+1) = w(t) - \eta \frac{\partial \mathcal{E}}{\partial w} \tag{6.12}$$

For example in the case of squared error we have,

$$\frac{\partial \mathcal{E}}{\partial w} = 2 \sum_{n=1}^{M} \Big[H_2(z_n, v(w)) - G(z_n) \Big] \frac{\partial H_2(z_n, v(w))}{\partial w} \tag{6.13}$$

As an approximation to step six, we can let the arbitrary set z_n be y_n, the outputs of the first network. We leave $v(w)$ fixed for a certain number (L) of training cycles. Then the change in $\mathcal{E}$, as w changes is due to a change in y_n. By the chain rule we have

$$\frac{\partial \mathcal{E}}{\partial w} = 2\sum_{n=1}^{N}\Big[H_2(y_n, v(w)) - G(y_n)\Big]\frac{\partial H_2(y_n, v)}{\partial w} \tag{6.14}$$

$$= 2\sum_{n=1}^{N}\Big[H_2(y_n, v(w)) - G(y_n)\Big]\frac{\partial H_2(y_n, v)}{\partial y_n}\frac{\partial H_1(x_n, w)}{\partial w} \tag{6.15}$$

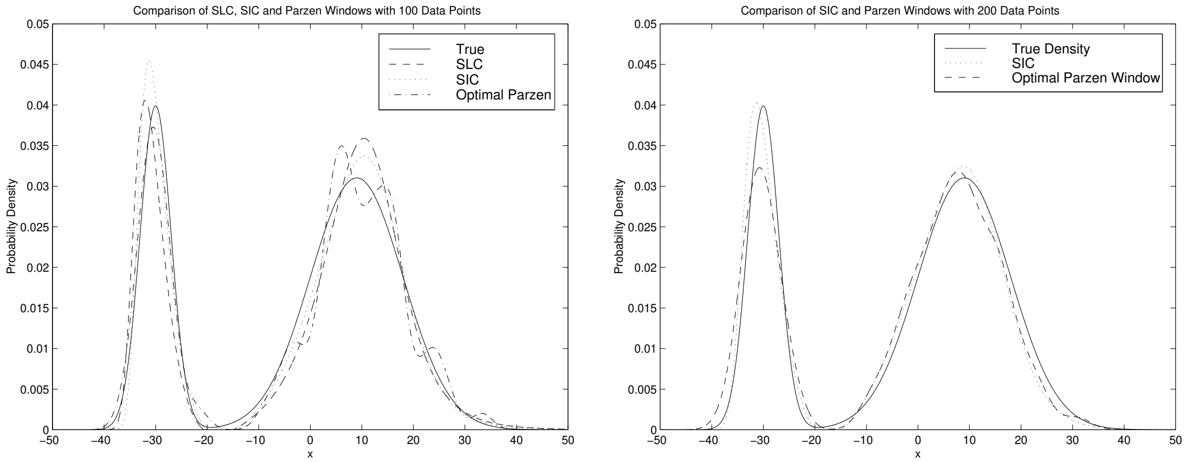
The weights in the first network can now be updated using a standard standard backpropagation scheme [Hertz *et al.*, 1991, pg 115], where one backpropagates the δ's first through Network 2, then through Network 1 and one only adjusts the weights of Network 1. We will specialize the remainder of the algorithm to this case. Appendix 6.6 considers the weight update rule for the general case where the density estimation is done using SIC[3].

7. After L cycles of weight update for Network 1 in step 6, train Network 2 for a few cycles on the new $y_n's$, to allow it to track the small change in the distribution of y. Thus we now change v which was kept constant in step 6.

8. Go to step 6 to perform another iteration of training, and continue doing so until the error becomes small enough.

9. After training is complete, the random numbers can be generated by generating x according to the standard density that was innitially used (e.g. Gaussian), and computing $y = H(x, w^*)$ (where w^* represents the final weights after the entire learning process). Now, y should be the distributed according to $g(y)$.

[3] It will be shown in the next chapter that SLC is essentially equivalent to SIC.



(a) **(b)**

Plotted are the true density and the estimates (SLC, SIC, Parzen). In (a), 100 data points were used, and in (b) 200 data points were used. The optimal Parzen window width (h) can be calculated for the true density as $h \approx 5.12/n^{1/5}$ *[Silverman, 1993, pg 40]. Though this is not possible in practice, we use it here for comparison. Notice that even the optimal Parzen window is bumpy, whereas the neural networks, which are also capable of implementing any density (by using arbitrarily many hidden units) is smooth.*

Figure 6.5: Comparison of optimal Parzen windows, with neural network estimators.

6.4 Simulation Results

We tested our techniques for density estimation on data drawn from a mixture of two Gaussians:

$$g(x) = \frac{3}{10}\frac{1}{\sqrt{18\pi}}e^{-\frac{(x+30)^2}{18}} + \frac{7}{10}\frac{1}{\sqrt{162\pi}}e^{-\frac{(x-9)^2}{162}} \tag{6.16}$$

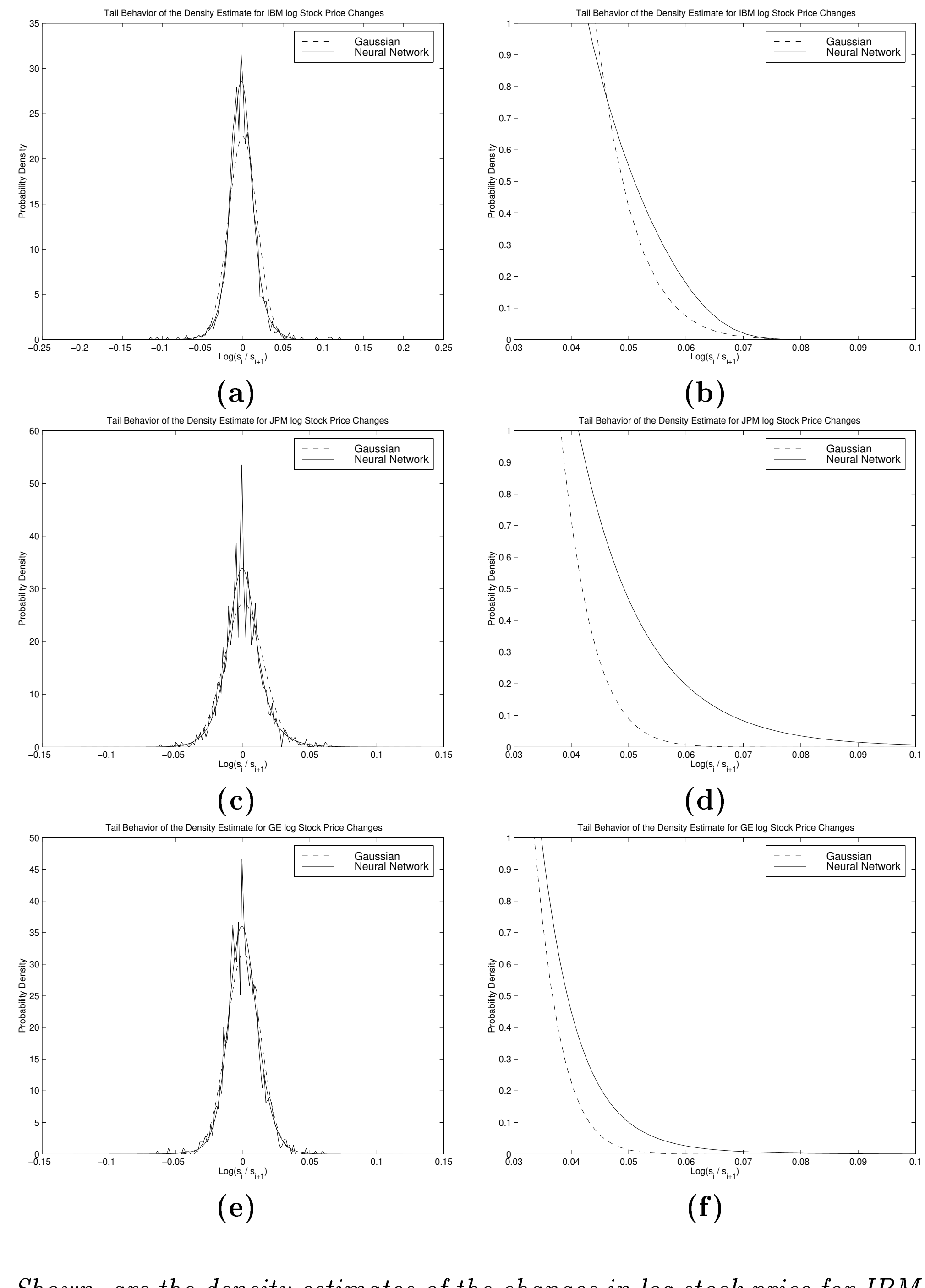
Data points were randomly generated and the density estimates using SLC or SIC (for 100 and 200 data points) were compared to the Parzen technique. Learning was done with a standard 1 hidden layer neural network with 3 hidden units. The hidden unit activation function used was *tanh* and the output unit was an *erf* function. A set of typical density estimates are shown in figure 6.5. A monotonic mapping was obtained by enforcing the monotonicity hint as a penalty with a weight 10000, and the learning algorithm used was conjugate gradient. As will become clear in the next chapter, one wishes to pick the "smoothest" interpolator. We softly enforced this smoothness by starting at small weights. As can be seen from figure 6.5, our method gives smoother results that the Parzen estimate. This will become quantified in the next chapter.

As an application we estimated the density of stock price changes for a group of companies. Refer to chapter 4 or [Wilmott *et al.*, 1995] for a more detailed discussion of modeling stock price change using stochastic differential equations. Essentially, the ratio $\log(S_{i+1}/S_i)$ should have a Gaussian distribution according to the Black-Scholes formulation. Using our techniques we estimate the densities for this ratio, illustrated in figure 6.6. We plot the estimated density along with the true histograms and the Gaussian with the same mean and variance (figure 6.6) for three representative companies (IBM, GE, JP Morgan). Notice how the true distribution significantly deviates from the Gaussian, displaying the well established fat-tailed behavior.

6.5 Comments

We have developed two techniques for density estimation based on the idea of learning the cumulative by mapping the data points to a uniform density. In so doing, we placed the density estimation problem within the supervised learning framework. Two techniques were presented, a stochastic technique (SLC), which is expected to inherit the characteristics of most stochastic iterative algorithms, and a deterministic technique (SIC). Both methods learn the distribution function by mapping the data to a uniform distribution. SLC tends to be slow in practice, however, because each set of targets is drawn from the uniform distribution, this is anticipated to have a smoothing/regularizing effect – this can be seen by comparing SLC and SIC in figure 6.5 (a). A similar outcome is obtained by adding small random perturbations to the targets of SIC. The distribution of these perturbations (for target $y_i = i/N+1$ should be a β-distribution with parameters i,(N+1-i).

Extensions along these lines are possible: why restrict oneself to a uniform density?



Shown are the density estimates of the changes in log stock price for IBM, JP Morgan (JPM) and General Electric (GE) using SIC. For each company, about 1530 data points were available. Also shown for comparison is the Gaussian with the same mean and variance as the data. The estimated distribution is significantly non-Gaussian. We magnify the tail behavior in (b), (d) and (f). Notice that the true density has a considerably fatter tail. (The data was obtained from [Momentum, 1996]

Figure 6.6: Density estimates for the log stock price changes of some U.S. stocks.

One could map to any standard density. For example, one can replace the uniform targets by targets drawn from (say) a Gaussian density. Let the network function being implemented be $H(x, w)$, and let the mapping that converts a Gaussian random variable to a uniform be $Q(x)$. This is just the error function $erf(x)$[4]. Then the composite mapping

$$Q(H(w, x))$$

is the required estimate $G(x)$, from which the density can be obtained by differentiation with respect to x. Thus our methods could be extended by mapping to any standard density for which $Q(x)$ is known analytically. The advantage of doing this can be seen by supposing that the true density is close to a Gaussian or some other standard density. Then we let the mapping $Q(x)$ "do the bulk of the work" and the neural network has to learn only the deviation of the true density from the standard density. This mapping could be considerably simpler to learn as it will by assumption be close to the identity mapping[5].

Our simulations demonstrated that the our techniques performed well in comparison to the Parzen technique using the optimal kernel width, which is unrealizable in practice. Further, there is no smoothing parameter that needs to be chosen. Smoothing occurs naturally by picking the interpolator with the lowest bound for a certain derivative – that this is what we should do becomes clear in the next chapter, where we consider the theoretical issues of convergence. In our simulations, we enforced smoothing by starting the network at small weights. For our methods, the majority of time is spent in the learning phase, but once learning is done, evaluating the density is fast. We used our methods to demonstrate the fat-tailed behavior in the stock markets – price changes in the stock markets obey a distribution that has a fatter-than-Gaussian tail.

---

[4] $erf(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{x^2}{2}}$

[5] A technicality in the proof of convergence with probability 1 requires that the random targets be bounded with probability 1. However we can ignore this technical requirement without significant practical loss.

We then developed techniques for non-uniform random variate generation that are applicable to generation of random variates from a density represented by a set of data points (SLCI or SICI) or from a density whose functional form is given, where we formulated the task as a control problem. Our methods are applicable to any density, and the generation of random variates is fast, once the learning is performed. The learning phase is the rate limiting step, but this will become insignificant when one needs to generate millions of points, many times.

In this chapter we have developed the techniques for density estimation. The techniques for random variate generation rely on the density estimation techniques. In the next chapter, we will discuss the theoretical soundness of these techniques in order to justify their use.

6.6 Appendix

In this appendix we will derive the exact weight update rule for the "control" algorithm in section 6.3.2. Suppose the error function we wish to minimize is $\mathcal{E}(H_2(w), G)$. Possible error functions are an L_1 or L_2 norm, or a cross entropy (Kulback-Leibler distance). We consider here the squared error. A similar analysis can be done for other error functions. Let $\{z_n\}_{n=1}^{M}$ be any set of points for calculating the error function ($\mathcal{E}$),

$$\mathcal{E} = \sum_{n=1}^{M} \Big[H_2(z_n, v(w)) - G(z_n) \Big]^2 \tag{6.17}$$

One alters the weights (w) of the first network in the direction of the negative gradient:

$$w(t+1) = w(t) - \eta \frac{\partial \mathcal{E}}{\partial w} \tag{6.18}$$

For the case of squared error we have,

$$\frac{\partial \mathcal{E}}{\partial w} = 2 \sum_{n=1}^{M} \Big[H_2(z_n, v(w)) - G(z_n) \Big] \frac{\partial H_2(z_n, v(w))}{\partial w} \tag{6.19}$$

We need to compute $\frac{\partial H_2(z_n, v(w))}{\partial w}$, which by the chain rule we can write as

$$\frac{\partial H_2((z_i), v(w))}{\partial w} = \sum_k \frac{\partial H_2((z_i), v)}{\partial v_k} \frac{\partial v_k}{\partial w} \tag{6.20}$$

When using SIC as the density estimator, v is a minimizer of the error function

$$\hat{\mathcal{E}} = \sum_i \left(H_2(y_i, v) - \frac{i}{N+1} \right)^2 \tag{6.21}$$

Define the matrix of second derivatives (the Hessian) of $\hat{\mathcal{E}}$ with respect to v by

$$\mathbf{H}_{ij} = \frac{\partial^2 \hat{\mathcal{E}}}{\partial v_i \partial v_j} \tag{6.22}$$

which is a function of v (and implicitly of w). Then, the lowest order term in the Taylor expansion of $\hat{\mathcal{E}}$ around v is a quadratic:

$$\hat{\mathcal{E}}(v + \Delta v) = \hat{\mathcal{E}}(v) + \Delta v^T \mathbf{H} \Delta v \tag{6.23}$$

A change in w_i produces a change in the y_n's. This in turn produces a change in $\hat{\mathcal{E}}$:

$$\hat{\mathcal{E}} \to \hat{\mathcal{E}} + 2\Delta w_i \underbrace{\left(\sum_n (H_2(y_n, v) - \frac{i}{N+1}) \frac{\partial H_2(y_n, v)}{\partial y_n} \frac{\partial H_1(x_n, w)}{\partial w_i} \right)}_{\Omega_i(v,w)} \tag{6.24}$$

Therefore, if $w_i \to w_i + \Delta w_i$, in the limit of small Δw_i, $\hat{\mathcal{E}}(v + \Delta v)$ is given by (in the limit of small Δv)

$$\hat{\mathcal{E}}(v + \Delta v) = \hat{\mathcal{E}}(v) + \Delta v^T \mathbf{H}(v, w) \Delta v + 2\Delta v \cdot \nabla_v \Omega_i(v, w) \tag{6.25}$$

Setting the gradient of this expression (with respect to Δv) to zero, one obtains for the change in v

$$\Delta v = -\Delta w_i \mathbf{H}^{-1}(v, w) \nabla_v \Omega_i(v, w) \tag{6.26}$$

therefore we find that

$$\frac{\partial v}{\partial w_i} = -\mathbf{H}^{-1}(v, w) \nabla_v \Omega_i(v, w) \tag{6.27}$$

Therefore we finally get

$$\frac{\partial \mathcal{E}}{\partial w_i} = 2 \sum_{i=1}^{M} [(H_2)(z_i, v) - G(z_i)](\nabla_v H_2(z_i, v)) \cdot (-\mathbf{H}^{-1}(v, w) \nabla_v \Omega_i(v, w)) \tag{6.28}$$

and the weight update is given by

$$w_i(t+1) = w_i(t) - \eta \frac{\partial \mathcal{E}}{\partial w_i} \tag{6.29}$$

(6.29) can be used in (6.26) to obtain the weight updates (Δv) for the density estimating network.

Chapter 7

Convergence of the Density Estimation Techniques

> — *Now, here, you see, it takes all the running you can do to keep in the same place. If you want to get somewhere else, you must run twice as fast as that.*
>
> –*Lewis Carroll*

This chapter deals with the convergence properties of the density estimation techniques that we have proposed. Our goal is to justify the methods introduced in the previous chapter by showing that convergence to the true density does occur. Further, we analyze the rate of convergence and compare with various other estimators. Due to the technical nature of such convergence issues, we will take this opportunity to summarize the essential details of the chapter.

First, we will consider the stochastic method (SLC). The targets are uniform in $[0, 1]$. The expected value of the target u_i, the i^{th} order statistic, is $i/N + 1$, therefore we should expect the learned function to be approximately performing the mapping

$$x_i \rightarrow \frac{i}{N+1} \tag{7.1}$$

where x_i represents the i^{th} element of the ordered data set. But this is exactly the mapping we are trying to learn in SIC (smooth interpolation of $G(x)$). Thus we expect SLC to converge to a solution that is also a solution of SIC. The formal statement and proof of this claim are the contents of section 7.1. For the proof, we will need some results from recursive stochastic approximation theory, which we will review briefly.

Having shown that SLC→SIC, we can now restrict our analysis to SIC. First we define a set of functions which we call generalized sample distribution functions. These functions are exactly that set of functions that interpolate the sample distribution function data points. We will prove that this set of functions converges uniformly to the true distribution function in the L_1 and L_2 (MISE – mean integrated square error) sense. Thus, at least we do learn the cumulative. In fact we will show that the rate of convergence is $O(N^{-1/2})$ for L_1 and $O(N^{-1})$ for L_2. For comparison, the L_∞ convergence [Serfling, 1980, pg 62] is $\sqrt{\log\log(N)/N}$. Further, because the neural network is an arbitrarily powerful class of functions, these generalized distribution functions can be implemented, therefore the neural network implementations as we have used in the previous chapter also converge.

How about the convergence to the true density. As was discussed in chapter 5, some assumptions have to be made about the true density. We assume boundedness of the derivatives. Therefore the cumulative will be implementable by generalized distributions with bounded derivatives (in the asymptotic limit, with probability 1). We will then obtain the convergence to the true density, and further, the rate of convergence approaches N^{-1} (in L_2). We will show that when the true distribution function has K bounded derivatives, the convergence rate is $O((\log\log(N)/N)^{1-1/K})$ which as $K \to \infty$ is faster than $1/N^{1-\epsilon}$, $\forall\epsilon > 0$. For comparison, the Parzen estimate has a convergence rate of $N^{-4/5}$ for the *optimal* smoothing parameter, which is inaccessible in practice as it depends on the parameters of the true density. We will illustrate our convergence rate with simulations using neural networks.

We will then analyze the random number generation techniques (SLCI and SICI), which are essentially techniques for estimating G^{-1}. We will restrict the analysis to the case where G^{-1} is continuous on $[0, 1]$. We will show convergence of SLCI to SICI with probability 1 as $N \to \infty$, and then we will obtain the L_1 and L_2 convergence rates for SICI. They are $O(N^{-1/2})$ for L_1 and $O(N^{-1})$ for L_2.

7.1 Convergence of SLC to SIC

In this section we will analyze the stochastic method (SLC). Let $D = \{x_1, x_2, \ldots, x_N\}$ be the data points sorted in ascending order, and let $u_1(t), u_2(t), ..., u_N(t)$ be the output targets for training cycle t, where the $u_i(t)$'s are generated from a uniform density and then sorted in an ascending order. Ignore for the time being the effect of the hint penalty term in (6.4), as this term can only help convergence, and, if monotonicity is satisfied (as in the case of convergence to the true density), then the hint term will equal zero. The convergence behavior is given in the following theorem:

Theorem 7.1.1 *SLC converges with probability 1 to a local minimum of the error function*

$$\mathcal{E} = \sum_{n=1}^{N} \left[H(x_n, w) - \frac{n}{N+1} \right]^2 \tag{7.2}$$

provided that the learning rate $\eta(t)$ is a decreasing sequence, that satisfies the following conditions

a. $\sum_{t=1}^{\infty} \eta(t) = \infty$,

b. $\sum_{t=1}^{\infty} \eta^p(t) < \infty$, for some p,

c. $\lim_{t \to \infty} \sup[1/\eta(t) - 1/\eta(t-1)] < \infty$.

