Monotonicity and Connectedness in Learning Systems

Thesis by

Joseph Sill

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1998

(Submitted May 21, 1998)

Acknowledgements

Although I am listed as the author of this thesis, many other people should share the credit for its production. I am grateful to my advisor, Yaser Abu-Mostafa, for so many things I needed during my time here as a graduate student: technical expertise, wisdom, flexibility, structure, and abundant good cheer. I benefited a great deal from discussions with my fellow researchers: Zehra Cataltepe, Malik Magdon-Ismail, Alexander Nicholson, Xubo Song, Amir Atiya, and Eric Bax. I thank Malik in particular for help with some of the more technical aspect of the thesis and, more importantly, for being a great friend and roommate all these years. I would also like to thank the people in my life outside of work here in Pasadena for all their support, most notably Tasos, Parag, and Charla.

Above all, I thank my parents.

Abstract

This thesis studies two properties– monotonicity and connectedness– in the context of machine learning. The first part of the thesis examines the role of monotonicity constraints in machine learning from both practical and theoretical perspectives. Two techniques for enforcing monotonicity in machine learning models are proposed. The first method adds to the objective function a penalty term measuring the degree to which the model violates monotonicity. The penalty term can be interpreted as a Bayesian prior favoring functions which obey monotonicity. This method has the potential to enforce monotonicity only approximately, making it appropriate for situations where strict monotonicity may not hold. The second approach consists of a model which is monotonic by virtue of functional form. This model is shown to have universal approximation capabilities with respect to the class $\mathbf{M}$ of monotonic functions. A variety of theoretical results are also presented regarding $\mathbf{M}$. The generalization behavior of this class is shown to depend heavily on the probability distribution over the input space. Although the VC dimension of $\mathbf{M}$ is ∞, the VC entropy (i.e., the expected number of dichotomies) is modest for many distributions, allowing us to obtain bounds on the generalization error. Monte Carlo techniques for estimating the capacity and VC entropy of $\mathbf{M}$ are presented.

The second part of the thesis considers broader issues in learning theory. Generalization error bounds based on the VC dimension describe a function class by counting the number of dichotomies it induces. In this thesis, a more detailed characterization is presented which takes into account the diversity of a set of dichotomies in addition to its cardinality. Many function classes in common usage are shown to possess a property called *connectedness*. Models with this property induce dichotomy sets which are highly clustered and have little diversity. We derive an improvement to the VC bound which applies to function classes with the connectedness property.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

This thesis investigates certain issues in machine learning, so it will be prudent to begin with a brief sketch of what machine learning entails. We wish to design a system which makes accurate predictions about an unknown quantity y when supplied with a vector of information $\mathbf{x}$ (referred to throughout this work as an *input vector* or *feature vector*). For instance, $\mathbf{x}$ could be a numerical profile (salary, age, etc.) of a credit applicant and y could be a binary variable indicating whether or not the applicant will default if extended credit. $\mathbf{x}$ and y are generated from some unknown joint distribution $P(\mathbf{x}, y)$ over a product space $X \times Y$. There exists some unknown *target function* $t(\mathbf{x})$ which does the best possible job of predicting y given $\mathbf{x}$. The optimality criterion can be defined many ways depending upon the application. Unless otherwise noted, we will assume the most common definitions of optimality. For applications where $Y = \{0, 1\}$, we take $t(\mathbf{x})$ to be the function which minimizes the probability of error

$$t(\mathbf{x}) = \min_f Pr\{y \neq f(\mathbf{x})\}$$

while for continuous output problems we define $t(\mathbf{x})$ to minimize expected squared error

$$t(\mathbf{x}) = \min_f \mathcal{E}[(y - f(\mathbf{x}))^2].$$

We are supplied with a *training set* of n examples $(\mathbf{x^{(1)}}, y_1) \ldots (\mathbf{x^{(N)}}, y_N)$ drawn independently from $P(\mathbf{x}, y)$. The $y_n's$ may be viewed as instances of t, possibly corrupted by noise. In the credit screening task, the examples would correspond to historical records of past applicants consisting of profiles and labels indicating whether or not the loan was repaid. We would like to use the examples to help us identify a good approximation to $t(\mathbf{x})$.

Some readers familiar with machine learning might object to this last sentence as an overly modest characterization of the field. They would argue that examples

do not merely *aid* our search for t. Indeed, the goal of machine learning is often portrayed as the inference of a suitable function t *solely* on the basis of examples, i.e., without any knowledge or assumptions about the problem. Recent developments in learning theory, however, have crystallized a certain fundamental fact: "assumption-free" learning is an ill-posed problem and a hopeless situation. With a little thought, this is actually not hard to see. A set of examples contains information only about the value of t at certain points in the input space X. Even in the best case, when no noise is present, all the examples tell us is the exact value of t at certain points. If there are no constraints on t, then knowing the value of t at certain points in the input space implies nothing at all about the value of t at other points in the space. Wolpert has formalized this observation in what is known as the "No Free Lunch" theorem [Wolpert, 1996]. Wolpert's theorem applies to discrete output spaces Y where performance is evaluated in terms of the probability of being correct. Suppose we are working in a learning environment where the target function t to be learned is generated from a distribution over all possible targets. Suppose we evaluate a learning method in terms of its expected out-of-sample performance, where the expectation is taken with respect to the distribution over targets. Roughly speaking, the No Free Lunch theorem states that if the target distribution is uniform, then the expected performance of all methods is equal. This result is quite intuitive given the observation that knowing only training data and nothing else about t, any value of t on an out-of-sample input vector is just as likely as any other. The No Free Lunch theorem essentially tells us that no learning algorithm is universally or inherently good. We can distinguish between the merits of different methods only if we have some information about the nature of the target functions which arise in a particular context.

A machine learning practitioner therefore has no choice but to make some assumptions about the problem and hope that they are at least roughly true. The most common way that assumptions are made is by restricting our search