We note that the conditions on $\eta(t)$ guarantee that the learning rate is decreasing in a way that will dampen the random fluctuations around the minimum, but at the same time, not decreasing too fast to prevent reaching the minimum. A possible choice could be

$$\eta(t) = \frac{\eta'}{t} \tag{7.3}$$

where η' is a constant[1]. Before we prove the theorem, we will summarize some aspects of the theory of stochastic approximation, that will be the main tool in proving the theorem. Stochastic approximation deals with the convergence analysis of iterative

[1] $\eta(t) = \eta' / \log(t)$, however, would not work.

algorithms where the inputs are random variables. We follow the analysis of Ljung [Ljung, 1977], though other approaches that have different sets of assumptions, such as Kushner and Clark [Kushner and Clark, 1978] could also be used. For more details on stochastic algorithms in general, see [Benveniste *et al.*, 1990].

A general stochastic iterative algorithm is given by

$$w(t+1) = w(t) + \eta(t) Q(t, w(t), u(t)) \tag{7.4}$$

where $u(t)$ is a sequence of independent random variables (Ljung considers a more general form, where $u(t)$ is given by an iteration in terms of the random input and w). The algorithm converges with probability 1 (w.p.1) *only* to a stable stationary point of the ordinary differential equation[2]

$$\frac{dw}{dt} = \lim_{t\to\infty} E\Big[Q(t, w, u(t))\Big] \tag{7.5}$$

if the following assumptions are satisfied (see [Ljung, 1977]):

A1: $u(t)$ is bounded w.p.1.

A2: The function $Q(t, w, u)$ is continuously differentiable w.r.t. w and u, and the derivatives, for fixed w are bounded in t.

A3: $\eta(t)$ is a decreasing sequence, and

- $\sum_{t=1}^{\infty} \eta(t) = \infty$,
- $\sum_{t=1}^{\infty} \eta^p(t) < \infty$ for some p,
- $\lim\limits_{t\to\infty} \sup[1/\eta(t) - 1/\eta(t-1)] < \infty$.

A4: $\lim_{t\to\infty} E\Big[Q(t, \bar{w}, u(t))\Big]$ exists for all $\bar{w}$.

By observing the proof of convergence in [Ljung, 1977], it can be shown that condition A2 can be replaced by the following condition:

[2] In this chapter, we use $E[\cdot]$ to denote expectations.

A2′: Q is locally Lipschitz continuous in w and u, meaning that $|Q(t, w_1, u_1) - Q(t, w_2, u_2)| < R(w, u, \rho, v)(|w_1 - w_2| + |u_1 - u_2|)$, for $w_1,\ w_2 \in B(w, \rho)$ for some $\rho = \rho(w)$, and $u_1,\ u_2 \in B(u, v)$ for $v \geq 0$, where $B(x, y)$ is the ball centered at x with radius y. Moreover, the Lipschitz constant $R(w, u(t), 0, v(t))$ is a bounded function of t for fixed w and bounded $v(t)$.

To see why this is the case, we note that condition A2 is used in the proof in [Ljung, 1977] only in Lemma 2, pg. 569. There, the equivalent condition A2′ can be used in its place. Now, we can proceed to the proof of the theorem.

PROOF OF THEOREM 7.1.1: The update equation for SLC is given by

$$w(t + 1) = w(t) + \eta(t)Q(w(t), u(t)) \tag{7.6}$$

where $u(t)$ is the vector of u_n's that are presented for training cycle t, and

$$Q(w(t), u(t)) = -\sum_{n=1}^{N} \Big[H(x_n, w(t)) - u_n(t)\Big] \frac{\partial H(x_n, w(t))}{\partial w} \tag{7.7}$$

Note that x_n is fixed and considered constant in the proof, because it does not change from one cycle to the next. The first condition, the independence of successive $u(t)$ vectors, is satisfied, by the construction of the algorithm. We now verify that conditions A1, A2′, A3 and A4 are satisfied.

- Condition A1 is satisfied, because $u_n(t)$ is generated from a uniform density, and hence it is bounded.

- Q is a piecewise (with respect to t) differentiable function of u and w (assuming standard sigmoidal neural networks). Further, the derivatives are bounded in t for fixed w because $u_n(t)$ is bounded in t. Therefore, Q is locally Lipschitz. In addition, the local Lipschitz constant $R(w, u(t), 0, v(t))$ is bounded in time (for fixed w) as the derivatives are bounded in time for fixed w. Hence condition A2′ is satisfied.

- Condition A3 is equivalent to the conditions of the theorem.
- Let us now consider Condition A4:

$$\lim_{t\to\infty} E\Big[Q(w,u(t))\Big] = -\lim_{t\to\infty}\sum_{n=1}^{N}\Big(H(x_n,w) - E[u_n(t)]\Big)\frac{\partial H(x_n,w)}{\partial w} \tag{7.8}$$

  The variables $u_n(t)$ are the order statistics of a sample of size N drawn from a distribution that is uniform in $[0,1]$. The density of the order statistic can be found in most texts on probability theory, e.g. [Billingsley, 1986]. We get

$$f(u_n) = \frac{N!}{(n-1)!(N-n)!}u_n^{n-1}(1-u_n)^{N-n} \tag{7.9}$$

  which is a beta distribution. The first moment can be obtained in a straightforward manner, as follows:

$$E(u_n) = \frac{n}{N+1} \tag{7.10}$$

  Substituting in (7.8), we see that the limit exists and is independent of t for all w.

Thus, all conditions of [Ljung, 1977] are satisfied, therefore we conclude that the algorithm converges w.p.1 to the stable stationary points of the following ordinary differential equation

$$\frac{dw}{dt} = -\sum_{n=1}^{N}\Big[H(x_n,w) - \frac{n}{N+1})\Big]\frac{\partial H(x_n,w)}{\partial w} \tag{7.11}$$

which are precisely the local minima of the error function

$$\mathcal{E} = \sum_{n=1}^{N}\Big[H(x_n,w) - \frac{n}{N+1}\Big]^2 \tag{7.12}$$

because the right-hand side of (7.11) is the negative gradient of the error function in (7.12)[3]. ■

Thus, we have shown that SLC essentially trains the network to map point x_n to $n/(N+1)$. By comparing with SIC, described in the previous chapter, we see that both methods possess similarities for the univariate case. In fact, as $N \to \infty$, both methods are essentially equivalent. Therefore, we will restrict the remaining analysis to SIC – smooth interpolation of the sample cumulative ($G(x)$). We will first look at the convergence to the true distribution function and then the convergence to the density.

7.2 Convergence to the Distribution Function

SIC gives an estimate of the distribution function, from which we get the density by differentiation. The distribution function is useful in its own right, so we will first look at the consistency of SIC as an estimator of the distribution function – in the limit of large N, does the estimate converge to the true distribution function?

Two well studied statistics of the sample distribution function are the Kolmogorov-Smirnov statistic ($\mathcal{D}_N$) and the Cramér–von Mises statistic ($\mathcal{C}_N$) defined as follows

$$\mathcal{D}_N = \sup_x |G_N(x) - G(x)| \tag{7.13}$$

$$\mathcal{C}_N = \int_{-\infty}^{\infty} (G_N(x) - G(x))^2 dG(x) \tag{7.14}$$

The following theorems are valid [Serfling, 1980, pg 61–64]

[3]Observing this proof, it should be clear that if the hint term had been kept in (6.4) the proof would have proceeded in exactly the same manner resulting in convergence to the error function in (6.4). If we require that the solution be monotonic, then the hint term must be zero, therefore, the function converged to must be a minimum of the remaining term which is precisely the error function (7.12). Ultimately, the same final result is obtained.

Theorem 7.2.1 *With probability 1*

$$\limsup_{N\to\infty} \mathcal{D}_N \sqrt{\frac{N}{2\log\log(N)}} = \frac{1}{2} \tag{7.15}$$

PROOF: See [Smirnov, 1944], [Chung, 1949] or [Csáki, 1968]. ■

Theorem 7.2.2 ([Finkelstein, 1971]) *With probability 1*

$$\limsup_{N\to\infty} \mathcal{C}_N \frac{N}{2\log\log(N)} = \frac{1}{\pi^2} \tag{7.16}$$

It is not hard to extend these theorems to apply to generalized sample distribution functions as we will define shortly. These theorems give the probability 1 convergence behavior. We will look at $\langle \mathcal{C}_N \rangle_D$, the expected performance in in the L_2 error criterion (MISE), as well as the corresponding quantity in the L_1 error criterion. The expectations are with respect to the data set. We will derive bounds on the estimation error for the distribution function estimator, as well as the density estimator. Let us start with the following definition.

Let $\mathcal{G}$ be the space of functions such that for every $X \in \mathcal{G}$, the following holds

- $X : \mathbf{R} \to [0, 1]$
- $X'(t)$ exists everywhere, is continuously differentiable and $X'(t) \geq 0$.
- $X(-\infty) = 0$ and $X(\infty) = 1$.

Thus, $\mathcal{G}$ is the space of distribution functions on the real line that possess continuous density functions, which is the class of functions that we will be interested in. We define a metric[4] with respect to $\mathcal{G}$ as follows

$$\| f \|_X^2 = \int_{-\infty}^{\infty} f(t)^2 X'(t) dt \tag{7.17}$$

[4] A straight-forward application of Hölder's inequality [Rudin, 1976, pg 139] verifies that the triangle inequality is satisfied. The other properties of a metric are easily checked, therefore what we defined is legitimate metric.

$\| f \|_X^2$ is the expectation of the squared value of f with respect to the distribution $X \in \mathcal{G}$. Let us name this the L_2 X-norm of f. We can analogously define the L_1 X-norm of f as

$$|f|_X = \int_{-\infty}^{\infty} |f(t)| X'(t) dt \tag{7.18}$$

Let the data set (D) be $\{x_1 \leq x_2 \leq \ldots \leq x_N\}$, and corresponding to each x_i, let the target be[5]

$$y_i = \frac{i}{N+1} \tag{7.19}$$

Definition 7.2.3 *A generalized sample distribution function, H, satisfies the following two conditions*

- $H \in \mathcal{G}$
- $H(x_i) = y_i, \ \forall i$

We will denote the set of all generalized sample distribution functions for a data set, D, by $\mathcal{H}_D$

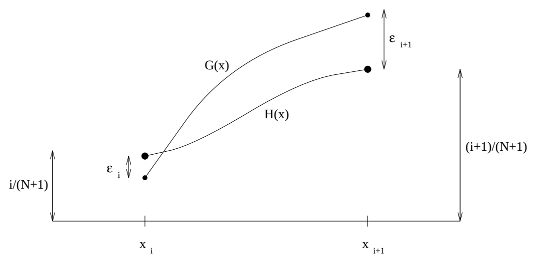
The set $\mathcal{H}_D$ contains those continuously differentiable distribution functions that interpolate the data set $\{(x_i, y_i)\}_{i=1}^N$. Naturally, this set will include a neural network, trained according to SLC or SIC after convergence to zero error. Note that the conventional distribution function estimator (6.7) is not in this class of generalized sample distribution functions, but it is the limit of functions that are in this class.

Let us now derive the estimation error for the distribution function estimator. Let $H \in \mathcal{H}_D$, and let $G \in \mathcal{G}$ be the true distribution function. As we mentioned, H estimates the whole distribution function, given a number N of observed values by interpolation. Therefore $H(x_i) = y_i = i/(N+1)$. The true distribution function

[5]For analytical convenience, this is the preferred target. In the previous chapter where SIC was introduced, the targets were $(i-1)/N$ in keeping with the traditional definition of the sample distribution function. Asymptotically, the two are equivalent and the convergence rate analysis could be done with either.

evaluated at x_i is generally not equal to y_i. Therefore we will write

$$G(x_i) = H(x_i) + \epsilon_i \tag{7.20}$$



There are two sources of estimation error: the error at each data point, ϵ_i, and the interpolation error between the data points – an error that would persist even if the ϵ_i's were all zero. As a first step, let us analyze the statistics of the ϵ_i's. $G(x)$ has a uniform distribution in [0,1] as has already been discussed. Therefore, $u_i = G(x_i)$ is the i^{th} order statistic of the uniform distribution. Thus, the density of (u_i, u_j) can be obtained by standard techniques (eg. [Billingsley, 1986, pg 200]), and is given by

$$g(u_i, u_j) = \frac{N!}{(i-1)!(j-i-1)!(N-j)!} u_i^{i-1}(1-u_j)^{N-j}(u_j-u_i)^{j-i-1} \quad j \geq i \tag{7.21}$$

Noticing that $\epsilon_i = u_i - i/(N+1)$, we can calculate the first two moments of the ϵ_i's as

$$\langle \epsilon_i \rangle = 0 \qquad \langle \epsilon_i \epsilon_j \rangle = \frac{i(N+1-j)}{(N+1)^2(N+1)} \tag{7.22}$$

The following theorem provides a bound for the estimation error, which will prove consistency of the distribution estimator in expectation.

Theorem 7.2.4 (L_2 convergence to the true distribution) *Let the data set D consist of N data points, x_i drawn i.i.d. from the (true) distribution $G \in \mathcal{G}$. For* *every $H \in \mathcal{H}_D$ and every $F \in \mathcal{G}$, the inequality*

$$\langle \| H - G \|_F^2 \rangle_D = \left\langle \int_{-\infty}^{\infty} (H(x) - G(x))^2 F'(x) dx \right\rangle_D \leq \frac{1}{2(N+1)} + \frac{3}{2(N+1)^2} \quad (7.23)$$

holds.

PROOF: Let $x_0 = -\infty$, $x_{N+1} = \infty$.

$$\| H - G \|_F^2 = \int_{-\infty}^{\infty} (H(x) - G(x))^2 F'(x) dx = \sum_{i=0}^{N} \int_{x_i}^{x_{i+1}} (H(x) - G(x))^2 F'(x) dx \quad (7.24)$$

Now by definition, $G(x_i) = H(x_i) + \epsilon_i$. Let $x \in [x_i, x_{i+1}]$. Because both H and G are monotonically increasing we have that $i/(N+1) = H(x_i) \leq H(x) \leq H(x_{i+1}) = (i+1)/(N+1)$ and $i/(N+1) + \epsilon_i = G(x_i) \leq G(x) \leq G(x_{i+1}) = (i+1)/(N+1) + \epsilon_{i+1}$. We then get

$$\epsilon_i - \frac{1}{N+1} \leq G(x) - H(x) \leq \epsilon_{i+1} + \frac{1}{N+1} \quad (7.25)$$

Thus

$$(G(x) - H(x))^2 \leq (\epsilon_i - \frac{1}{N+1})^2 + (\epsilon_{i+1} + \frac{1}{N+1})^2 \quad (7.26)$$

Therefore using (7.22)

$$\begin{aligned} \langle (G(x) - H(x))^2 \rangle_D &= \frac{2}{(N+1)^2} + \langle \epsilon_i^2 + \epsilon_{i+1}^2 \rangle + \frac{2}{N+1}(\langle \epsilon_{i+1} - \epsilon_i \rangle), \\ &= \frac{2}{(N+1)^2} + \frac{i(N+1-i) + (i+1)(N-i)}{(N+1)^2(N+2)}, \\ &\leq \frac{3}{2(N+1)^2} + \frac{1}{2(N+1)}, \quad (7.27) \end{aligned}$$

where the last inequality follows by maximizing over i. Thus we find

$$\begin{aligned} \langle \| H - G \|_F^2 \rangle_D &= \sum_{i=0}^{N} \int_{x_i}^{x_{i+1}} \langle (H(x) - G(x))^2 \rangle_{x_i} F'(x) dx, \quad (7.28) \\ &\leq \sum_{i=0}^{N} \int_{x_i}^{x_{i+1}} \left(\frac{3}{2(N+1)^2} + \frac{1}{2(N+1)} \right) F'(x) dx, \quad (7.29) \end{aligned}$$

$$= \left(\frac{3}{2(N+1)^2} + \frac{1}{2(N+1)}\right) \sum_{i=0}^{N} \int_{x_i}^{x_{i+1}} F'(x)dx, \qquad (7.30)$$

$$= \left(\frac{3}{2(N+1)^2} + \frac{1}{2(N+1)}\right) \int_{-\infty}^{\infty} F'(x)dx, \qquad (7.31)$$

$$= \frac{3}{2(N+1)^2} + \frac{1}{2(N+1)}. \qquad (7.32)$$

The last line follows because $\int_{-\infty}^{\infty} F'(x)dx = 1$. ■

Note 1: The theorem applies to any $H \in \mathcal{H}_D$ and any $F \in \mathcal{G}$, in particular to $F(x) = G(x)$, the true distribution function. Thus the expected squared error approaches zero asymptotically at a rate $O(1/N)$.

Note 2: The same proof can be modified for the case of the integrated squared error over some bounded interval.

Note 3: Without much extra effort, this result can be extended to a larger class $\mathcal{H}_D^*$ of interpolators that are defined by $H \in \mathcal{H}_D^*$ if

$$x \in [x_i, x_{i+1}] \quad \Rightarrow \quad H(x) \in [\frac{i}{N+1}, \frac{i+1}{N+1}] \qquad (7.33)$$

with no other constraints being made on continuity, the existence of derivatives or even monotonicity. To see this notice that the monotonicity of H is used only in obtaining (7.25), where (7.33) can be used in its place. Thus, it is not imperative for the distribution function estimator to be monotonic to get convergence to the true distribution. In fact this will carry over even for the density estimate convergence. Therefore, the monotonicity hints used in the previous chapter are not crucial for convergence. We used them in our applications and required monotonicity in our definition of $\mathcal{G}$ simply to ensure that we obtained legitimate density functions. Note that the conventional distribution function (6.7) belongs to this extended class $\mathcal{H}_D^*$.

Note 4: Exact interpolation is not required. The result ($O(1/N)$ convergence) can be obtained without much added effort as long as $|H(x_i) - i/(N+1)| \leq A/\sqrt{N}$, $\forall i$.

To see this we can look at (7.25) and note that this approximate interpolation would only add terms that are $O(1/N)$ throughout the rest of the proof.

Note 5: The theorem applies to *any* interpolator satisfying (7.33). In particular a large enough neural network will be one such monotonic interpolator, provided that the network can be trained to zero error, and hence the target outputs ($y_i = i/(N+1)$) can be realized by the network exactly. For a large enough neural network, this can be accomplished [Atiya, 1994]. Further, because we only need to approximate a generalized distribution function to within $O(1/\sqrt{N})$, one can apply the universal approximation results for multilayer networks ([Hornik *et al.*, 1989], [Cybenko, 1989], [Funahashi, 1989], [Barron, 1993]) to show that as long as the number of hidden units $> N$, this interpolation to $O(1/N)$ can be accomplished. To apply these theorems, certain technical restrictions need to be made on the true distribution function, which we will assume are satisfied[6].

The previous theorem can easily be extended to obtain the L_1 convergence.

Theorem 7.2.5 (L_1 convergence to the true distribution) *Let the data set D consist of N data points, x_i drawn i.i.d. from the (true) distribution $G \in \mathcal{G}$. For every $H \in \mathcal{H}_D$ and every $F \in \mathcal{G}$, the inequality*

$$\langle |H - G|_F \rangle_D = \left\langle \int_{-\infty}^{\infty} |H(x) - G(x)| F'(x) dx \right\rangle_D \leq \frac{1}{\sqrt{N+2}} + \frac{2}{N+1} \tag{7.34}$$

holds.

PROOF: The same reasoning that led up to (7.25) is still valid. Thus, we find for $x \in [x_i, x_{i+1}]$

$$|H(x) - G(x)| \leq |\epsilon_i| + |\epsilon_{i+1}| + \frac{2}{N+1} \tag{7.35}$$

[6]Essentially one needs boundedness of sufficiently high order derivatives.

By Jensen's inequality, $\langle|\epsilon_i|\rangle \leq \sqrt{\langle\epsilon_i^2\rangle}$ therefore we find that $\langle|\epsilon_i|\rangle \leq \frac{1}{2}\sqrt{1/(N+2)}$. The rest of the analysis is exactly analogous to the L_2 convergence proof, the final result being (7.34). ■

Thus we have L_1 convergence at a rate $O(1/\sqrt{N})$.

Note 6: The bounds on the estimation error derived in this section are not asymptotic, and hold for every N.

7.3 Convergence to the True Density Function

We have seen in the last section that SIC provides a consistent estimator for the distribution function. In addition, we obtained bounds on the estimation error. In this section, we derive the estimation error for the density estimator. As we have already discussed, the density estimator is simply the derivative of the distribution function estimator (i.e. the neural network output), and thus we need to evaluate $\left\langle \| H' - G' \|_F^2 \right\rangle_{x_i}$.

Obtaining a tight convergence rate for the density estimation error is a tougher job. The reason is that the derivative operation accentuates the noise. Another reason is that, as we discussed in chapter 5, the density estimation problem is an ill-posed problem. Without any *a priori* assumptions on the true density function, it will be hard to judge the suitability of an estimator. Even the trivial estimator consisting of the summation of delta functions centered at the data points could be as valid as other estimators. In fact, [Barron *et al.*, 1992] and [Devroye and Györfi, 1990] show that no density estimator is consistent for certain types of error measure unless one makes some *a priori* assumptions about the distribution function. Typical *a priori* assumptions used in the density estimation literature are smoothness constraints on the class of considered densities. These are realistic assumptions that are usually obeyed by densities that are typically encountered in real world applications. We will consider such constraints in our analysis. First, we need the following results relating the higher derivatives of a function to lower order derivatives.

Lemma 7.3.1 *Let a twice differentiable function* $f(x)$ *be defined on the interval* (a, ∞) *where the lower limit (a) could be* $-\infty$*. Suppose* $\sup_x |f| = M_0$*,* $\sup_x |f'| = M_1$*, and* $\sup_x |f''| = M_2$*, where the* $\sup_x$ *is taken over* $x \in (a, \infty)$*. Then,*

$$M_1^2 \leq 4M_0M_2 \tag{7.36}$$

Proof: Let $h > 0$. Then for $x \in (a, \infty)$, by Taylor's theorem, we have

$$f(x + 2h) = f(x) + 2hf'(x) + (2h)^2\frac{f''(\zeta)}{2} \tag{7.37}$$

for some $\zeta \in (x, x + 2h)$. Therefore

$$f'(x) = \frac{f(x + 2h) - f(x)}{2h} - hf''(\zeta) \tag{7.38}$$

so

$$|f'(x)| \leq \frac{M_0}{h} + hM_2 \tag{7.39}$$

This inequality holds for every $h > 0$. In particular for the h that minimizes the right hand side. This minimum is attained at $h = \sqrt{M_0/M_2}$ and so we have the bound

$$|f'(x)| \leq 2\sqrt{M_0M_2} \tag{7.40}$$

Thus $\sup_x |f'(x)| \leq 2\sqrt{M_0M_2}$. ■

Corollary 7.3.2 *Let* $G(x)$ *be differentiable* m *times on the interval* (a, ∞)*. Let* $\sup_x \left|G^{(i)}\right| = M_i$ *for* $i = 1 \ldots m$*.* $G^{(i)}$ *denotes the* i^{th} *derivative. Then*

$$M_i^2 \leq 4M_{i-1}M_{i+1} \tag{7.41}$$

for $i = 1 \ldots (m - 1)$*.*

Proof: Let $f(x)$ in lemma 7.3.1 be $G^{(i-1)}(x)$. ■

Theorem 7.3.3 *Let $G(x)$ be differentiable m times on the interval (a, ∞). Let $\sup_x \left|G^{(i)}\right| = M_i$ for $i = 1 \ldots m$. Then*

$$M_1 \le 2^{m-1} M_0^{1-\frac{1}{m}} M_m^{\frac{1}{m}} \tag{7.42}$$

Proof: We prove the theorem by induction on m. For $m = 2$, the theorem is equivalent to lemma 7.3.1. Suppose (7.42) holds for $m = P$. We show that it holds for $m = P + 1$. From lemma 7.3.1 we have

$$M_1 \le 2 M_0^{\frac{1}{2}} M_2^{\frac{1}{2}} \tag{7.43}$$

By applying the induction hypothesis to G' we have the following equation

$$M_2 \le 2^{P-1} M_1^{1-\frac{1}{P}} M_{P+1}^{\frac{1}{P}} \tag{7.44}$$

Combining these two equations we get

$$\begin{aligned} M_1 &\le 2 M_0^{\frac{1}{2}} \left(2^{P-1} M_1^{1-\frac{1}{P}} M_{P+1}^{\frac{1}{P}}\right)^{\frac{1}{2}}, && (7.45) \\ &= 2^{\frac{P+1}{2}} M_0^{\frac{1}{2}} M_1^{\frac{P-1}{2P}} M_{P+1}^{\frac{1}{2P}}. && (7.46) \end{aligned}$$

Thus we find that

$$\begin{aligned} M_1 &\le \left(2^{\frac{P+1}{2}} M_0^{\frac{1}{2}} M_{P+1}^{\frac{1}{2P}}\right)^{\frac{2P}{P+1}}, && (7.47) \\ &= 2^P M_0^{1-\frac{1}{P+1}} M_{P+1}^{\frac{1}{P+1}}, && (7.48) \end{aligned}$$

and thus (7.42) holds for $m = P + 1$. This concludes the proof. ∎

Corollary 7.3.4 *Let $G(x) \in \mathcal{C}^\infty$ on the interval (a, ∞). Let $\sup_x \left|G^{(i)}\right| = M_i$. Define the sequence $\{s_k\}_{k=0}^{\infty}$ by $s_k = 2^{k-1} M_k^{1/k}$. Suppose that $\lim_{k\to\infty} s_k = \beta < \infty$. Then*

$$M_1 \le \beta M_0 \tag{7.49}$$

PROOF: Taking the limit $m \to \infty$ in Theorem 7.3.3 and using the fact that the term by term product of two convergent sequences converges to the product of the limits, the corollary follows. ■

After these preliminary results, we now come to our density estimator. We will assume that the true distribution function has bounded derivatives up to order K. Then, in the asymptotic limit $N \to \infty$, one expects that with probability 1, an interpolator should exist with bounded derivatives with the same bounds, because in the asymptotic limit, $G(x)$ itself becomes an interpolator. To proceed in a more formal manner, we will extend $\mathcal{H}_D$ to include the approximate interpolators. Thus we define the set of approximate sample distribution functions $\mathcal{H}_D^\nu$ as follows

Definition 7.3.5 *Fix $\nu > 0$. A <u>ν-approximate sample distribution function</u>, H, satisfies the following two conditions*

- $H \in \mathcal{G}$
- $\left|H(x_i) - \frac{i}{N+1}\right| \leq \nu\sqrt{\frac{\log\log(N)}{2N}},\ \forall i$

We will denote the set of all ν-approximate sample distribution functions for a data set, D, and a given ν by $\mathcal{H}_D^v$.

Note that $\mathcal{H}_D \subset \mathcal{H}_D^v$ for every $\nu > 0$, and $\mathcal{H}_D = \mathcal{H}_D^0$. Let $A_i = \sup_x \left|G^{(i)}\right|,\ i = 1 \ldots K$ and define $B_i^\nu(D)$ by

$$B_i^\nu(D) = \inf_{H \in \mathcal{H}_D^v} \sup_x \left|H^i\right| \tag{7.50}$$

for fixed $\nu > 0$. Note that by definition, for all $\epsilon > 0$, $\exists\ H \in \mathcal{H}_D^v$ such that $\sup_x \left|H^{(i)}(x)\right| \leq B_i^\nu + \epsilon$. $B_i^\nu(D)$ is the lowest possible bound on the i^{th} derivative for the ν-approximate sample distribution functions given a particular data set. In a sense, the "smoothest" approximating sample distribution function with respect to the i^th derivative has an i^{th} derivative bounded by $B_i^\nu(D)$ One expects that $B_i \leq A_i$, at least in the limit $N \to \infty$. This is the content of the next theorem. Let $B_i^\nu = A_i + \eta_i^\nu(D)$.

Theorem 7.3.6 *For every $\nu > 1$, with probability 1,*

$$\eta_i^\nu(D) \overset{N\to\infty}{\leq} 0 \tag{7.51}$$

By this is meant that for all $\delta > 0$

$$\lim_{N\to\infty} P[\eta_i^\nu(D) < \delta] = 1 \tag{7.52}$$

PROOF: This is a straightforward application of theorem 7.2.1 as follows. Suppose the theorem were false. Then there exists $\delta > 0$ such that $\lim_{N\to\infty} P[\eta_i^\nu(D) > \delta] \geq \gamma > 0$. Therefore with probability$\geq \gamma$, $G \notin \mathcal{H}_D^v$ as $N \to \infty$, as if $G \in \mathcal{H}_D^v$ then $\eta \leq 0$. Thus with probability $\geq \gamma$,

$$\mathcal{D}_N = \sup_x |G_N(x) - G(x)| > \nu\sqrt{\frac{\log\log(N)}{2N}} \tag{7.53}$$

as $N \to \infty$. Thus,

$$\lim_{N\to\infty} D_N\sqrt{\frac{N}{2\log\log(N)}} \geq \lim_{N\to\infty} \frac{\nu}{2} > \frac{1}{2}$$

with probability$\geq \gamma > 0$, implying that

$$\limsup_{N\to\infty} D_N\sqrt{\frac{N}{2\log\log(N)}} > \frac{1}{2} \tag{7.54}$$

with probability$\geq \gamma > 0$, contradicting theorem 7.2.1. ■

In the next theorem, we present the main result of the chapter, namely a bound on the estimation error for the density estimator obtained by using the approximate sample distribution functions. It is embedded in a large amount of technical machinery, but its essential content is that if the true distribution function has bounded derivatives to order K then picking the approximate distribution function with "minimum" value for B^K, we obtain a convergence rate for the estimation error of $O((\log\log(N)/N)^{1-1/K})$.

Theorem 7.3.7 (L_2 convergence to the true density) *Let N data points, x_i be drawn i.i.d. from the distribution $G \in \mathcal{G}$. Let $\sup_x \left|G^{(i)}\right| = A_i$ for $i = 0 \ldots K$, where $K \geq 2$. Fix $\nu > 1$ and $\epsilon > 0$. Let $B_K^\nu(D) = \inf_{H \in \mathcal{H}_D^\nu} \sup_x \left|H^K\right|$. Let $H \in \mathcal{H}_D^\nu$ be a ν-approximate distribution function with $B_K = \sup_x \left|H^K\right| \leq B_K^\nu + \epsilon$ (by the definition of B_K^ν, such a ν-approximate sample distribution function must exist). Then, for any $F \in \mathcal{G}$, as $N \to \infty$, the inequality*

$$\left\langle \| H' - G' \|_F^2 \right\rangle_D \leq 2^{2(K-1)} (2A_K + \epsilon)^{\frac{2}{K}} \mathcal{F}(N) \tag{7.55}$$

where

$$\mathcal{F}(N) = \left[(1+\nu) \left(\frac{2 \log \log(N)}{N} \right)^{\frac{1}{2}} + \frac{2}{N+1} \right]^{2 - \frac{2}{K}} \tag{7.56}$$

holds with probability 1, as $N \to \infty$.

Proof: Let $S(x) = G(x) - H(x)$. Then $M_K = \sup_x \left|S^{(K)}\right| \leq A_K + B_K$, and

$$\begin{aligned} \| H' - G' \|_F^2 &= \int_{-\infty}^{\infty} S'(x)^2 F'(x) dx && (7.57) \\ &\leq \sup_x S'(x)^2 && (7.58) \end{aligned}$$

By Theorem 7.3.3

$$S'(x)^2 \leq 2^{2(K-1)} (A_K + B_K)^{\frac{2}{K}} \underbrace{\sup_x S(x)^{2 - \frac{2}{K}}}_{I}. \tag{7.59}$$

where $B_K = \sup_x \left|H^{(K)}\right|$. By construction, $B_K \leq B_K^\nu(D) + \epsilon$. By theorem 7.3.6, as $N \to \infty$, $B_K^\nu(D) \leq A_K + \epsilon$ with probability 1, so,

$$(A_K + B_K)^{\frac{2}{K}} \leq (2A_K + \epsilon)^{\frac{2}{K}} \tag{7.60}$$

holds with probability 1 in the asymptotic limit. We are interested in the expected value of (7.59) with respect to D. Therefore, we need the expected value of I,

$$\langle I \rangle = \left\langle (\sup_x |S(x)|)^{2-\frac{2}{K}} \right\rangle_{\epsilon_i} \tag{7.61}$$

Jensen's inequality gives for $0 < a < 1$ that $\langle |\phi|^a \rangle \leq \langle |\phi| \rangle^a$. Applying this to $\phi = S^2$ yields

$$\langle I \rangle \leq \left\langle \sup_x S^2(x) \right\rangle_{\epsilon_i}^{1-\frac{1}{K}} \tag{7.62}$$

Because H is a ν-approximate sample distribution function, $H(x_i) = i/(N+1) + \eta_i$ where $|\eta_i| \leq \nu\sqrt{\log\log(N)/2N}$. Similar reasoning to that which led to (7.25) can now be applied to show that for $x \in [x_i, x_{i+1}]$,

$$\epsilon_i - \eta_{i+1} - \frac{1}{N+1} \leq G(x) - H(x) \leq \epsilon_{i+1} - \eta_i + \frac{1}{N+1} \tag{7.63}$$

therefore

$$|S(x)| \leq |\epsilon_i| + |\epsilon_{i+1}| + \nu \left(\frac{2\log\log(N)}{N} \right)^{\frac{1}{2}} + \frac{2}{N+1} \tag{7.64}$$

and using theorem 7.2.1, we see that with probability 1, as $N \to \infty$,

$$\sup_x S^2(x) \leq \underbrace{\left[(1+\nu) \left(\frac{2\log\log(N)}{N} \right)^{\frac{1}{2}} + \frac{2}{N+1} \right]^2}_{\mathcal{F}(N)} \tag{7.65}$$

This inequality holds with probability 1 in the asymptotic limit, therefore, its expectation, $\langle I \rangle_D$, is also bounded by the same quantity. To see this, suppose that $\lim_{N\to\infty} \langle I \rangle_D > \mathcal{F}(N)$, then with some probability $\epsilon > 0$, $I > \mathcal{F}(N)$, otherwise $\langle I \rangle_D \leq \mathcal{F}(N)$. Therefore $\sup_x S^2(x) > \mathcal{F}(N)$ with probability $\epsilon > 0$ in the asymptotic limit, contradicting (7.65) . Using (7.60) in (7.59), and the fact that $\langle I \rangle$ is bounded by (7.65), the theorem is proved. ∎

Notes 1–5 at the end of theorem 7.2.4 apply here as well. Note 5 will still apply because a neural network can approximate a function and its derivative [Hornik *et al.*, 1990].

Note 7: This theorem holds for any $\epsilon > 0$ and $\nu > 1$. Therefore we see that for smooth density functions, with bounded higher derivatives, the convergence rate approaches $O(\log\log(N)/N)$ which is faster than $1/N^{1-\epsilon}$ convergence for any $\epsilon > 0$. Thus, we have faster convergence than the standard kernel density estimator (for which the rate is $O(N^{-4/5})$).

Note 8: No smoothing parameter needs to be determined unlike the other traditional techniques.

Note 9: From the theorem, it is clear that one should try to find a generalized ν-approximate distribution function with the smallest possible derivatives. Specifically, of all the sample distribution functions, pick the one that "minimizes" B_K, the bound on the K^{th} derivative. Thus, when using optimization techniques to find the generalized distribution function, one would be justified in introducing penalty terms, penalizing the magnitudes of the derivatives (for example Tikhonov type regularizers [Tikhonov and Arsenin, 1977]).

L_1 convergence follows by using very similar arguments to the previous theorem.

Theorem 7.3.8 (L_1 convergence to the true density) *Let N data points, x_i be drawn i.i.d. from the distribution $G \in \mathcal{G}$. Let $\sup_x \left|G^{(i)}\right| = A_i$ for $i = 0 \ldots K$, where $K \geq 2$. Fix $\nu > 1$ and $\epsilon > 0$. Let $B_K^\nu(D) = \inf_{H \in \mathcal{H}_D^\nu} \sup_x \left|H^K\right|$. Let $H \in \mathcal{H}_D^\nu$ be a ν-approximate distribution function with $B_K = \sup_x \left|H^K\right| \leq B_K^\nu + \epsilon$ (by the definition of B_K^ν, such a ν-approximate sample distribution function must exist). Then, for any $F \in \mathcal{G}$, as $N \to \infty$, the inequality*

$$\left\langle \| H' - G' \|_F^2 \right\rangle_D \leq 2^{(K-1)}(2A_K + \epsilon)^{\frac{1}{K}} \mathcal{F}(N) \tag{7.66}$$

where

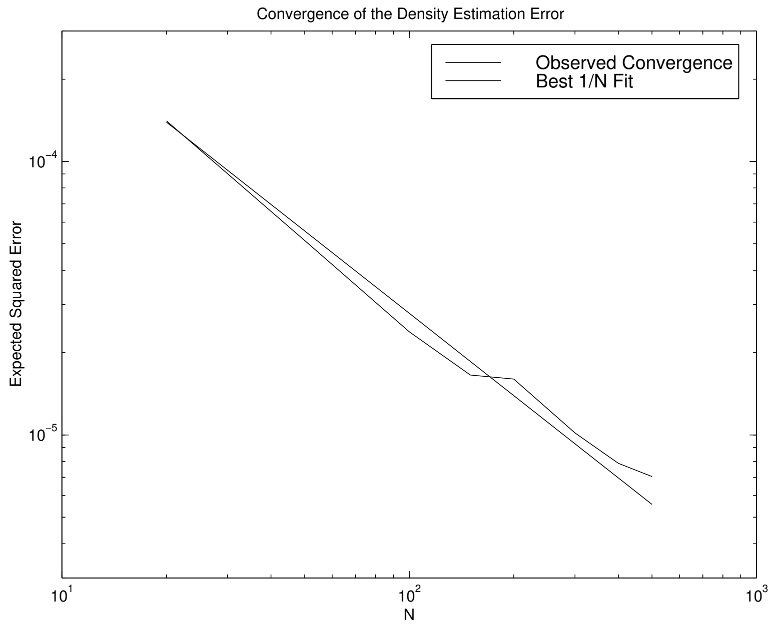
$$\mathcal{F}(N) = \left[(1+\nu)\left(\frac{2\log\log(N)}{N}\right)^{\frac{1}{2}} + \frac{2}{N+1}\right]^{1-\frac{1}{K}} \tag{7.67}$$

holds with probability 1.

PROOF: The proof proceeds in an analogous fashion to the proof of theorem 7.3.7 with $\sup_y S^2(y)$ replaced by $\sup_y |S(y)|$. ∎

Once again, the theorem applies to any $\epsilon > 0$ and any $\nu > 1$, therefore, for smooth density functions, we have L_1 convergence at a rate $O((\log\log N/N)^{\frac{1}{2}})$, which is faster than $O(1/N^{\frac{1}{2}-\epsilon})$ for any $\epsilon > 0$.

7.4 Simulation Results



A five hidden unit two layer neural network was used to perform the mapping $x_i \to i/(N+1)$, trained according to SIC. For various$N$, the resulting density estimation error was computed for over 100 runs for each N. Plotted are the results on a Log-Log scale. For comparison, also shown is the best $1/N$ fit.

Figure 7.1: Convergence of the density estimation error for SICI

We demonstrate the convergence results for SIC with a simulation run on the same test density used in section 6.4. A five hidden unit neural network was trained using SIC to estimate the sample distribution function, for which the density estimate is the derivative of the learned function. The density estimation error was then computed.

Shown in figure 7.1 is the behavior of $\log(E)$ versus $\log(N)$, and for comparison, we show a slope -1 curve. The optimal linear fit had slope -0.97.

7.5 Convergence of SLCI and SICI

In this section we will briefly analyze the random variate generation methods to show that they generate the true distribution as $N \to \infty$.

7.5.1 Convergence of SLCI to SICI

We will use ideas very similar to those used in section 7.1. The update rule for SLCI is

$$w(t) = w(t+1) - \eta(t) \cdot \underbrace{2\sum_{n=1}^{N}(H(u_n(t), w(t)) - x_n)\frac{\partial H(u_n(t), w(t))}{\partial w}}_{Q(w(t),u(t))} \tag{7.68}$$

Just as in section 7.1, $Q(w(t), u(t))$ satisfies requirements $A1$, $A2'$, $A3$ and $A4$ of [Ljung, 1977]. Therefore, convergence to a stable stationary point of the differential equation

$$\frac{dw}{dt} = \lim_{t\to\infty} E[2\sum_{n=1}^{N}(H(u_n(t), w(t)) - x_n)\frac{\partial H(u_n(t), w(t))}{\partial w}] \tag{7.69}$$

occurs with probability 1, provided $\eta(t)$ satisfies the required conditions. Thus, we have proved the following theorem:

Theorem 7.5.1 *SLCI converges with probability 1 to a local minimum of the error function*

$$\mathcal{E} = E\left[\sum_{n=1}^{N}\left[H(u_n, w) - x_n\right]^2\right], \tag{7.70}$$

(where the expectation is with respect $u_1, \ldots, u_N$, the N order statistics of a uniform distribution), provided that the learning rate $\eta(t)$ is a decreasing sequence, that satisfies the following conditions

a. $\sum_{t=1}^{\infty}\eta(t) = \infty$,

b. $\sum_{t=1}^{\infty} \eta^p(t) < \infty$, for some p,

c. $\lim_{t\to\infty} \sup[1/\eta(t) - 1/\eta(t-1)] < \infty$.

We are now almost there, because in the limit $N \to \infty$, we expect $u_n \to n/(N+1)$ with probability 1. This, however, is just an application of theorem 7.2.1. Another way to see this is to write $u_i = i/N+1+\epsilon_i$ in (7.70) and expand as a power series in ϵ_i. The ϵ_i terms disappear and the higher order expectations decay to zero as $N \to \infty$. Therefore, we have the following theorem.

Theorem 7.5.2 *If the conditions of theorem 7.5.1 hold, then in the limit $N \to \infty$, SLCI converges with probability 1 to a minimum of the error function*

$$\mathcal{E} = \sum_{n=1}^{N} \left[H\left(\frac{n}{N+1}, w\right) - x_n \right]^2, \tag{7.71}$$

Thus we see that SLCI converges to a solution of SICI (in the limit $N \to \infty$) with probability 1. We now look at the convergence of SICI

7.5.2 Convergence of SICI to G^{-1}

We will make the simplifying assumption that $G^{-1}(u)$ is continuous on $[0, 1]$. Therefore the input support is compact. Since $G(x)$ was assumed differentiable, then so is $G^{-1}(u)$, and further because $[0, 1]$ is compact, we have that $|dG^{-1}(u)/du|$ is bounded. Let the bound on the derivative be S. Then, the input distribution is bounded in the range $[-S, S]$.

Let $u_i = i/(N+1)$, then SICI trains the network to implement the inverse of a generalized distribution function that interpolates the points $\{u_i, x_i\}$. Let the inverse of this generalized distribution function be $H(u)$. We would like to analyze the rate of convergence of the integrated squared error $\mathcal{E} = E\left[\int_0^1 d\mu(u)\ (H(u) - G^{-1}(u))^2\right]$, where μ is any continuous measure on $[0, 1]$. Write

$$x_i = G^{-1}(u_i) + \epsilon_i \tag{7.72}$$

Because both G^{-1} and H are monotonic, the same arguments that led up to (7.25) can be used to get that for $u \in [u_i, u_{i+1}]$,

$$x_i - G^{-1}(u_{i+1}) \leq H(u) - G^{-1}(u) \leq x_{i+1} - G^{-1}(u_i) \tag{7.73}$$

from which follows the inequality

$$(H(u) - G^{-1}(u))^2 \leq (x_i - G^{-1}(u_{i+1}))^2 + (x_{i+1} - G^{-1}(u_i))^2 \tag{7.74}$$

The density for the order statistic x_i is [Billingsley, 1986, pg 200]

$$\tilde{g}(x_i) = \frac{N!}{(i-1)!(N-i)!} g(x_i) G(x_i)^{i-1} (1 - G(x_i))^{N-i} \tag{7.75}$$

We will require the expected value $E[(x_i - G^{-1}(u_{i+1}))^2]$ and $E[(x_{i+1} - G^{-1}(u_i))^2]$. They can be bounded as follows

$$\begin{aligned}
E[(x_i - G^{-1}(u_{i+1}))^2] &= \int_{-\infty}^{\infty} dx_i \; (x_i - G^{-1}(u_{i+1}))^2 \tilde{g}(x_i) && (7.76)\\
&\overset{(a)}{=} \int_0^1 du \; \beta_{i,N+1-i}(u) (G^{-1}(u) - G^{-1}(u_{i+1}))^2 && (7.77)\\
&\overset{(b)}{=} \int_0^1 du \; \beta_{i,N+1-i}(u) \frac{dG^{-1}(\zeta)}{du} (u - u_{i+1})^2 && (7.78)\\
&\leq S^2 E[(u - u_{i+1})^2]_{\beta_{i,N+1-i}} && (7.79)
\end{aligned}$$

where

$$\beta_{\alpha,\beta}(u) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} u^{\alpha-1} (1-u)^{\beta-1} \tag{7.80}$$

(a) follows by using (7.75) and making a transformation of variables to $u = G(x_i)$, and (b) by Taylor's theorem, for some $\zeta(u)$. A similar calculation yields $E[(x_{i+1} - G^{-1}(u_i))^2] \leq S^2 E[(u - u_i)^2]_{\beta_{i+1,N-i}}$ The expectations are with respect to the relevent β-distributions and can be calculated using standard methods. Then, straightforward bounds for these expectations gives that for $u \in [1/(N+1), N/(N+1)]$,

$$E\left[(H(u) - G^{-1}(u))^2\right] \leq S^2 \frac{N-1}{4(N+1)^2} \tag{7.81}$$

In the two end cells $[0, 1/(N+1)]$ and $[N/(N+1), 1]$, the difference between H and G^{-1} is bounded by $2S$. Putting all this together, we find

$$\begin{aligned} E\left[\int_0^1 d\mu(u)\ (H(u) - G^{-1}(u))^2\right] &= \sum_{i=0}^{N} \int_{\frac{i}{N+1}}^{\frac{i+1}{N+1}} d\mu(u)\ E\left[(H(u) - G^{-1}(u))^2\right] \\ &\leq 4S^2\mu\left(\left[0, \frac{1}{N+1}\right] \bigcup \left[\frac{N}{N+1}, 1\right]\right) + \\ &\quad S^2 \frac{N-1}{4(N+1)^2} \int_{\frac{1}{N+1}}^{\frac{N}{N+1}} d\mu(u) \\ &\leq S^2 \left(\frac{8B}{N+1} + \frac{N-1}{4(N+1)^2}\right) \end{aligned} \tag{7.82}$$

where the last inequality follows because $d\mu(u)$ was assumed to be a continuous measure and therefore can be bounded on this compact set (by B). Thus we have proved the following theorem

Theorem 7.5.3 (L_2 convergence of SICI) *Let the data set D consist of N data points, x_i drawn i.i.d. from the (true) distribution $G \in \mathcal{G}$. Assume that G^{-1} is continuous on the closed interval $[0, 1]$ and let its derivative be bounded by S. For every $H \in \mathcal{H}_D^*$ that is zero outside the interval $[-S, S]$ and any continuous measure $d\mu(u)$ on $[0, 1]$, the inequality*

$$\left\langle \| H^{-1} - G^{-1} \|_\mu^2 \right\rangle_D \leq S^2 \left(\frac{8B}{N+1} + \frac{N-1}{4(N+1)^2}\right) \tag{7.83}$$

holds. ■

Notes 1–5 appended to theorem 7.2.4 apply. Thus we have proved that SICI converges to G^{-1} provided the conditions of theorem 7.5.3 hold, and further that the rate of convergence is $O(1/N)$. Just as the L_1 convergence of SIC and its corresponding density estimate were obtained, we can get L_1 convergence of SICI.

Theorem 7.5.4 (L_1 convergence of SICI) *Let the data set D consist of N data points, x_i drawn i.i.d. from the (true) distribution $G \in \mathcal{G}$. Assume that G^{-1} is continuous on the closed interval $[0,1]$ and let its derivative be bounded by S. For every $H \in \mathcal{H}_D^*$ that is zero outside the interval $[-S, S]$ and any continuous measure $d\mu(u)$ on $[0,1]$, the inequality*

$$\left\langle \int_0^1 d\mu(u) \left| H^{-1} - G^{-1} \right| \right\rangle_D \leq S \left(\frac{4B}{N+1} + \frac{\sqrt{N-1}}{2(N+1)} \right) \tag{7.84}$$

holds.

The proof is analogous to the L_2 convergence, with an application of Jensen's inequality to bound $E[|u - u_i|]$ by $\sqrt{E[(u-u_i)^2]}$.

7.6 Comments

Our main goal in this chapter was to provide convergence results applicable to the use of generalized distribution functions. Thus we have laid a theoretical foundation upon which the methods of the previous chapter stand. The conditions that we required of the true distribution function are not unreasonable from the practical point of view. We see that the convergence rate for the density estimation approaches $O(\log\log(N)/N)$. A similar result holds in L_1. Our convergence results do not require any restrictions to be placed on the support of the true distribution.

Comparable rates of convergence are obtained only for parametric density estimation, where the true density has the same form as the estimating density and it is only the parameters that need to be estimate. For example, [Scott, 1988, pg 19-40] shows that for estimating a Gaussian density and the parameter of a uniform density, O(1/N) convergence is obtained. The only case we find in the literature that obtains faster than $O(1/N)$ convergence is for the special case of the uniform density where $O(1/N^2)$ can be obtained using an estimator of the form cx_N where x_N is the n^{th}

order statistic and $c \approx 1 + \log 2/N$. The optimal rate for the integrated squared error for density functions with bounded r-Sobolev norm is $O(N^{-(1-1/(2r+1))})$, which also approaches N^{-1} for smooth density functions[7], [Bretagnolle and Huber, 1979] and [Efroimovich and Pinsker, 1983]. [Barron and Sheu, 1991] has obtained a similar rate of convergence for a cross entropy error function. Our rate has a similar form to this optimal rate for bounded Sobolev norm – note that there is no easy connection between bounded Sobolev norm and bounded derivatives in general.

Because lemma 7.3.1 can be saturated [Rudin, 1976, pg 114], we conjecture that the convergence rate is optimal in the sense that for a true density with bounded derivatives $A_1, \ldots, A_K$,

$$\sup_{\epsilon}\{\epsilon : \mathcal{E} = O(N^{-\epsilon})\} = 1 - \frac{1}{K} \tag{7.85}$$

where this *sup* may or may not be attainable.

[7] r is the maximum degree for which $\int |f^{(r)}| < \infty$.

Part III

A Natural Prior

Introduction

— *A mathematician who assumes nothing also proves nothing.*

The application of Bayesian inference assumes the knowledge of a prior. Prior information can be used to greatly improve upon learning performance, if the *assumed* prior information is correct. Effectively, the search space becomes narrower without sacrificing any approximation capability. Thus, one expects the BIAS to remain constant and the VARIANCE to decrease [Geman and Bienenstock, 1992]. The capacity of the learning model will generally have decreased [Sill, 1998a], and possibly the VC dimension as well [Abu-Mostafa, 1993]. One therefore expects the generalization performance to be closer to the in-sample performance, and since we have not sacrificed any in-sample performance, one expects the generalization performance to be better.

One technique for incorporating prior information about the target function into the learning process is through the use of hints [Abu-Mostafa, 1995]. Hints can be enforced "softly" by using virtual examples (extra examples generated using (for example) a known invariance of the target function) or by introducing a penalty function that is minimized when the hint is obeyed. These techniques have been applied with success in the financial markets (ex. [Abu-Mostafa and Atiya, 1996]) and to medical, bond, and credit data by [Sill and Abu-Mostafa, 1997]. Another way to enforce hints is to have a learning model that contains only functions that obey the hint (ex.[Sill, 1998b]). [Sill and Abu-Mostafa, 1997] shows how a "softly" enforced penalty hint can be interpreted in a Bayesian formalism by assigning a probability distribution to the hint violation. Incorporating prior information into the learning process will improve the performance. On the other hand, if the *assumed* prior information is incorrect, it could be potentially disastrous.

- Is it necessary to incorporate some prior information into the learning process if one is to get anywhere at all?

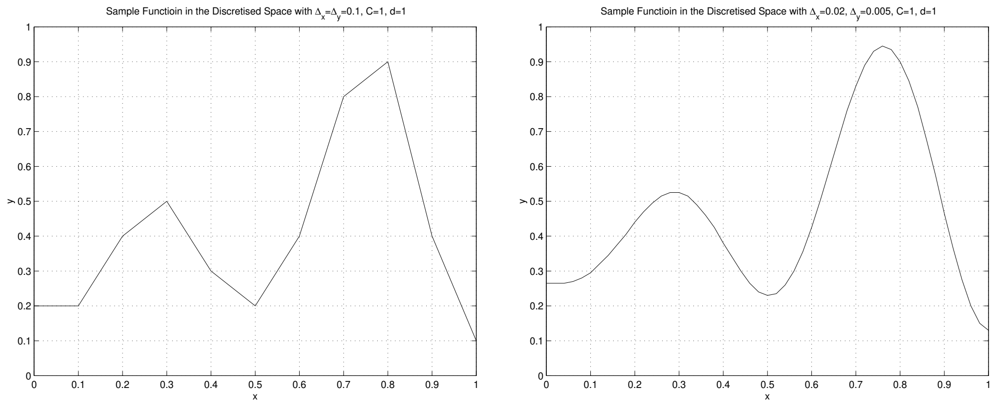
By enforcing hints (and other prior information), one is assigning a prior probability measure to the functions according to whether the hint (other prior information) is obeyed or not. We are subtly doing this by restricting ourselves to a learning model in the first place – a uniform prior over the learning model and zero outside. One expects that the better our prior, the better the performance should be – approximating a sine function in the Fourier basis should certainly do better than (say) using a polynomial basis, with the same amount of effort. Unfortunately, obtaining prior information requires knowledge about the target function, which is the *unknown*. As a result, in many applications, one simply assumes a uniform prior.

In the limit of infinite data, the dependence on the prior usually disappears. In the real world, however, one is dealing with a finite amount of data. In this real world, can one get away with assuming as uninformative a prior as the uniform one? Various no free lunch results to be discussed in the next chapter show that unless one learns with a specific prior in mind, all hypotheses are just as likely to yield a given generalization behavior. Any non-trivial performance can only be claimed relative to a prior, therefore one is forced to assume a prior.

The next two chapters will focus on this issue of the prior. We will discuss the prediction of noise distributions from noisy data, and the need for a prior on the target function. Then, we will discuss how priors could be used in the derivation of regularizers, such as the first order Tikhonov regularizer. We begin here by laying the foundations for the uniform prior, that which is almost always assumed, and proceed from there.

We will concentrate on compact spaces, and assume that the input space is the unit hypercube in d dimensions. Let the output space be the bounded interval $[0, C]$. Discretize the input and output spaces as follows. Let the input variable, $\mathbf{x}$, take on values in the set

$$\mathcal{X} = \{0, \Delta_x, 2\Delta_x, \ldots, 1 - \Delta_x, 1\}^d \tag{7.86}$$



(a) (b)

The discretized function space has only a finite number of functions. We show a particular function on different quantization scales. An arbitrary precision can be obtained by making the grid finer and finer.

Figure 7.2: The discretised function space

so there are $N_x = 1 + \frac{1}{\Delta_x}$ possible values for each input variable. Let the output variable, y, take a value in the set

$$\mathcal{Y} = \{0, \Delta_y, 2\Delta_y, \ldots, C - \Delta_y, C\} \tag{7.87}$$

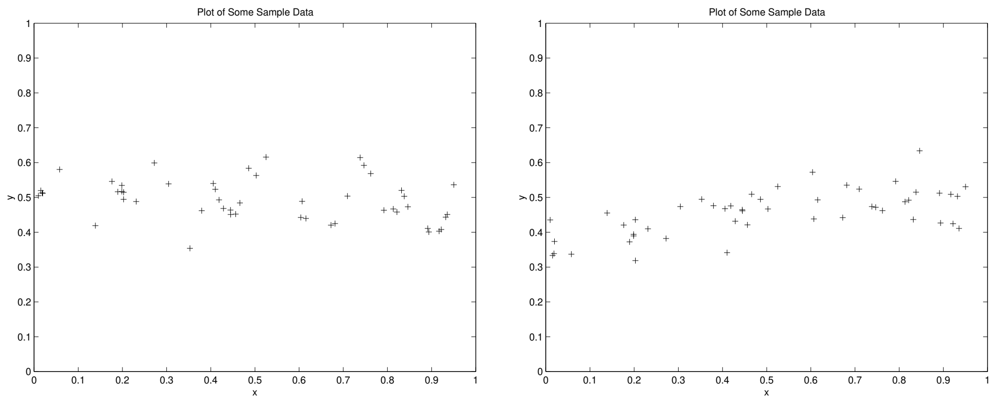
There are $N_y = \frac{C}{\Delta_y}+1$ possible y values. A function (f) will be a mapping, $f : \mathcal{X} \to \mathcal{Y}$. There are $\Gamma = {N_y}^{{N_x}^d}$ different functions. By making Δ_x, Δ_y as small as we choose, we can achieve an arbitrary precision (for example see figure 7.2). In all applications, only a finite number of functions are available (computers are finite precision machines). Due to this finiteness, we can define a probability distribution on the function space – a prior over the function space. One natural prior is to take this distribution to be uniform. Therefore to every function we assign a probability of $\frac{1}{\Gamma}$. Many other "natural" priors are plausible.

Chapter 8

No Noisy Free Lunch (NNFL)

— We all have our own priors. Ultimately, what matters is the degree of expression.

Which data set has more noise?



(a) (b)

Two sample noisy data sets created by taking two functions and adding noise to them. The x*-values are the same in both cases.*

Figure 8.1: Two sample noisy data sets.

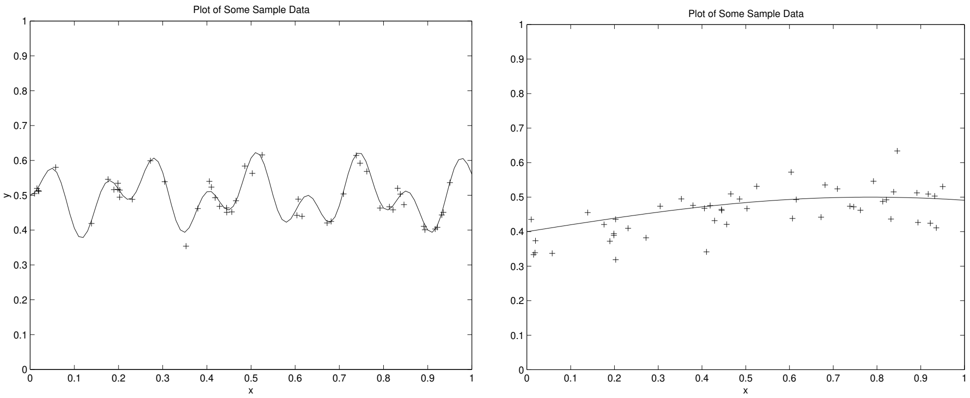
8.1 Introduction

Already established in the learning theory literature are a set of "No Free Lunch" (NFL) theorems for various types of learning systems, [Wolpert, 1996c], [Wolpert, 1996b], [Zhu, 1996], [Cataltepe *et al.*, 1998] and for optimization methods [Wolpert, 1996a]. The essential content of these theorems is that no learning algorithm can be universally good. A learning algorithm that performs exceptionally well in certain situations will perform comparably poorly in other situations. Thus, in particular, random algorithms that always yield average performance, cross validation, bootstrapping and anti-cross validation are all, in some sense, on an equal footing. Stopping early at a higher training error than the minimum achievable gains nothing (for generalized linear models) if the hypotheses yielding that training error are chosen with equal probability. An optimization technique that performs well with one cost function will perform poorly on another cost function. Thus, one needs to make some assumptions about the underlying structure of the problem to be able to claim any kind of superior performance.

Here we extend these ideas to the prediction of noise distributions. The basic problem is to deduce properties of the noise distribution (for example, the noise variance) from a finite data set that consists of input–output pairs. The outputs have a deterministic dependence on the inputs (the target function) and a noisy component. One approaches the problem with some prior belief about the noise distribution, and one hopes that the data will enable a fine tuning of that prior belief to the extent that more precise statements can be made about the properties of the noise. An accurate estimate of the noise distribution is useful because the noise sets a fundamental limit on the performance of a learning system [Cortes *et al.*, 1994]. Noise may play other roles as well, for example, in financial markets, the noise level is itself a tradable quantity.

We will show that having a uniform prior on the possible realizations of the target function does not allow one to update a well behaved prior on the noise distributions (in the Bayesian sense) from any finite data set. Lets go back to the question posed at the beginning (figure 8.1). Which data set has more noise? Figure 8.2 shows the same data, but this time with the function that created the data. Now, which data set has more noise? The task is considerably easier given this extra information. We will show that some extra information of this form is essential.

This chapter is organized as follows. We start with an example in section 8.2 where we attempt to predict a noise variance when learning using linear models. Section 8.3



(a) (b)

The same noisy data sets that were shown in figure 8.1, and the functions that were used to produce them. In (b), the noise variance is roughly six times larger.

Figure 8.2: Noisy data and the functions that generated them

sets up the problem in general and section 8.4 presents the main results, beginning with the simple case of boolean functions, and continuing to the more general case. We will show that information about the noise distribution cannot be obtained from a finite data set if the target distribution is assumed uniform. We conclude with section 8.5 where we also discuss the implications of the NFL theorems.

8.2 An Example Using Linear models

In this section we illustrate the conclusions of this paper for the case of linear learning models on the input space $\mathbb{R}^d$. It is not our goal to be strictly rigorous, but rather to illustrate the point. The hypothesis functions are of the form

$$g_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + \mathbf{w}_0 \tag{8.1}$$

$\mathbf{w}, \mathbf{w}_0$ are called the weights. We are given a data set $\mathcal{D} = \{\mathbf{x}_\alpha, y_\alpha = f(\mathbf{x}_\alpha) + n_\alpha\}_{\alpha=1}^{N}$. $f(\mathbf{x}_\alpha)$ is some deterministic function (we place no other restrictions on the function) and n_α is noise, drawn *i.i.d.* with zero mean and variance σ^2. We use mean squared deviation as the measure of error. Define the training error as

$$\mathcal{E}_{tr} = \frac{1}{N}\sum_{\alpha}(\mathbf{w}\cdot\mathbf{x}_\alpha + w_0 - y_\alpha)^2 \tag{8.2}$$

Define $\mathbf{w}^*, \mathbf{w}_0^*$ as those weights that minimize $\mathcal{E}_{tr}$. It can then be shown that at this minimum, the expected training error is given by (theorem 8.6.1)

$$\langle {\mathcal{E}_{tr}}^* \rangle_{\mathcal{D},\mathbf{x},\epsilon} = E_0 + \sigma^2 - \frac{\sigma^2(d+1)+B}{N} + o(\frac{1}{N}) \tag{8.3}$$

we use $\langle\cdot\rangle$ to denote expectations and the expectation here is over possible data sets, the input space and the noise. E_0 is the $BIAS$, where the $BIAS$ is the expected training error of the best hypothesis function in our learning model. B is a constant given by theorem 8.6.1. For the purposes of this section, the exact expression for B is not essential. It suffices to know that B is related to the variance of a sample statistic around its population value.

The training error used as an estimate of the noise variance is asymptotically biased. It over predicts the noise variance by the $BIAS$. Since we know the form of $\langle {\mathcal{E}_{tr}}^* \rangle$, perhaps we can do better. We can sample N_1 data points from the N data points and estimate the training error. We use bootstrapping [Shao and Tu, 1996] to estimate $\langle {\mathcal{E}_{tr}}^*(N_1) \rangle$, and by varying N_1 we can fit the resulting dependence of $\langle {\mathcal{E}_{tr}}^*(N_1) \rangle$ on $1/N_1$ using (8.3). Let the slope of this fit be $\hat{A}$. From (8.3), we see that $\hat{A}$ is an estimate for $\sigma^2(d+1)+B$, therefore, $\hat{A}/(d+1)$ used as an estimate of σ^2 is off by $B/(d+1)$. Perhaps we can estimate B by the bootstrap as well. Suppose we draw N_B data points and estimate B (using the bootstrap technique) by $\hat{B}$. It can be shown that (theorem 8.6.2)

$$\left\langle \hat{B} \right\rangle_\epsilon = B + \sigma^2(d+1)(1-\frac{N_B}{N}) + \frac{\sigma^2}{N}\underbrace{tr\left\langle \Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\right\rangle}_{O(d+1)} \tag{8.4}$$

Thus we now have two quantities, $\hat{A}$, an estimate of $B+\sigma^2(d+1)$ and $\hat{B}$, an estimate of $B+\sigma^2((d+1)(1-N_B/N)+tr\left\langle\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\right\rangle/N)$. Solving for σ^2, we have

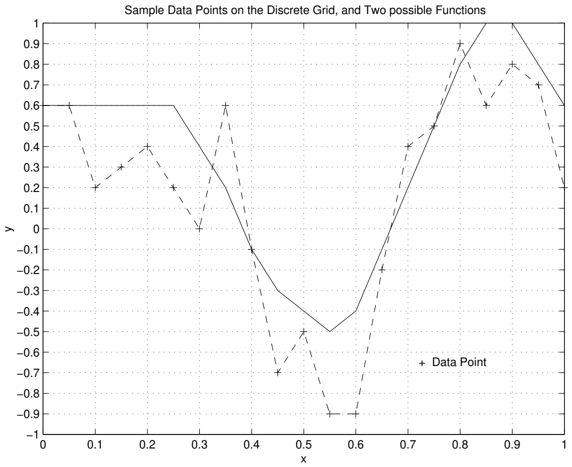
$$\sigma^2 = \frac{\hat{A}-\hat{B}}{(d+1)\frac{N_B}{N}-\frac{tr\left\langle\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\right\rangle}{N}} \tag{8.5}$$

However, we are bootstrapping a quantity B, which is related to the deviation of a statistical quantity of the sample from its population value. This estimate is good only as long as N represents the population – i.e., $N_B/N \to 0$. In this case, $\hat{B} = B+\sigma^2(d+1)$. Combining this with $\hat{A}$, our other estimate for $\sigma^2(d+1)+B$, we get no information on σ^2. We are faced with a dilemma. For finite N_B/N we can get an estimate of σ^2 but this estimate is not very good. However, we have no way to make it better because by making the bootstrap more effective, the final result loses its dependence on σ^2 - technically speaking, we cannot in this way get a consistent estimate of σ^2, even as $N \to \infty$.

Thus, to get information on σ^2, perhaps we should look at terms with higher order in $1/N$. Perhaps some method other than simply using the training error would yield a better result.

The purpose of the next few sections is to show that such an exhaustive search will necessarily be fruitless unless we make some statement about f – in this case it would suffice to tell us "how good our model is at estimating f" (i.e., E_0). Rather than explore every other potential method for estimating σ^2 in this linear scenario, we will set up a more general framework for the prediction of noise and show that unless some assumptions are made about the target function, all noise models are just as probable, given the data. The intuition is that the data points are consistent with *any* noise level if we do not restrict the target function. We pursue these issues next.

8.3 Problem Set Up



Suppose that the two functions shown are both equally likely (probabiliity 1/2) to be the function that created the data. Then, the two hypotheses noise=0 and noise>0, are equally likely to be true.

Figure 8.3: An example discretised grid

A finite data set $\mathcal{D} = \{\mathbf{x}_\alpha, y_\alpha\}_{\alpha=1}^{l}$ is given, where $\mathbf{x}_\alpha \in \mathbb{R}^d$, $y_\alpha = f(\mathbf{x}_\alpha) + n_\alpha$and $f : \mathbb{R}^d \to \mathbb{R}$. We will concentrate on the case where the noise distribution is fixed. The results can be modified to accommodate noise distributions that vary with $\mathbf{x}$. It will always be understood that α indexes points in the data set.

We will use the grid representation of functions as has already been defined, with one small modification. We will allow the y values to be negative as well. Thus, we allow the output to take the values $\{-C, -C+\Delta_y, C+2\Delta_y, \ldots, C-\Delta_y, C\}$. There are now $N_y = \frac{2C}{\Delta_y} + 1$ possible function values. We assume that C can be made as large as we wish. Further, without loss of generality, we can scale the outputs by $\frac{1}{\Delta_y}$, therefore we can assume that $y \in \{-\tilde{C}, -\tilde{C}+1, -\tilde{C}+2, \ldots, \tilde{C}-1, \tilde{C}\}$, $N_y = 2\tilde{C}+1$ and $\tilde{C} = C/\Delta_y$. With this representation, the noise $n \in \{0, \pm 1, \pm 2, \ldots\}$ has a distribution given by a vector $\mathbf{P}$ where $P_i = Pr[n = i]$ and $\sum_i P_i = 1$. Assume a prior probability measure on the possible noise distributions $g(\mathbf{P})$. We are interested in the Bayesian posterior on these noise distributions

$$g(\mathbf{P}|\mathcal{D}) = \frac{Pr[\mathcal{D}|\mathbf{P}]g(\mathbf{P})}{Pr[\mathcal{D}]} \tag{8.6}$$

where $Pr[\mathcal{D}] = \sum_{\mathbf{P}} Pr[\mathcal{D}|\mathbf{P}]g(\mathbf{P})$. We can calculate $Pr[\mathcal{D}|\mathbf{P}]$ as follows.

$$Pr[\mathcal{D}|\mathbf{P}, f] = \prod_{\alpha=1}^{l} Pr[n_\alpha|f(\mathbf{x}_\alpha)] \tag{8.7}$$

multiplying by $P[f]$ and summing over f we have

$$\begin{aligned} Pr[\mathcal{D}|\mathbf{P}] &= \frac{1}{\Gamma}\sum_f \prod_{\alpha=1}^{l} Pr[n_\alpha|f(\mathbf{x}_\alpha)] \\ &\overset{(a)}{=} \frac{1}{N_y^l}\prod_{\alpha=1}^{l}\sum_{f(\mathbf{x}_\alpha)=-C}^{C} Pr[n_\alpha|f(\mathbf{x}_\alpha)] \end{aligned} \tag{8.8}$$

where (a) follows when we sum over all points not in the data set, and we have used the fact that $Pr[f] = 1/\Gamma$ - uniform prior.

8.4 No Free Lunch for Noise Prediction

8.4.1 Boolean Functions

Notation:

$$\begin{aligned} \mathbf{x} &\in \{0,1\}^d \\ f : \mathbf{x} &\to \{0,1\} \\ (1-p) \in [0,1] &= \textit{probability of flip} \end{aligned}$$

There are $\Gamma = 2^{2^d}$ possible functions.

Theorem 8.4.1 *Let $g(p)$ be the prior distribution for the noise parameter, p. If the prior distribution on the possible functions is uniform then $g(p|\mathcal{D}) = g(p)$.*

PROOF.

Let m_i be the number of times f_i agrees with $\mathcal{D}$ where we use i to index functions. Then $Pr[\mathcal{D}|p, f_i] = p^{m_i}(1-p)^{l-m_i}$, so

$$Pr[\mathcal{D}, f_i|p] = Pr[\mathcal{D}|p, f_i]Pr[f_i|p] = \frac{1}{\Gamma}p^{m_i}(1-p)^{l-m_i} \tag{8.9}$$

Let $\rho(m_i)$ be the number of functions agreeing exactly m_i times with the data set $\mathcal{D}$. $\rho(m_i) = \binom{l}{m_i} 2^{2^n - l} = \binom{l}{m_i} 2^{-l}\Gamma$. Summing (8.9) over f_i, we get

$$\begin{aligned} Pr[\mathcal{D}|p] &= \sum_{f_i} Pr[\mathcal{D}, f_i|p] \\ &= \frac{1}{\Gamma}\sum_{m_i=0}^{l} p^{m_i}(1-p)^{l-m_i}\rho(m_i) \\ &= \frac{1}{2^l}\sum_{m_i=0}^{l} p^{m_i}(1-p)^{l-m_i}\binom{l}{m_i} \\ &= 2^{-l} \end{aligned}$$

independent of p. Therefore,

$$g(p|\mathcal{D}) = \frac{Pr(\mathcal{D}|p)g(p)}{Pr(\mathcal{D})} = g(p)$$

. ∎

8.4.2 Extension to More General Functions

We would like to extend the previous analysis to the case where the output is not binary and the input is the discretized version of $[0,1]^d$. Figure 8.3 illustrates the discretised grid we will work with and some sample functions and data.

Cyclic Noise

When the noise is additive modulo N_y then $\mathbf{P} = [P_0, P_1, \ldots, P_{N_y-1}]$ fully specifies the noise distribution.

example: The binary case with probability of flip $1-p$ we have $N_y = 2,\ P_0 = p,\ P_1 = 1-p$

Lemma 8.4.2 $\sum_{f(\mathbf{x}_\alpha)} Pr[n_\alpha | f(\mathbf{x}_\alpha)] = 1$ *for all* $\mathbf{x}_\alpha \in \mathcal{D}$.

PROOF:

$$\sum_{f(\mathbf{x}_\alpha)=-C}^{C} Pr[n_\alpha | f(\mathbf{x}_\alpha)] = \sum_{n_\alpha = y_\alpha + C}^{y_\alpha - C} Pr[n_\alpha | f(\mathbf{x}_\alpha)] = \sum_{n_\alpha=0}^{N_y-1} P_{n_\alpha} = 1$$

■

Combining Lemma 8.4.2 with (8.8) we have the following theorem.

Theorem 8.4.3 *Let $g(\mathbf{P})$ be the prior distribution for the cyclic noise. If the prior distribution for the functions is uniform then $g(\mathbf{P}|\mathcal{D}) = g(\mathbf{P})$*

PROOF:

By Lemma 8.4.2 and (8.8), $Pr[\mathcal{D}|\mathbf{P}] = \frac{1}{N_y^l}$. Therefore,

$$g(\mathbf{P}|\mathcal{D}) = \frac{Pr[\mathcal{D}|\mathbf{P}]g(\mathbf{P})}{Pr[\mathcal{D}]} = g(\mathbf{P})$$

■

Note that the boolean case is a special case of the cyclic noise case. In the language of information theory [Cover and Thomas, 1991], the noise forms a symmetric channel between the function value and the output value. A uniform distribution for the function values induces a uniform distribution on the output values, independent of the details of the symmetric noisy channel. Hence, observing the output conveys no information about the noisy channel.

Thresholded Additive Noise

Noise is added to the target and then thresholded so,

$$y_i = \begin{cases} \min\{C, f(\mathbf{x}_i) + n_i\}, & f(\mathbf{x}_i) + n_i \geq 0 \\ \max\{-C, f(\mathbf{x}_i) + n_i\}, & f(\mathbf{x}_i) + n_i < 0 \end{cases}$$

$\mathbf{P} = [P_0, P_{\pm 1}, P_{\pm 2}, \ldots]$. Unlike in the previous case, edge effects make $Pr[\mathcal{D}|\mathbf{P}]$ dependent on $\mathbf{P}$. These effects can be made as small as we please by allowing C to be as large as we please. The intuition is that the data set is finite (y_i is bounded) and so, because the noise distribution must decay to zero, if C is large enough, the probability that $f(\mathbf{x}_\alpha)$ is close to the edge and y_α so far away becomes negligible. This is formalized in the next lemma.

Lemma 8.4.4 *Given $\epsilon > 0, \exists C$ large enough such that*

$$\frac{1}{N_y^l}(1 - \epsilon) \leq Pr[\mathcal{D}|\mathbf{P}] \leq \frac{1}{N_y^l} \tag{8.10}$$

$N_y = 2C + 1$. *Intuitively,* $\epsilon \stackrel{C \to \infty}{\longrightarrow} 0$.

PROOF:

Because $\{P_i\}$ is summable, there exists η such that $\sum_{|i|>\eta} P_i < \frac{\epsilon}{l}$. Let $y_{max} = \max\{y_i\}$ and $y_{min} = \min\{y_i\}$. Choose $C > \max\{|y_{max} + \eta|, |y_{min} - \eta|\}$. We then have,

$$\begin{aligned}
\sum_{f(\mathbf{x}_\alpha)=-C}^{C} Pr[n_\alpha|f(\mathbf{x}_\alpha)] &= \sum_{n_\alpha = y_\alpha + C}^{y_\alpha - C} Pr[n_\alpha|f(\mathbf{x}_\alpha)] \\
&= 1 - \sum_{n_\alpha < y_\alpha - C} Pr[n_\alpha|f(\mathbf{x}_\alpha)] - \sum_{n_\alpha > y_\alpha - C} Pr[n_\alpha|f(\mathbf{x}_\alpha)] \\
&\geq 1 - \frac{\epsilon}{l}
\end{aligned}$$

Where the last inequality follows because by construction $y_{max} - C < -\eta$ and $y_{min} + C > \eta$. Using (8.8) we have

$$\frac{1}{N_y^l} \geq Pr[\mathcal{D}|\mathbf{P}] \geq \frac{1}{N_y^l}(1 - \frac{\epsilon}{l})^l$$

The lemma now follows by using the inequality $(1-x)^n \geq 1 - nx$ for $0 \leq x \leq 1$. ■

Thus we see that by choosing C, the bound on our function f to be large enough, the noise distribution has almost no effect on the probability of obtaining a particular data set. All data sets become equally probable, in the limit of large C. This in turn tells us that the data should not be telling us any more about the noise distribution than we already know. The prior distribution for the noise is unaffected given *any finite* data set. To formalize this intuition, we need to impose a technical condition on the prior over the noise distribution space.

Definition 8.4.5 *Let* $0 < \epsilon \leq 1$*. Define* $\eta_{\mathbf{P}}(\epsilon)$ *by*

$$\eta_{\mathbf{P}}(\epsilon) = \min \left\{ \eta : \sum_{|i|>\eta} P_i < \epsilon \right\}$$

For all $0 < \epsilon \leq 1$, $\eta_{\mathbf{P}}(\epsilon) < \infty$, by the summability of $\mathbf{P}$.

Definition 8.4.6 *Let* $0 < \epsilon \leq 1$*. Define* $\bar{\eta}(\epsilon)$ *by*

$$\bar{\eta}(\epsilon) = \sup_{\mathbf{P}} \{\eta_{\mathbf{P}}(\epsilon)\}$$

where the sup *is taken over* $\mathbf{P} \in$ *support*$\{g(\mathbf{P})\}$*.*

Definition 8.4.7 *A prior on the space in which* $\mathbf{P}$ *is defined,* $g(\mathbf{P})$*, is* <u>*totally bounded*</u> *if*

$$\bar{\eta}(\epsilon) < \infty$$

for all $\epsilon > 0$*.*

Essentially this means that the support set for the prior on the noise distributions is not "too large." For example, comparing only a finite number of noise distributions yields a totally bounded prior over the noise distribution space. We are now in a position to make precise the intuition mentioned above.

Theorem 8.4.8 (No Noisy Free Lunch (NNFL)) *Let* $g(\mathbf{P})$ *be totally bounded. For all* $0 < \epsilon < 1$, $\exists C > 0$:

$$(1-\epsilon)g(\mathbf{P}) \leq g(\mathbf{P}|\mathcal{D}) \leq \frac{g(\mathbf{P})}{1-\epsilon} \tag{8.11}$$

Therefore,

$$\lim_{C\to\infty} g(\mathbf{P}|\mathcal{D}) = g(\mathbf{P})$$

because $\epsilon \to 0$ *can be attained by letting* $C \to \infty$*.*

PROOF:

Following lemma 8.4.4 now let $C > \max\{|y_{max} + \bar{\eta}(\epsilon)|, |y_{min} - \bar{\eta}(\epsilon)|\}$. Then Lemma 8.4.2 applies uniformly to every $\mathbf{P} \in support\{g(\mathbf{P})\}$. Therefore (8.10) holds for every $\mathbf{P} \in support\{g(\mathbf{P})\}$.

$$\begin{aligned} g(\mathbf{P}|\mathcal{D}) &= \frac{Pr[\mathcal{D}|\mathbf{P}]g(\mathbf{P})}{Pr[\mathcal{D}]} \\ &= \frac{Pr[\mathcal{D}|\mathbf{P}]g(\mathbf{P})}{\sum\limits_{\mathbf{P}} Pr[\mathcal{D}|\mathbf{P}]g(\mathbf{P})} \end{aligned} \tag{8.12}$$

Using (8.10), an upper bound can be obtained by substituting $Pr[\mathcal{D}|\mathbf{P}]$ with $1/N_y^l$ in the numerator, and $(1-\epsilon)/N_y^l$ in the denominator. To get a lower bound, we do the opposite. Then, using the fact that $\sum_{\mathbf{P}} g(\mathbf{P}) = 1$, we get (8.11). ■

Therefore, having no bound on f and allowing all f's to be equally likely does not allow anything new to be deduced about the noise distribution given any finite data set. The preceding results essentially formalize the intuition that if every target function value is equally likely, then, each noise realization for every data point is equally likely. Hence nothing can be deduced about the relative likelihoods of different noise realizations. It should now be clear how to extend these results to the case where the noise distribution has $\mathbf{x}$ dependence – one looks at each point independently.

The non-thresholded additive noise model is asymptotically obtained by letting C become arbitrarily large. Further the results apply to arbitrary $\Delta_{\mathbf{x}}$, Δ_y, so by allowing $\Delta_{\mathbf{x}}$, $\Delta_y \to 0$, these results can be applied to the non-discretized model in this asymptotic sense.

8.5 Comments

The updated prior $g(\mathbf{P}|\mathcal{D})$ is given by

$$g(\mathbf{P}|\mathcal{D}) \propto g(\mathbf{P}) \sum_f Pr[\mathcal{D}|\mathbf{P}, f]P[f] \tag{8.13}$$

where $P[f]$ is the prior over target functions. We have demonstrated that when $P[f]$ is "uniform", it is not possible to update a prior on the noise distribution from a finite data set, provided that a certain technical condition is satisfied. A non trivial $P[f]$ is needed for $g(\mathbf{P}|\mathcal{D})$ to be different from $g(\mathbf{P})$. Exactly how $P[f]$ will factor into $g(\mathbf{P}|\mathcal{D})$ is given by (8.13).

The result is more useful than it appears at first sight, for one might argue that no one ever has a uniform prior. A restatement of the result is: Given two noise distributions, $\mathbf{P}_1$ and $\mathbf{P}_2$, there are just as many (non-uniform) priors that favor $\mathbf{P}_1$ over $\mathbf{P}_2$ (when one weights by the amount by which $\mathbf{P}_1$ is favored), as vice versa.

This result fits into a set of NFL theorems that might be considered a collection of negative results, offering no hope for learning theory. The situation is considerably more positive, however. One is never in a situation with a uniform $P[f]$. Further, it is rare that no information about $P[f]$ is known (excepting that it is non-uniform). Therefore, incorporating $P[f]$ into the noise prediction or learning algorithm is a must in order to claim superior performance. To this end, distribution independent bounds such as the VC bounds [Vapnik, 1995], though valid, are of little use – one might be able to guarantee that the training error is close to the test error but what use is that, if one cannot guarantee that, on the average, the training error will be low. Results that exemplify the superiority of algorithm A over algorithm B on a certain bench mark problem are also of little use, for the reverse will be true on some other "benchmark" problem.

NFL focuses our attention on the importance of the prior information available. One should not attempt to find learning systems that are universally good. Instead, one should study the possibilities that various priors present. Unfortunately, identifying and justifying the prior for the problem at hand is already a daunting task, let alone incorporating it into the learning problem. One way would be through the use of hints [Abu-Mostafa, 1995], [Sill, 1998b], [Sill and Abu-Mostafa, 1997]. In the next chapter we will consider another way of incorporating prior information into the learning system.

A possible starting point is the identification of priors, $P[f]$, with learning systems that perform well with respect to them, for example [Barber and Williams, 1997], [Zhu and Rohwer, 1996]. One might then be able to relate real problems to these priors and thus choose an appropriate learning system, or at least it might become possible to make precise the assumptions behind a belief that a certain learning system is appropriate for a certain problem. Further, VC type bounds derivable with the knowledge of the prior, rather than with respect to any target function, are likely to be much tighter. What would be a useful result? One of the form "Algorithm A (or Learning System A) performs better than average on problems drawn from the (useful) class B". We attempt to address some of these issues next.

8.6 Appendix

We use the notation

$$\mathbf{X} = [\,\mathbf{x}_1 \quad \mathbf{x}_2 \quad \ldots \quad \mathbf{x}_N\,], \qquad \mathbf{y} = f(\mathbf{x}_\alpha) + \epsilon$$

$$\Sigma \equiv \left\langle \mathbf{x}\mathbf{x}^T \right\rangle_{\mathbf{x}}, \qquad \mathbf{q} = \langle \mathbf{x} f(\mathbf{x}) \rangle_{\mathbf{x}}$$

The law of large numbers gives us that $\mathbf{X}\mathbf{X}^T \underset{N\to\infty}{\longrightarrow} N\Sigma$ and $\mathbf{X}f(\mathbf{x}_\alpha) \underset{N\to\infty}{\longrightarrow} N\,\langle \mathbf{x} f(\mathbf{x}) \rangle_{\mathbf{x}}$. where we assume that the conditions for this to happen are satisfied.

Theorem 8.6.1 *Let the learning model be the set of linear functions* $\mathbf{w}\cdot\mathbf{x} + w_0$ *and let the learning algorithm be minimization of the squared error. Then*

$$\mathcal{E}_{tr}{}^{*} = E_0 + \sigma^2 - \frac{\sigma^2(d+1) + B}{N} + o\left(\frac{1}{N}\right) \tag{8.14}$$

where E_0 *and* B *are constants given by*

$$E_0 = \left\langle f^2 \right\rangle - \mathbf{q}^T \Sigma^{-1} \mathbf{q} \tag{8.15}$$

$$B = \frac{\left\langle \mathbf{q}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q} + \mathbf{a}^T\Sigma^{-1}\mathbf{a} - 2\mathbf{a}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q}\right\rangle}{N} \tag{8.16}$$

Proof: The Least Squares estimate of $\mathbf{w}$ is given by

$$\hat{\mathbf{w}} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y} \tag{8.17}$$

from which we calculate the expected training error as

$$\begin{aligned}
\langle \mathcal{E}_{tr} \rangle &= \left\langle \frac{1}{N}(\hat{\mathbf{X}}^T\mathbf{w} - \mathbf{y})^2 \right\rangle = \left\langle \frac{1}{N}((\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X} - \mathbf{1})y)^2 \right\rangle \\
&= \frac{\left\langle f(\mathbf{x}_\alpha)^T f(\mathbf{x}_\alpha) \right\rangle - \left\langle f^T\mathbf{X}(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}f(\mathbf{x}_\alpha) \right\rangle + \left\langle \epsilon^T\epsilon \right\rangle - \left\langle \epsilon^T\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\epsilon \right\rangle}{N} \\
&= \underbrace{\left\langle f^2 \right\rangle - \frac{\left\langle f(\mathbf{x}_\alpha)^T X^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}f(\mathbf{x}_\alpha) \right\rangle}{N}}_{T_1} + \sigma^2 - \frac{\sigma^2(d+1)}{N}
\end{aligned}$$

By the law of large numbers, we note that $(\mathbf{X}\mathbf{X}^T)^{-1} \underset{N\to\infty}{\longrightarrow} N\Sigma$ and $\mathbf{X}f(\mathbf{x}_\alpha) \underset{N\to\infty}{\longrightarrow} N\mathbf{q}$, so we write

$$\mathbf{X}\mathbf{X}^T = N\Sigma + \sqrt{N}\mathbf{V}(\mathbf{X}), \qquad \mathbf{X}f(\mathbf{x}_\alpha) = N\mathbf{q} + \sqrt{N}\mathbf{a}(\mathbf{X}) \tag{8.18}$$

where $\langle \mathbf{V} \rangle_{\mathbf{X}} = \langle \mathbf{a} \rangle_{\mathbf{X}} = 0$ and $\mathrm{Var}(\mathbf{V})$ and $\mathrm{Var}(\mathbf{a})$ are $O(1)$. Using (8.18) and the identity $[1+\lambda\mathbf{A}]^{-1} = 1 - \lambda\mathbf{A} + \lambda^2\mathbf{A}^2 + O(\lambda^3)$ we evaluate T_1 as

$$\begin{aligned} T_1 &= \left\langle f^2 \right\rangle - \mathbf{q}^T\Sigma^{-1}\mathbf{q} \\ & \quad - \frac{\left\langle \mathbf{q}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q} + \mathbf{a}^T\Sigma^{-1}\mathbf{a} - 2\mathbf{a}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q} \right\rangle}{N} + O(\frac{1}{N^{\frac{3}{2}}}) \end{aligned}$$

The theorem now follows. ■

This result is similar to the results obtained by [Amari *et al.*, 1995],[Moody, 1991]. By theorem 8.6.1, we have that

$$B = \frac{1}{N}\left(\underbrace{\left\langle \mathbf{q}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q} \right\rangle}_{T_1} + \underbrace{\left\langle \mathbf{a}^T\Sigma^{-1}\mathbf{a} \right\rangle}_{T_2} - 2\underbrace{\left\langle \mathbf{a}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q} \right\rangle}_{T_3} \right) \tag{8.19}$$

We wish to estimate T_1, T_2, T_3. Suppose that we try to bootstrap them as follows. For T_1 we take $\mathbf{q} = \mathbf{X}\mathbf{y}/N = \mathbf{X}(f(\mathbf{x}_\alpha)+\epsilon)/N$ and $\Sigma^{-1} = (\mathbf{X}\mathbf{X}^T/N)^{-1}$. We estimate $\mathbf{V}$ by sampling N_B of the N data points and compute $\sqrt{N_B}((\mathbf{X}\mathbf{X}^T/N)^{-1} - (\mathbf{X}_B\mathbf{X}_B^T/N_B)^{-1})$. We then estimate T_1 by $\hat{T}_1$, a bootstrapped expectation of $\mathbf{q}^T\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{q}$. This requires N_B/N to be small. Doing all this, we find that

$$\hat{T}_1 = \frac{f(\mathbf{x}_\alpha)^T\mathbf{X}^T}{N}\left\langle \Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\Sigma^{-1} \right\rangle \frac{\mathbf{X}f(\mathbf{x}_\alpha)}{N} + \frac{\sigma^2}{N} tr\left\langle \Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V} \right\rangle \tag{8.20}$$

where we have taken expectation with respect to the noise. Notice that the first term is the unbiased estimator of T_1.

For $\hat{T}_2 = \left\langle \mathbf{a}^T\Sigma^{-1}\mathbf{a} \right\rangle$ we use $\mathbf{a} = \sqrt{N}(\mathbf{q} - \mathbf{X}_B\mathbf{y}_B/N)$. Partitioning the input matrix and noise vector into the part in the sampled N_B points and the remaining part, write

$\mathbf{X} = [\mathbf{X}_B\ \tilde{\mathbf{X}}_B]$ and $\epsilon = [\epsilon_B^T\ \tilde{\epsilon}_B^T]^T$. Noting that ϵ_B and $\tilde{\epsilon}_B$ are independent, we have for $\hat{T}_2$

$$\begin{aligned}
\hat{T}_2 &= \tilde{T}_2 + \frac{1}{N^2 N_B}(N_B \mathbf{X}\epsilon - N\mathbf{X}_B\epsilon_B)\Sigma^{-1}(N_B\mathbf{X}\epsilon - N\mathbf{X}_B\epsilon_B) && (8.21)\\
&= \tilde{T}_2 + \frac{N_B^2(N-N_B)^2}{N^2 N_B}\left[\frac{\tilde{\mathbf{X}}_B\tilde{\epsilon}_B}{N-N_B} - \frac{\tilde{\mathbf{X}}_B\epsilon_B}{N_B}\right]\Sigma^{-1}\left[\frac{\tilde{\mathbf{X}}_B\tilde{\epsilon}_B}{N-N_B} - \frac{\tilde{\mathbf{X}}_B\epsilon_B}{N_B}\right] && (8.22)\\
&= \tilde{T}_2 + \sigma^2(d+1)\left(1 - \frac{N_B}{N}\right) + o\left(\frac{1}{N}\right) && (8.23)
\end{aligned}$$

where we have taken the expectation with respect to the noise. $\tilde{T}_2$ is an unbiased estimate of T_2.

Performing the same kind of analysis for $\hat{T}_3$ we find that $\hat{T}_3 = \tilde{T}_3$, an unbiased estimate of T_3. Now we estimate $\hat{B}$ by $\hat{T}_1 + \hat{T}_2 - 2\hat{T}_3$ which is the unbiased estimate of B plus some correction terms. Therefore we have proved the following theorem.

Theorem 8.6.2 *For $N \to \infty$, $N_B/N \to 0$, the bootstrapped estimate of B, which we call $\hat{B}$ has an expectation given by*

$$\left\langle \hat{B} \right\rangle_\epsilon = B + \frac{\sigma^2(d+1)}{N}\left(1 - \frac{N_B}{N}\right) + \frac{\sigma^2}{N} tr \left\langle \Sigma^{-1}\mathbf{V}\Sigma^{-1}\mathbf{V}\right\rangle_{\mathbf{X}} \qquad (8.24)$$

■

Chapter 9

Natural Priors and Regularization

— *Natura non fecit saltum*

We have seen that the issue of the prior is important in order to make substantial statements about learning systems. Perhaps the most commonly considered reference prior for Bayesian analysis is the Jeffrey's prior [Jeffreys, 1946] which is invariant under reparametrization and is approximately noninformative [Clarke and Wasserman, 1993], [Clarke and Barron, 1994]. Our goal here is to look at reasonable priors when prior information is available, and to deduce what the optimal learning strategy should be (in the Bayesian sense).

We will diverge from learning theory to consider a purely combinatorial problem. We would like to count the number of functions that satisfy a certain constraint. We will formulate and present an analysis of the problem for the finite grid that we have already discussed. In the asymptotic limit ($\Delta_x \to 0$), we use the central limit theorem to obtain an approximate expression for the solution to the counting problem. We will then return to learning theory in section 9.3 where we will demonstrate how Tikhonov regularization follows by assuming a certain prior and applying some of the results obtained from counting functions.

In what follows, we will retain the discretized grid already defined, and restrict ourselves to one input dimension ($d = 1$). In this case, a function can be viewed as a path on a two dimensional grid, with x monotonically increasing along the path. If we assign a prior probability distribution to the functions, we can then view a function as a random walk on this two dimensional grid, with the distribution of the starting point and the distributions of future jumps being related to that prior.

Suppose we can define some natural characteristic (ρ) of a path. Let $N(\rho)$ be the number of functions with the value of this natural characteristic equal to ρ. Proceed as follows. Let a data set $\mathcal{D}$ be given and assume that the conditional density (likelihood) $P(\mathcal{D}|f)$ is known. Usually, this involves assuming a noise model and then computing the density that a particular noise realization occurred. Taking the Bayesian approach, we would like to find f that maximizes

$$P(f|\mathcal{D}) = \frac{P(\mathcal{D}|f)P(f)}{P(D)} \tag{9.1}$$

or, equivalently, one can maximize its logarithm, $\mathcal{E}$, given by

$$\mathcal{E} = \log[P(f|\mathcal{D})] = \log[P(\mathcal{D}|f)] + \log[P(f)] - \log[P(D)] \tag{9.2}$$

The last term can be ignored, and typically $P(f)$ is assumed to be uniform, so the $\log(P(f))$ term is also ignored, resulting in the maximum likelihood method for estimation. In the case of Gaussian noise, this results in having to minimize a sum of squared errors[1]. From the NFL results, we know that assuming $P(f)$ to be uniform is not a fruitful approach.

Suppose we can define some natural characteristic (ρ) of a path. Let $N(\rho)$ be the number of functions with the value of this natural characteristic equal to ρ. Because ρ is implied by knowing the function, we can write

$$P(f) \quad = \quad P(f, \rho) = P(f|\rho)P(\rho) \tag{9.3}$$

Now suppose that ρ is the *determining* characteristic that governs how the target function was picked. Then it seems reasonable to assume that all functions with the

[1]Technically, Gaussian noise can not allowed on our grid but we will just use it as an illustrative, familiar example.

same value for ρ are equally likely:

$$P(f|\rho) = \frac{1}{N(\rho)}\delta_{\rho,\rho(f)} \tag{9.4}$$

Further if one also stipulates that all ρ's are equally likely ($P(\rho) = const$), then, (9.2) leads to the following problem

$$\max_f \{\log[P(\mathcal{D}|f)] - \log[N(\rho(f))]\} \tag{9.5}$$

The following definition should now be intuitive.

Definition 9.0.3 *A natural characteristic, ρ, of the target function distribution is a property according to which functions are chosen such that*

- *All target functions with the same value of ρ are equally likely.*

If in addition,

- *All values of ρ are equally likely.*

then we call it a maximally informative characteristic (MIC).

Now, by definition, (9.5) is the problem we need to solve for MIC's. In general, we solve

$$\max_f \{\log[P(\mathcal{D}|f)] - \log[N(\rho)] + \log[P(\rho)]\} \tag{9.6}$$

where $\rho = \rho(f)$. All that remains is to find a suitable ρ for the problem at hand. $P(\rho)$ is a prior belief about how ρ is distributed. If $P(\rho) \propto N(\rho)$ then (9.6) reduces to maximum likelihood estimation. If ρ is a determining characteristic, and we have no reason to believe that any value of ρ is more likely than another, then we get (9.5). We have broken down the problem into two parts. The calculation of $N(\rho)$ which is purely a combinatorial problem, and the finding of a natural characteristic, the property of the prior on the target function required for (9.5) and the calculation of $N(\rho)$ to go through. The art is all in the natural characteristic – the ability to choose a natural characteristic that represents the true nature of the learning problem, and one for which $N(\rho)$ can be calculated. Given a problem, one needs some idea about how the target function was picked. We will reverse this process. We will consider a certain set of natural characteristics and from them, determine the objective function that one should optimize. This is a minor beginning toward the development of a catalog of natural characteristics and the objective functions they imply. A practitioner, with this catalog in hand, approaches a problem and tries to decide which natural characteristic to use. Then what remains is to optimize the objective function (9.5) that is implied by that characteristic. The characteristics that we will consider are variations of the *total variation* of a function, and some other reasonable bound constraints.

Define the variations, $\mathcal{T}_{ij}$, of a function by

$$\mathcal{T}_{ij}(f) = \int dx \left| \frac{\partial^i f}{\partial x^i} \right|^j \tag{9.7}$$

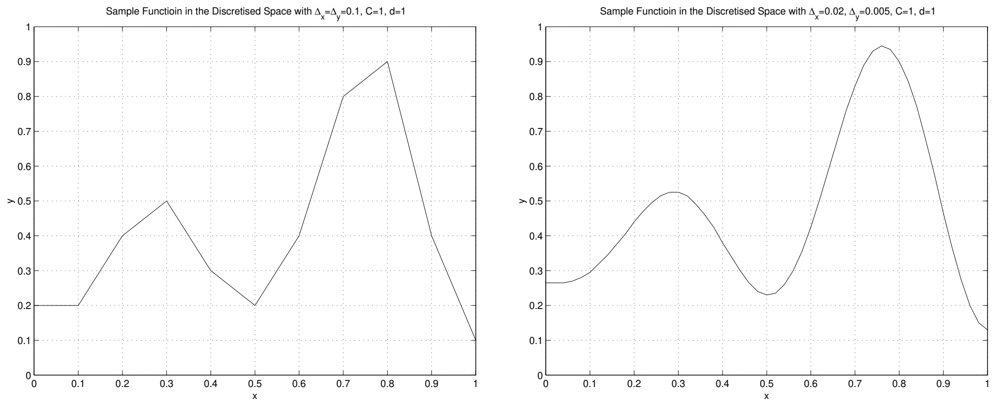
We will be interested in discretized versions of $\mathcal{T}_{11}, \mathcal{T}_{12}, \mathcal{T}_{22}$, the most commonly used Tikhonov type regularization terms.

9.1 Counting Functions

Let ρ for a function (path) be defined by

$$\rho(f) = \sum_{i=0}^{N_x - 1} e_i \tag{9.8}$$

where i indexes the possible x values. For example, if $e_0 = 0$, $e_i = |s_i - s_{i-1}|$ where $s_i = y_i/\Delta_y$, then ρ is the sum of the number of jumps in the path. It is a discrete analog of the integrated magnitude of the gradient, $\mathcal{T}_{11}$. In figure 9.1 (a), $\rho = 21$. For figure 9.1 (b), $\rho = 417$ (though this is not obvious from simply looking at the figure). Let $N(\rho)$ be the number of functions with $\sum e = \rho$. Our goal is to calculate $N(\rho)$ for



(a) (b)

The discretized function space has only a finite number of functions. We show a particular function on different quantization scales. An arbitrary precision can be obtained by making the grid finer and finer.

Figure 9.1: The discretised grid

various functional forms $\rho(f)$. We will assume that $e_i \geq 0$.

9.1.1 Setting up the Combinatorial Problem

Let $B(\rho, N_x, N_y, i, j)$ be the number of paths with $\sum e = \rho$ on the grid of size $N_x \times N_y$ with $s_0 = i$ and $s_{N_x} = j$. If either of the conditions $\rho < 0$, $N_x \leq 0$, $N_y < 0$, then B is defined to be zero. In this section, we will consider ρ of the form

$$\rho = \sum_{i=1}^{N_x} e(s_i, s_{i-1}) \tag{9.9}$$

when $e(x, y) = |x - y|$, ρ is the total variation. Assume that $e(\cdot, \cdot)$ is a symmetric function. From $B(\rho, N_x, N_y, i, j)$ we can calculate $N(\rho)$ as

$$N(\rho) = \sum_{i,j=0}^{N_y} B(\rho, N_x, N_y, i, j) \tag{9.10}$$

By symmetry, $B(\rho, N_x, N_y, i, j) = B(\rho, N_x, Ny, j, i)$.

9.1.2 Counting the Number of Functions

We will present two methods for calculating $B(\rho, N_x, N_y, i, j)$, one based on a recursive argument, and one based on computing the generating function for $B(\rho, N_x, N_y, i, j)$.

Recursive Approach

Proposition 9.1.1 *Let $N_1 > 0$, $N_2 > 0$ be two integers such that $N_1 + N_2 = N_x$, then*

$$B(\rho, N_x, N_y, i, j) = \sum_{k=0}^{N_y} \sum_{r} B(r, N_1, N_y, i, k) B(\rho - r, N_2, N_y, k, j) \tag{9.11}$$

PROOF: Fix k, r in the summations. Then, $B(r, N_1, N_y, i, k)B(\rho - r, N_2, N_y, k, j)$ is the number of paths from i to j passing through the point (N_1, k) with $\sum e = r$ in the first N_1 steps and $\sum e = \rho - r$ in the remaining N_2 steps. Summing over the possible values for r, we get the number of paths from i to j passing through the point (N_1, k) with $\sum e = \rho$. Summing over the possible values of k we get the number of paths from i to j with $\sum e = \rho$. ■

Given (9.11), we can compute $B(\rho, N_x, N_y, i, j)$ by starting at $N_x = 1$ and building upwards. More precisely, note that $B(\rho, 1, N_y, i, j) = \delta_{\rho, e(i,j)}$. Substituting this into (9.11) we get the following recursion

$$B(\rho, 1, N_y, i, j) = \delta_{\rho, e(i,j)} \tag{9.12}$$

$$B(\rho, N_x + 1, N_y, i, j) = \sum_{k=0}^{N_y} B(\rho - e(j,k), N_x, N_y, i, k) \tag{9.13}$$

Now, $B(\rho, N_x, N_y, i, j)$ for any values of its arguments can be recursively computed by starting with $N_x = 1$ and increasing N_x by increments of 1 to the desired value. In the case where $e(x, y) = |x - y|$, the recursion becomes

$$B(\rho, 1, N_y, i, j) = \delta_{\rho, |x-y|} \tag{9.14}$$

$$B(\rho, N_x+1, N_y, i, j) \;=\; \sum_{k=0}^{N_y} B(\rho - |x-y|, N_x, N_y, i, k) \tag{9.15}$$

Generating Function for $B(\rho, N_x, N_y, i, j)$

We would like to calculate $P(\rho|i,j)$, the probability of obtaining a path from i to j with $\sum e = \rho$, where we take s_k to be $i.i.d.$ and $P(s_k = m) = 1/N_y$. Then,

$$B(\rho, N_x, N_y, i, j) = {N_y}^{N_x-2} P(\rho|i,j) \tag{9.16}$$

We can also calculate $P(\rho)$ from $P(\rho|i,j)$:

$$P(\rho) = \sum_{i,j} P(\rho|i,j) P(i,j) = \frac{1}{{N_y}^2} \sum_{i,j} P(\rho|i,j) \tag{9.17}$$

and now $N(\rho)$ is given by ${N_y}^{N_x} P(\rho)$. Let $\psi_{ij}(s) \equiv \left\langle e^{\hat{i}s\rho} \right\rangle_{P(\rho|i,j)}$. Since ρ takes on a discrete set of values, we can write

$$\psi_{ij}(s) = \sum_{\rho} P(\rho|i,j) r^{\rho} \tag{9.18}$$

where $r = e^{\hat{i}s}$. Given ψ_{ij}, we can read off $P(\rho|i,j)$ as the coefficient of r^{ρ}. Therefore we concentrate on ψ_{ij}.

$$\begin{aligned}
\psi_{ij}(s) &= \frac{1}{{N_y}^{N_x-2}} \sum_{s_1=0}^{N_y} \cdots \sum_{s_{N_x-1}=0}^{N_y} r^{\left(\sum_{i=1}^{N_x} e(s_i, s_{i-1})\right)} && (9.19)\\
&= \frac{1}{{N_y}^{N_x-2}} \sum_{s_1 \ldots s_{N_x-1}} r^{e(i,s_1)} r^{e(s_1,s_2)} \ldots r^{e(s_{N_x-1},s_j)} && (9.20)\\
&= \frac{1}{{N_y}^{N_x-2}} [\mathbf{M}^{N_x-1}]_{ij} && (9.21)
\end{aligned}$$

where we define an $N_y \times N_y$ "transfer matrix" $\mathbf{M}$ by

$$\mathbf{M}_{ij} = r^{e(i,j)} \tag{9.22}$$

To get $B(\rho, N_x, N_y, i, j)$ we need to find the coefficient of r^ρ in the expression for ψ_{ij}

$$B(\rho, N_x, N_y, i, j) = [\mathbf{M}^{N_x-1}]_{ij}\Big|_{r^\rho} \tag{9.23}$$

and for $N(\rho)$ we find the coefficient of r^ρ in the expression for $\psi = \sum_{i,j} \psi_{ij}$

$$N(\rho) = \sum_{i,j=0}^{N_y} [\mathbf{M}^{N_x-1}]_{ij}\Bigg|_{r^\rho} \tag{9.24}$$

When $e(x, y) = |x - y|$, $\mathbf{M}$ becomes the Toeplitz matrix given by $\mathbf{M}_{ij} = r^{|i-j|}$

$$\mathbf{M} = \begin{pmatrix} 1 & r & r^2 & \dots & r^{N_y-1} \\ r & 1 & r & \dots & r^{N_y-2} \\ \vdots & & \ddots & & \vdots \\ r^{N_y-2} & \dots & r^2 & r & 1 \end{pmatrix} \tag{9.25}$$

Technical Aside

We note a connection here to counting paths for a bounded random walk with variable step size, each step size having the same probability. The bounded grid is exactly the grid we have been talking about all along. If we want to calculate the probability that, having started at point i and ended at point j, the total distance traveled in the y direction is ρ, then we need to calculate the coefficient of the $e^{\hat{i}s\rho}$ term in $[\mathbf{M}^{N_x-1}]_{ij}$ where $\mathbf{M}$ is given by (9.25). If N_x is large, then, the largest eigenvalue of $\mathbf{M}$ will exponentially dominate. Diagonalizing $\mathbf{M}$ and ignoring the lower eigenvalues of $\mathbf{M}$ we find that

$$[\mathbf{M}^{N_x-1}]_{ij} \approx \alpha(e^{\hat{i}s}) z_i(e^{\hat{i}s}) z_j(e^{\hat{i}s}) \tag{9.26}$$

where $\alpha(s)$ is the largest eigenvalue and $\mathbf{z}$ is the corresponding eigenvector. [Grenander and Szegö, 1958, pg 69-75] studies this eigenvalue in some detail, including its limiting form as $N_y \to \infty$.

9.2 Asymptotic Solution to the Counting Problem

A closed form solution to the the recursive approach (9.11) or the generating function approach (9.21) has yet to be found for the cases of interest. For reasonable grid sizes, $N(\rho)$ could perhaps be computed numerically using either of the above methods. In practice, $N_x$and N_y are large ($O(2^{53})$), and the asymptotic limits $N_x, N_y \to \infty$ become relavent, so we can make use of the central limit theorem. We will need a version of the central limit theorem that is valid for dependent random variables. For convenience we state it here.

Theorem 9.2.1 (Central Limit Theorem (CLT), [Billingsley, 1986, pg 375]) *Suppose that $X_1, X_2, \ldots$ is stationary and α-mixing with $\alpha_n = O(n^{-5})$ and that $\langle X_n \rangle = 0$ and $\langle X_n^{12} \rangle < \infty$. If $S_n = X_1 + \cdots + X_n$, then*

$$\frac{Var[S_n]}{n} \to \sigma^2 = \left\langle X_1^2 \right\rangle + 2\sum_{k=1}^{\infty} \langle X_1 X_{1+k} \rangle \tag{9.27}$$

where the series converges absolutely. If $\sigma > 0$, then $S_n/\sigma\sqrt{n} \Rightarrow N(0,1)$ ■

where $N(0,1)$ is the standard normal distribution. Figure 9.2 illustrates this convergence to a Gaussian for $N_x \sim 100, N_y \sim 50$.

In general, the conditions of the theorem will be satisfied for $X_i = e_i$, so, the probability $P(\rho)$ approaches a Gaussian density. Therefore, to investigate $P(\rho)$ in the asymptotic limits $N_x, N_y \to \infty$ it suffices to calculate $\langle\rho\rangle$ and $Var[\rho]$. Then $N(\rho)$ is given by

$$N(\rho) = {N_y}^{N_x} P(\rho) \to \frac{1}{\sqrt{2\pi Var[\rho]}} \exp\left[-\frac{(\rho - \langle\rho\rangle)^2}{2Var[\rho]}\right] \tag{9.28}$$

In the remainder of this section we will list $\langle\rho\rangle$ and $Var[\rho]$ for different forms of ρ as $N_x, N_y \to \infty$. The details can be found in appendix 9.5.1.

1. $e_i = |s_i - s_{i-1}|$*(Total Variation Constraint* $\sim \mathcal{T}_{11}$*).*

$$\langle\rho\rangle \to \frac{N_x N_y}{3}, \qquad Var[\rho] \to \frac{N_x {N_y}^2}{15} \tag{9.29}$$

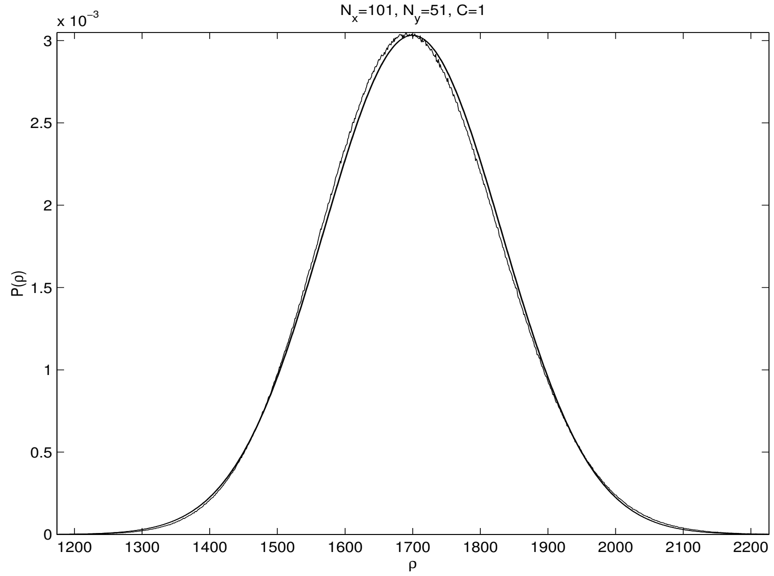


Illustration of the convergence of $N(\rho)$ to a Gaussian for $\rho = |s_i - s_{i-1}|$. Plotted are three! curves, one a montecarlo simulation to obtain $P(\rho)$, the second, a Gaussian with the same mean and variance of ρ that was obtained from the monte carlo simulation and the third is the Gaussian implied by the theoretically computed mean and variance of ρ . The fit is impressive given that N_x was only 101.

Figure 9.2: $N(\rho)$ and its Gaussian approximation

$N(\rho) = {N_y}^{N_x} N(\langle\rho\rangle\,, Var[\rho])$.

2. $e_i = (s_i - s_{i-1})^2$ *(Squared Total Variation Constraint* $\sim \mathcal{T}^2_{12}$*).*

$$\langle\rho\rangle \rightarrow \frac{N_x {N_y}^2}{6}, \qquad Var[\rho] \rightarrow \frac{N_x {N_y}^4}{20} \tag{9.30}$$

$N(\rho) = {N_y}^{N_x} N(\langle\rho\rangle\,, Var[\rho])$.

3. $e_i = (2s_i - s_{i-1} - s_{i+1})^2$ *(Smoothness Constraint* $\sim \mathcal{T}_{21}$*).*

$$\langle\rho\rangle \rightarrow \frac{N_x {N_y}^2}{2}, \qquad Var[\rho] \rightarrow \frac{121 N_x {N_y}^4}{180} \tag{9.31}$$

$N(\rho) = {N_y}^{N_x} N(\langle\rho\rangle\,, Var[\rho])$.

4. $e_i = g(s_i)$. In this case, $\langle \rho \rangle = N_x \langle g(s_0) \rangle$ and $Var[\rho] = N_x Var[g(s_0)]$

   (a) $g(s_i) = s_i/N_x$ *(Mean Constraint).*

$$\langle \rho \rangle \rightarrow \frac{N_y}{2}, \qquad Var[\rho] \rightarrow \frac{{N_y}^2}{12N_x} \tag{9.32}$$

   $N(\rho) = {N_y}^{N_x} N(\langle \rho \rangle, Var[\rho])$.

   (b) $g(s_i) = s_i^2/N_x$ *(Power Constraint).*

$$\langle \rho \rangle \rightarrow \frac{{N_y}^2}{6}, \qquad Var[\rho] \rightarrow \frac{{N_y}^4}{45N_x} \tag{9.33}$$

   $N(\rho) = {N_y}^{N_x} N(\langle \rho \rangle, Var[\rho])$.

5. $\rho = \max_i\{s_i\}$ *(Maximum Power Constraint)* To get $N(\rho)$, note that of the $(1+\rho)^{N_x}$ paths that are available, the ρ^{N_x} that do not reach the level ρ are excluded, so

$$N(\rho) = (\rho+1)^{N_x} - \rho^{N_x} \tag{9.34}$$

6. *Multiple Constraints* Suppose that the natural characteristic is a combination of properties of the path – a vector characteristic ($\vec{\rho}$). One needs to compute the multivariate probability distribution $P(\vec{\rho})$. Once again, in the fine grid limit, we can appeal to the CLT and reduce the computation to computing the parameters of the multivariate Gaussian. These are the mean vector ($\langle \vec{\rho} \rangle$) and the covariance matrix (Σ) given by

$$\Sigma = \left\langle \vec{\rho}\vec{\rho}^{\,T} \right\rangle - \langle \vec{\rho}\, \rangle \langle \vec{\rho}\, \rangle^T \tag{9.35}$$

Because $\Delta\rho = 1$, $N(\rho)$ is also the number density of functions at ρ, the number of functions with ρ in the interval $[\rho, \rho + \Delta\rho]$. It is this number density that is needed in what follows.

Characteristic	Discrete Version	$N(\rho)$	Limiting Form
$\int_0^1 dx\left\|\frac{dy}{dx}\right\|$	$\frac{C}{N_y}\sum_i \|s_i - s_{i-1}\|$	$N(\frac{N_x C}{3}, \frac{N_x C^2}{15})$	$\propto e^{\frac{5\rho}{C}}$
$\int_0^1 dx(\frac{dy}{dx})^2$	$N_x\left(\frac{C}{N_y}\right)^2\sum_i(s_i - s_{i-1})^2$	$N(\frac{{N_x}^2 C^2}{6}, \frac{{N_x}^3 C^4}{20})$	$\propto e^{\frac{10\rho}{3N_x C^2}}$
$\int_0^1 dx(\frac{d^2y}{dx^2})^2$	${N_x}^3\left(\frac{C}{N_y}\right)^2\sum_i(2s_i - s_{i-1} - s_{i+1})^2$	$N(\frac{{N_x}^4 C^2}{2}, \frac{121{N_x}^7 C^4}{180})$	$\propto e^{\frac{90\rho}{121{N_x}^3 C^2}}$
$\int_0^1 dx\|y\|$	$\left(\frac{C}{N_y}\right)\frac{1}{N_x}\sum_i s_i$	$N(\frac{C}{2}, \frac{C^2}{15N_x})$	$N(\frac{C}{2}, \frac{C^2}{15N_x})$
$\int_0^1 dx\ y^2$	$\left(\frac{C}{N_y}\right)^2\frac{1}{N_x}\sum_i s_i^2$	$N(\frac{C^2}{6}, \frac{C^4}{45N_x})$	$N(\frac{C^2}{6}, \frac{C^4}{45N_x})$
$\sup\{\|y\|\}$	$\frac{C}{N_y}\max_i\{s_i\}$	$\left(\frac{N_y\rho}{C}+1\right)^{N_x} - \left(\frac{N_y\rho}{C}\right)^{N_x}$	$\propto \rho^{N_x} e^{\frac{C}{\lambda\rho}}$ $\left(\frac{N_y}{N_x}\to\lambda\right)$

In the last column we give the limit as $N_x \to \infty$ and we assume that $P(\rho)$ (9.6) is non-zero only on some bounded region . Taking the log of this column gives the relevant "regularization" term.

Table 9.1: Various characteristics and the form for $N(\rho)$ that they lead to.

9.3 Application to Regularization

In the previous section, we discussed the computation of $N(\rho)$. Remember that the presumption is that ρ is a natural characteristic, so $P(f|\rho) = 1/N(\rho)$. ρ was generally a function of the s_i's. We are more interested in characteristics defined in terms of $y_i = \Delta_y s_i = Cs_i/N_y$. For example,

$$\int_0^1 dx\left|\frac{dy}{dx}\right| = \sum_i |y_i - y_{i-1}| = \frac{N_y}{C}\sum_i |s_i - s_{i-1}| \tag{9.36}$$

Table 9.1 gives some characteristics that might be of interest, and the corresponding discrete versions, along with $N(\rho)$, the number density of functions at ρ.

Given $N(\rho)$, we form the objective function to minimize using (9.5) by taking the logarithm of $N(\rho)$. Suppose that we assume that the data was created by taking the true targets and adding Gaussian noise, which we assume to have zero mean and variance σ^2. Then up to the addition of a constant, $\log[P(D|f)] = -1/2\sigma^2\sum_l r_l^2$ where we have defined the residuals $r_l = f(x_l) - y_l$, and l indexes the data points.

Up to constants, which we ignore, the optimization problem (9.5) reduces to

$$\min_f \left\{ \frac{1}{2\sigma^2} \sum_l r_l^2 + \log[N(\rho(f))] \right\} \tag{9.37}$$

where we have assumed that every value of ρ is equally likely. The second term can be considered a regularization term, and is given by the logarithm of the final column in table 9.1. If one has some other prior on the possible values of ρ (the natural characteristic) then this can be incorporated by using (9.6).

9.3.1 Derivation of the Tikhonov Regularizer

Many times, one wishes to enforce smoothness constraints on the learned function, so one heuristically introduces Tikhonov type regularization terms. Suppose that ρ is one of the smoothness measures. Then, if one would like to distinguish between target functions on the basis of ρ, and only ρ, one is making the statement that all target functions with the same value of ρ are equally likely, ie. that ρ is a natural characteristic. Thus we can apply the results from table 9.1 to get the form of the regularization term. We see that for all the smoothness characteristics, the dependence on ρ is given by $\exp(\lambda\rho)$, where λ can be read off from table 9.1. This leads to a regularization term $\lambda\rho$, which is precisely Tikhonov type regularization for the discrete setup we have described.

The Continuum Limit

Applying these methods to the continuum limit (though practically not necessary) could be interesting from the theoretical point of view. A little bit of care has to be used in passing to the continuum limit because for the derivatives to approach the true derivative, one needs $\Delta_y << \Delta_x$ (better resolution on the $y-$axis than the $x-$axis). We can define a precision to which we would like to measure the derivatives by the ratio Δ_y/Δ_x which we will assume tends to $1/\lambda$.

1. $\rho = \frac{C}{N_y}\sum_i |s_i - s_{i-1}| \to \mathcal{T}_{11}$. The regularization term is $5\mathcal{T}_{11}/C$. This is exactly the Tikhonov regularizer with regularization parameter equal to $5/C$.

2. $N_x\left(\frac{C}{N_y}\right)^2 \sum_i (s_i - s_{i-1})^2 \to \mathcal{T}_{12}$. The regularization term is $(10\mathcal{T}_{12}/3C^2)\lambda\Delta_y$. This is exactly the Tikhonov regularizer with regularization parameter equal to $(10/3C^2)\lambda\Delta_y$ (a similar result holds for $\mathcal{T}_{22}$–the regularization parameter is $(90/121C^2)\lambda^3{\Delta_y}^3$). Note that in these cases, the precision in the derivative and the $y-$quantization appear. We offer the following intuition. Regularization is found most useful in situations where noise is present, and one can view the quantization $\lambda\Delta_y = \Delta_x$ as "artificial noise", thus as Δ_x decreases, we expect the need for less regularization. In fact, [Bishop, 1995b] shows that training with input noise is equivalent to having a regularization term $\sim \mathcal{T}_{12}$.

9.4 Comments

The role of priors is important. Here we have discussed how to go from a particular prior (natural characteristic) to a regularization term by using counting arguments for paths on a grid. Given a problem, the aim should be to formulate a prior belief about what the most determining feature of the target function is, its natural characteristic (or set of natural characteristics), and then one can compute the relevant regularization term using a Bayesian framework as we have illustrated. We have shown that smoothness as a natural characteristic leads in general to Tikhonov type regularization terms, the minimization of an error function of the form

$$\mathcal{E} = \frac{1}{N}\sum_l r_l^2 + \frac{2\sigma^2\lambda}{N}\mathcal{T}_{ij} \tag{9.38}$$

where we factored out the $1/2\sigma^2$ in (9.37). We have given a prescription for the computation of λ, and have computed the regularization terms for a variety of natural characteristics (priors). Note that the term C that appears as a bound for the grid is also a prior.

We have addressed not what regularization terms one should use, but why regularization terms should arise naturally from the choice of a prior, and how to calculate them. Why one should use a certain regularization term is entirely a function of the problem at hand.

9.5 Appendix

9.5.1 $N(\rho)$ for various functional forms $\rho(f)$

1. $e_i = |s_i - s_{i-1}|$*(Total Variation Constraint)*$\sim \mathcal{T}_{11}$.

   For the expected value of ρ we have

$$\begin{aligned} \langle\rho\rangle &= \left\langle \sum_{i=1}^{N_x-1} |s_i - s_{i-1}| \right\rangle = \sum_{i=1}^{N_x-1} \langle |s_i - s_{i-1}| \rangle & (9.39) \\ &= (N_x - 1)\langle |s_1 - s_0| \rangle = \frac{N_x - 1}{{N_y}^2} \sum_{s_0, s_1 = 0}^{N_y - 1} |s_1 - s_0| & (9.40) \\ &= \frac{(N_x - 1)(N_y - 1)(N_y + 1)}{3N_y} & (9.41) \end{aligned}$$

   To compute the variance of ρ, first we will compute the expected value of ρ^2

$$\left\langle \rho^2 \right\rangle = \left\langle \sum_{i=1}^{N_x-1} \sum_{j=1}^{N_x-1} |(s_i - s_{i-1})(s_j - s_{j-1})| \right\rangle \tag{9.42}$$

   We need to consider four cases separately: $j = i$; $j = i + 1$;$j = i - 1$; neither of the above. Taking these four cases into consideration we have

$$\begin{aligned} \left\langle \rho^2 \right\rangle = \; & (N_x - 1)\left\langle (s_1 - s_0)^2 \right\rangle + 2(N_x - 2)\langle |(s_1 - s_0)(s_2 - s_1)| \rangle + \\ & ((N_x - 1)^2 - 3(N_x - 1) + 2)\langle |s_1 - s_0| \rangle^2 & (9.43) \end{aligned}$$

   We need to compute these expectations with respect to the probability distribution $P(s_0, s_1) = 1/{N_y}^2$. Then, $Var[\rho]$ is given by $\langle \rho^2 \rangle - \langle \rho \rangle^2$. A tedious calculation yields:

$$\begin{aligned} Var[\rho] = \; & \frac{(N_x - 1)(N_y - 1)(N_y + 1)({N_y}^2 + 1)}{15{N_y}^2} - \\ & \frac{(N_y - 1)(N_y - 2)(N_y + 1)(N_y + 2)}{90{N_y}^2} & (9.44) \end{aligned}$$

In the limit $N_x, N_y \to \infty$,

$$\langle \rho \rangle \to \frac{N_x N_y}{3}, \qquad Var[\rho] \to \frac{N_x {N_y}^2}{15} \tag{9.45}$$

2. $e_i = (s_i - s_{i-1})^2$ *(Squared Total Variation Constraint* $\sim \mathcal{T}_{12}$*)*

   Following similar techniques as above, we find

$$\begin{aligned} \langle \rho \rangle &= \sum_{i=1}^{N_x-1} \left\langle (s_i - s_{i-1})^2 \right\rangle && (9.46) \\ &= \frac{(N_x-1)(N_y-1)(N_y+1)}{6} && (9.47) \end{aligned}$$

   Once again, for the variance, we need to consider separately the four cases as above. A tedious calculation yields:

$$\begin{aligned} Var[\rho] &= \frac{(N_x-1)(N_y-1)(3(N_y-1)^3+12(N_y-1)^2+8(N_y-1)-8)}{60} \\ &\quad - \frac{N_y((N_y-1)^3+4(N_y-1)^2+(N_y-1)-6)}{90} && (9.48) \end{aligned}$$

   In the limit $N_x, N_y \to \infty$,

$$\langle \rho \rangle \to \frac{N_x {N_y}^2}{6}, \qquad Var[\rho] \to \frac{N_x {N_y}^4}{20} \tag{9.49}$$

3. $e_i = (2s_i - s_{i-1} - s_{i+1})^2$ *(Smoothness Constraint)* $\sim \mathcal{T}_{21}$

   Again, using similar techniques as above, we find

$$\begin{aligned} \langle \rho \rangle &= \frac{(N_x-2)(N_y-1)(N_y+1)}{2} && (9.50) \\ Var[\rho] &= \frac{(N_x-1)(N_y-1)(121(N_y-1)^3+484(N_y-1)^2+376(N_y-1)-216)}{180} \\ &\quad - \frac{N_y(181(N_y-1)^3+724(N_y-1)^2+556(N_y-1)-336)}{180} && (9.51) \end{aligned}$$

   In the computation of the variance, the six cases ($j = i$; $j = i \pm 1$;$j = i \pm 2$;

neither of the above) need to be handled separately. The limiting forms are

$$\langle \rho \rangle \rightarrow \frac{N_x {N_y}^2}{2}, \qquad Var[\rho] \rightarrow \frac{121 N_x {N_y}^4}{180} \tag{9.52}$$

4. $e_i = |2s_i - s_{i-1} - s_{i+1}|$ *(Smoothness Constraint* $\sim \mathcal{T}_{21}$*)* Though tedious, this can also be calculated using the above techniques.

Chapter 10
Conclusion

— In hope we can derive pleasure, even if those hopes turned out in vain.

Yaser S. Abu-Mostafa

We have answered some of the fundamental questions about learning systems regarding convergence, choice of regularization terms in the presence of noise, incorporation of prior information within a Bayesian framework and the prediction of noise parameters. We have also managed to fit the density estimation problem within the supervised learning framework and obtain consistent density estimation.

Buried in every chapter is the notion that if one has no prior information about the learning task, then one cannot choose learning systems in a systematic way so as to improve the chances of better test performance. Smooth learning models learn well because target functions tend to be smooth in general. The message is that one has to incorporate priors into the learning process and here we have developed techniques that illustrate where our prior information can be used.

- In choosing a learning system, there is a trade off between E_0 and $\mathcal{C}$, which depend on prior information about the targets.

- Choice of regularization in the presence of noise requires in addition to E_0 and $\mathcal{C}$, an estimate of the noise variance.

- Estimating noise distributions effectively requires a prior on the targets. Given a prior (knowledge of the target function), we could then analyze the maximum likelihood method for noise variance prediction.

- Given a prior in the form of a natural characteristic, it was possible to incorporate the prior into the learning process.

- The density estimation error was bounded in terms of the bounds on the derivatives of the true density, which in turn implied that the algorithm should be to minimize the bound of a certain derivative in addition to fitting the data.

The issue of the prior is important, and though we would like to avoid making assumptions, we have to do so in order to learn.

> *The longest period of time for which a modern painting has hung upside down in a public gallery unnoticed is 47 days. This occurred to* Le Bateau *by Matisse in the Museum of Modern Art, New York City. In this time 116,000 people had passed through the gallery.*
>
> [McWhirter and McWhirter, 1971]

Bibliography

[Abu-Mostafa and Atiya, 1996] Y. S. Abu-Mostafa and A. F. Atiya. Introduction to financial forecasting. *Applied Intelligence*, 6:205–213, 1996.

[Abu-Mostafa, 1990] Y. Abu-Mostafa. Learning from hints in neural networks. *Journal of Complexity*, 6:192–198, 1990.

[Abu-Mostafa, 1993] Y. S. Abu-Mostafa. Hints and the vc dimension. *Neural Computation*, 4:278–288, 1993.

[Abu-Mostafa, 1994] Y. S. Abu-Mostafa. Learning from hints. *Journal of Complexity*, 10:165–178, 1994.

[Abu-Mostafa, 1995] Y. Abu-Mostafa. Hints. *Neural Computation*, 4(7):639–671, 1995.

[Akaike, 1970] H. Akaike. Statistical predictor identification. *Annals of the Institute of Statistical Mathematics*, 22:203, 1970.

[Amari and Murata, 1991] S. Amari and N. Murata. Statistical theory of learning curves under entropic loss criterion. Technical Report METR 91-12, University of Tokyo, 1991.

[Amari *et al.*, 1995] S. Amari, N. Murata, K. R. Müller, and H. Yang. Asymptotic statistical theory of overtraining and cross validation. Technical Report METR 95-06, University of Tokyo, 1995.

[Atiya, 1994] Amir Atiya. On the required size of multilayer networks for implementing real-valued functions. In *Proc. IEEE World Congress on Computational Intelligence*, 1994.

[Baram and Roth, 1996] Y. Baram and Z. Roth. Multidimensional density shaping by sigmoids. *IEEE Transactions on Neural Networks*, 7(5):1291–1298, 1996.

[Barber and Williams, 1997] D. Barber and C. K. I. Williams. Gaussian processes for bayesian classification via hybrid monte carlo. In M. C. Mozer, M. I. Jordan, and

T. Petsche, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 9, pages 340–346. Morgan Kaufmann, 1997.

[Barron and Cover, 1991] A. Barron and T. Cover. Minimimum complexity density estimation. *IEEE Transactions on Information Theory*, 37(4):1034–1054, July 1991.

[Barron and Sheu, 1991] A. R. Barron and C-H Sheu. Approximation of density functions by sequences of exponential families. *Annals of Statistics*, 19(3):1347–1369, 1991.

[Barron *et al.*, 1992] A. Barron, L. Györfi, and E. van der Meulen. Distribution estimation consistent in total variation and in two types of information divergence. *IEEE Transactions on Information Theory*, 38(5):1437–1454, September 1992.

[Barron, 1984] A. Barron. Predicted square error: A criterion for automatic model selection. In S. Farlow, editor, *Self Organizing Methods in Modeling*. Marcel Dekker, New York, 1984.

[Barron, 1993] A. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, May 1993.

[Baum and Haussler, 1989] E. B. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1:151–160, 1989.

[Benveniste *et al.*, 1990] A. Benveniste, M. Métivier, and P. Priouret. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, New york, 1990.

[Bernardo, 1979] J. M. Bernardo. Reference posterior distributions for bayesian inference. *Journal of the Royal Statistical Society*, Ser. B(41):113–147, 1979.

[Bialek *et al.*, 1996] W. Bialek, C. G. Callan, and S. P. Strong. Field theories for learning probability distributions. *Physical Review Letters*, 77(23):4693–4697, 1996.

[Billingsley, 1986] P. Billingsley. *Probability and Measure*. Wiley Series in Probability and Mathematical Statistics. Wiley, 1986.

[Bishop, 1993] C. M. Bishop. Curvature-driven smoothing: A learning algorithm for feedforward networks. *Neural Computation*, 7(1):108–116, 1993.

[Bishop, 1995a] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[Bishop, 1995b] C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995.

[Black and Scholes, 1973] F. Black and M. S. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 3:637–654, 1973.

[Bollerslev, 1986] T. Bollerslev. Generalized autoregressive conditional heteroscedasticity. *Journal of Econometrics*, 31:307–327, 1986.

[Bretagnolle and Huber, 1979] J. Bretagnolle and C. Huber. Estimation des densités: Risque minimax. *Z. Wahrsch. Verw. Gebiete*, 47:119–137, 1979.

[Cataltepe *et al.*, 1998] Z. Cataltepe, Y. S. Abu-Mostafa, and M. Magdon-Ismail. No free lunch for early stopping. *To Appear in Neural Computation*, 1998.

[Chung, 1949] K. L. Chung. An estimate concerning the kolmogorov limit distribution. *Transactions of the American Mathematical Society*, 67:36–50, 1949.

[Clarke and Barron, 1994] B. S. Clarke and A. R. Barron. Jeffery's prior is asymptotically least favorable under entropy risk. *Journal of Statistical Planning and Inference*, 41:37–60, 1994.

[Clarke and Wasserman, 1993] B. Clarke and L. Wasserman. Noninformative priors and nuisance parameters. *American Statistical Association*, 88(424):1427–1432, 1993.

[Cortes *et al.*, 1994] C. Cortes, L. D. Jackel, and W-P. Chiang. Limits on learning machine accuracy imposed by data quality. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Neural Information Processing Systems (NIPS)*, volume 7, pages 239–246, 1994.

[Cover and Thomas, 1991] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley Series in Telecomunications. Wiley, 1991.

[Crain, 1973] B. R. Crain. A note on density estimation using orthogonal expansions. *Journal of the American Statistical Association*, 68(344):964–965, 1973.

[Crain, 1974] B. R. Crain. Estimation of distributions using orthogonal expansions. *Annals of Statistics*, 2:454–463, 1974.

[Crain, 1977] B. R. Crain. An information theoretic to approximating a probability distribution. *SIAM Journal of Applied Mathematics*, 32:339–346, 1977.

[Crouhy and Galai, 1995] M. Crouhy and D. Galai. Hedging with a volatility term structure. *The Journal of Derivatives*, Spring:45–52, 1995.

[Csáki, 1968] E. Csáki. An iterated logarithm law for semimartingales and its application to the empirical distribution function. *Studia Sci. Math. Hung.*, 3:287–292, 1968.

[Csiszár, 1975] I. Csiszár. I-divergence geometry of probability distributions and minimization problems. *Annals of Probability*, 3:146–158, 1975.

[Cwik and Koronacki, 1996] J. Cwik and J. Koronacki. Probability density estimation using a gaussian clustering algorithm. *Neural Computing & Applications*, 4(3):149–160, 1996.

[Cybenko, 1989] G. Cybenko. Approximation by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[Datta and Ghosh, 1996] G. S. Datta and M. Ghosh. On the invariance of noninformative priors. *The Annals of Statistics*, 24(1):141–159, 1996.

[DeGroot, 1989] M. H. DeGroot. *Probability and Statistics.* Addison–Wesley, Reading, Massachusetts, 1989.

[Devroye and Györfi, 1990] L. Devroye and L. Györfi. No empirical probability measure can converge in total variation sense for all distributions. *Annals of Statistics*, 18:1496–1499, 1990.

[Devroye *et al.*, 1996] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Aplications of Mathematics. Springer, New York, 1996.

[Devroye, 1985] L Devroye. *Non-Parametric Density Estimation: The L_1 View.* Wiley, New york, 1985.

[Devroye, 1986] L. Devroye. *Non Uniform Random Variate Generation.* Springer-Verlag, New york, 1986.

[Duda and Hart, 1973] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* J. Wiley & Sons, 1973.

[Durbin, 1973] J. Durbin. *Distribution Theory for Tests Based on the Sample Distribution Function.* Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, 1973.

[Efroimovich and Pinsker, 1983] S. Y. Efroimovich and M. S. Pinsker. Estimation of square integrable probability density of a random variable. *Problems Inform. Transmission*, 18:175–189, 1983.

[Efroimovich, 1985] S. Y. Efroimovich. Non-parametric estimation of a density of unknown smoothness. *Theory of Probability and its Applications*, 30(3):557–568, 1985.

[Efroimovich, 1986] S. Y. Efroimovich. Sequential non-parametric estimation of a density. *Theory of Probabability and its Applications*, 34(2):228–239, 1986.

[Engle, 1982] R.F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of u.k. inflation. *Econometrica*, 50:987–1008, 1982.

[Eubank, 1988] R. Eubank. *Spline Smoothing and Non-Parametric Regression.* Marcel Dekker, New York, 1988.

[Fama, 1965] E.E. Fama. The behavior of stock market prices. *Journal of Business*, 38:34–105, 1965.

[Feller, 1968a] W. Feller. *An Introduction to Probability Theory and its Applications*, volume 1. John Wiley & Sons, New york, 1968.

[Feller, 1968b] W. Feller. *An Introduction to Probability Theory and its Applications*, volume 2. John Wiley & Sons, New york, 1968.

[Finkelstein, 1971] H. Finkelstein. The law of the iterated logarithm for empirical distributions. *Annals of Mathematical Statistics*, 42(2):607–615, 1971.

[French *et al.*, 1987] K.R. French, G.W. Schwert, and R.F. Stambaugh. Expected stock returns and volatility. *Journal of Financial Economics*, 19:3–29, 1987.

[French, 1980] K.R. French. Stock returns and the weekend effect. *Journal of Financial Economics*, 8:55–69, 1980.

[Fukunaga and Hostetler, 1973] K. Fukunaga and L. D. Hostetler. Optimization of k-nearest neighbor density estimates. *IEEE Transactions on Information Theory*, 19(3):320–326, 1973.

[Funahashi, 1989] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[Gardiner, 1997] C. W. Gardiner. *Handbook of Stochastic Methods, 2nd. Edition.* Springer, New York, 1997.

[Geman and Bienenstock, 1992] S. Geman and E. Bienenstock. Neural networks and the bias variance dilemma. *Neural Computation*, 4:1–58, 1992.

[Gersho and Gray, 1992] A. Gersho and R. Gray. *Vector Quantization and Signal Compression.* Kluwer Scientific Publishers, 1992.

[Golub *et al.*, 1979] G. Golub, M. Heath, and G. Wahba. Generalized cross validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–224, 1979.

[Grenander and Szegö, 1958] U. Grenander and Szegö. *Toeplitz Forms and their Applications.* California Monographs in Mathematical Sciences. University of California Press, Los Angeles, 1958.

[Hamilton, 1994] J. D. Hamilton. *Time Series Analysis.* Princeton University Press, Princeton, New Jersey, 1994.

[Hertz *et al.*, 1991] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, CA, 1991.

[Hornik *et al.*, 1989] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[Hornik *et al.*, 1990] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3:551–560, 1990.

[Hornik *et al.*, 1994] K. Hornik, M. Stinchcombe, H. White, and P. Auer. Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives. *Neural Computation*, 6:1262–1275, 1994.

[Huang, 1987] K. Huang. *Statistical Mechanics, 2nd. edition.* John Wiley & Sons, New York, 1987.

[Hull and White, 1987] J. Hull and A. White. The pricing of options on assets with stochastic volatilities. *Journal of Finance*, 2:281–300, 1987.

[Hull, 1993] John C. Hull. *Options, Futures and other Derivative Securities* 2^{nd} *Ed.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[Ibrahim and Purushottam, 1991] J. G. Ibrahim and W. L. Purushottam. On bayesian analysis of generalized linear models. *American Statistical Association*, 86(416):981–986, 1991.

[Ishiguro and Sakamoto, 1984] M. Ishiguro and Y. Sakamoto. A bayesian approach to the probability density estimation. *Annals of the Institute of Statistical Mathematics*, 36, Part B:523–538, 1984.

[Ito, 1951] K. Ito. On stochastic differential equations. *Memoirs, American Mathematical Society*, 4:1–51, 1951.

[Ivanov, 1962] V. V. Ivanov. On linear problems which are not well posed. *Soviet Math. Docl.*, 3(4):981–983, 1962.

[Jeffreys, 1946] H. Jeffreys. An invariant form for the prior probability in estimation problems. In *Proceedings of the Royal Society of London*, volume Ser A., 196, pages 453–461, 1946.

[Johnson, 1994] G. Johnson. Construction of particular random processes. In *Proceedings IEEE*, volume 82(2), pages 270–281, February 1994.

[Kat, 1993] H.M. Kat. Replicating ordinary call options: A stochastic simulation study. In *13th AMEX Options and Derivatives Colloquium*, New York, 1993.

[Knuth, 1981a] D. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, 1981.

[Knuth, 1981b] D. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, 1981.

[Koistinen, 1997] P. Koistinen. Asymptotic theory for regularization: One-dimensional linear case. In *Advances in Neural Information Processing Systems (NIPS)*, volume 10, 1997.

[Krogh and Hertz, 1992a] A. Krogh and J. A. Hertz. Generalization in a linear perceptron in the presence of noise. *Journal of Physics A*, 25:1135–1147, 1992.

[Krogh and Hertz, 1992b] A. Krogh and J. A. Hertz. Learning with noise in a linear perceptron. *Journal of Physics A*, 25:1119–1133, 1992,.

[Kushner and Clark, 1978] H. Kushner and D. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems.* Springer-Verlag, 1978.

[Larsen and Marx, 1986] R. Larsen and M. Marx. *An Introduction to Mathematical Statistics.* Prentice-Hall, 1986.

[Lavielle, 1995] M. Lavielle. A stochastic algorithm for parametric and non-parametric estimation in the case of incomplete data. *Signal Processing*, 42:3–17, 1995.

[Leich and Tanner, 1991] G. Leich and J. E. Tanner. Economic forecast evaluation: Profit versus the conventional error measures. *American Economic Review*, 81:580–590, 1991.

[Ljung, 1977] L. Ljung. Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, 22(4):551–575, August 1977.

[Mackay, 1992a] D. J. C. Mackay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.

[Mackay, 1992b] D. J. C. Mackay. The evidence framework applied to classification networks. *Neural Computation*, 4:720–736, 1992.

[Mackay, 1992c] D. J. C. Mackay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472, 1992.

[Magdon-Ismail *et al.*, 1998] M. Magdon-Ismail, A. Nicholson, and Y. S. Abu-Mostafa. Financial markets: very noisy information processing. *submitted for publication, IEEE Special Issue on Information Processing*, 1998.

[Malkiel, 1985] B. Malkiel. *A Random Walk Down Wall Street.* W. W. Norton & Co., New York, 1985.

[Masry, 1994] E. Masry. Probability density estimation from dependent observations using wavelets orthonormal bases. *Statistics & Probability Letters*, 21:181–194, 1994.

[McWhirter and McWhirter, 1971] N. McWhirter and R. McWhirter. *The Guinness Book of Records.* Guinness Superlatives, Ltd., Enfield, Middlesex, 1971.

[Modha and Fainman, 1994] D. Modha and Y. Fainman. A learning law for density estimation. *IEEE Transactions on Neural Networks*, 5(3):519–523, May 1994.

[Momentum, 1996] Momentum. *Historical Market Data.* Traders Press, Inc., P.O. Box 6206, Greenville, South Carolina 29606, 1996.

[Montgomery *et al.*, 1990] D. Montgomery, L. Johnson, and J. Gardiner. *Forecasting and Time Series Analysis.* McGraw-Hill, New York, 1990.

[Moody, 1991] J. E. Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In John E. Moody, Steve J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 4, pages 847–854, 1991.

[Murata *et al.*, 1993] N. Murata, S. Yoshizawa, and S. Amari. Learning curves, model selection and complexity of neural networks. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 607–614. Morgan Kaufmann, 1993.

[Narendra and Parthasarathy, 1990] K. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[Nelson, 1991] D.B. Nelson. Conditional heteroscedasticity in asset returns: A new approach. *Econometrica*, 59:347–370, 1991.

[Nowlan and Hinton, 1992] S. Nowlan and G. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4(4):473–493, 1992.

[Parrondo and Van den Broeck, 1993] J. M. R. Parrondo and C. Van den Broeck. Vapnik-chervonenkis bounds for generalization. *Journal of Physics A: Math. Gen.*, 26:2211–2223, 1993.

[Parzen, 1962] E. Parzen. On the estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[Plaut *et al.*, 1986] D. Plaut, S. Nowlan, and G. Hinton. Experiments on learning by backpropagation. Technical Report CMU-CS-86-126, Carnegie Mellon University, 1986.

[Poterba and Summers, 1986] J. Poterba and L. Summers. The persistence of volatility and stock market fluctuations. *American Economic Review*, 76:1142–1151, 1986.

[Press *et al.*, 1988] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes.* Cambridge University Press, Cambridge, UK, 1988.

[Rissanen, 1992] J. Rissanen. Density estimation by stochastic complecity. *IEEE Transactions on Information Theory*, 38(2):315–323, 1992.

[Rudin, 1976] W. Rudin. *Principles of Mathematical Analysis.* International Series in Pure and Applied Mathematics. McGraw Hill, New York, 1976.

[Scalettar and Zee, 1988] R. Scalettar and A. Zee. Emergence of grandmother memory in feedforward networks: Learning with noise and forgetfulness. In D. Waltz and J. Feldman, editors, *Connectionist Models and Their Implications: Readings from Cognitive Science.* Ablex Pub. Corp, 1988.

[Schioler and Kulczyki, 1997] H. Schioler and P. Kulczyki. Neural network for estimating conditional distributions. *IEEE Transactions on Neural Networks*, 8(5):1015–1025, September 1997.

[Schwert, 1989] G.W. Schwert. Why does stock market volatility change over time. *Journal of Finance*, 44:1115–1153, 1989.

[Scott, 1988] D. W. Scott. *Multivariate Density Estimation, Theory, Practice and Visualization.* John Wiley & Sons, Inc., New York, 1988.

[Serfling, 1980] R. Serfling. *Approximation Theorems of Mathematical Statistics.* Wiley Series in Probability and Mathematical Statistics. Wiley, 1980.

[Shao and Tu, 1996] J. Shao and D. Tu. *The Jackknife and the Bootstrap.* Springer-Verlag, New York, 1996.

[Shiller, 1993] R. J. Shiller. *Market Volatility.* MIT Press, Cambridge, MA, 1993.

[Sibisi and Skilling, 1997] S. Sibisi and J. Skilling. Prior distributions on measure space. *Journal of the Royal Statistical Society B*, 59(1):217–235, 1997.

[Sill and Abu-Mostafa, 1997] J. Sill and Y. S. Abu-Mostafa. Monotonicity hints. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 9, pages 634–640. Morgan Kaufmann, 1997.

[Sill, 1998a] J. Sill. The capacity of monotonic functions. *Discrete Applied Mathematics*, Special Issue on VC Dimension, 1998.

[Sill, 1998b] J. Sill. Monotonic networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 10, 1998.

[Silverman, 1982] B. W. Silverman. On the estimation of a probability density function by the maximum penalized likelihood method. *The Annals of Statistics*, 10(3):795–810, 1982.

[Silverman, 1993] B. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, London, UK, 1993.

[Smirnov, 1944] N. V. Smirnov. An approximation to the distribution laws of random quantiles detremined by empirical data. *Uspehi Mat. Nauk*, 10:179–206, 1944.

[Smyth and Wolpert, 1998] P. Smyth and D. Wolpert. Stacked density estimation. In ****, editor, *Advances in Neural Information Processing Systems (NIPS)*, volume 10, page ****, 1998.

[Stone, 1990] C. J. Stone. Large sample inference for log-spline models. *Annals of Statistics*, 18(2):717–741, 1990.

[Tikhonov and Arsenin, 1977] A. N. Tikhonov and V. I. Arsenin. *Solutions of Ill-Posed Problems.* Scripta Series in Mathematics. Distributed solely by Halsted Press, Winston; New York, 1977. Translation Editor: Fritz, John.

[Tikhonov, 1963] A. N. Tikhonov. On solving ill-posed problems and the method of regularization. *Doklady Akademii Nauk USSR*, 153:501–504, 1963.

[Traven, 1991] H. Traven. A neural network approach to statistical pattern classification by semiparametric estimation of probability density functions. *IEEE Transactions on Neural Networks*, 2(3):366–377, May 1991.

[Trippi and Turban, 1993] R. R. Trippi and E. Turban. *Neural Networks in Finance and Investing.* Probos Publishing Company, Chicago, 1993.

[Valavanis, 1959] S. Valavanis. *Econometrics; an introduction to maximum likelihood methods.* McGraw-Hill, New York, 1959.

[van Hulle, 1996] m. van Hulle. Topographic map formation by maximizing unconditional entropy: a plausible strategy for on-line unsupervised competitive learning and nonparametric density estimation. *IEEE Transactions on Neural Networks*, 7(5):1299–1305, September 1996.

[van Trees, 1968] H. van Trees. *Detection, Estimation, and Modulation theory: Part I.* Wiley, New York, 1968.

[Vapnik and Chervonenkis, 1971] V. N. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

[Vapnik, 1982] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer Series in Statistics. Springer Verlag, New york, 1982.

[Vapnik, 1995] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer–Verlag, 1995.

[Weigend and Gershenfeld, 1993] A. S. Weigend and N. A. Gershenfeld, editors. *Time Series Prediction.* Santa Fe Institute, 1993.

[Weigend and Nix, 1995] A.S. Weigend and D.A. Nix. Learning local error bars for nonlinear regression. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 7, pages 489–496. Morgan Kaufmann, 1995.

[Weigend and Srivastava, 1995] A. Weigend and A. Srivastava. Predicting conditional probability distributions: a connectionist approach. *International Journal of Neural Systems*, 6(2):109–118, 1995.

[White, 1988] H. White. Economic prediction using neural networks: The case of IBM daily returns. In *IEEE International Conference on Neural Networks*, volume 2, pages 451–458, 1988.

[Wilks, 1963] S. S. Wilks. *Mathematical Statistics.* Wiley Publications in Statistics. John Wiley & Sons, New York, 1963.

[Wilmott *et al.*, 1995] P. Wilmott, S. Howison, and J. Dewynne. *The Mathematics of Financial Derivatives.* Cambridge University Press, 1995.

[Wolpert, 1996a] D. Wolpert. The mathematics of search. Technical Report SFI–TR–95–02–010, Santa Fe Institute, 1996.

[Wolpert, 1996b] D. H. Wolpert. The existence of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1391–1420, 1996.

[Wolpert, 1996c] D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[Wu and Moody, 1996] L. Wu and J. Moody. A smoothing regularizer for feedforward and recurrent neural networks. *Neural Computation*, 8(3):463–491, 1996.

[Zhu and Rohwer, 1996] H. Zhu and R. Rohwer. Bayesian regression filters and the issue of priors. *Neural Computation Applications*, 4:130–142, 1996.

[Zhu, 1996] H. Zhu. No free lunch for cross validation. *Neural Computation*, 8(7):1421–1426, 1996.

Index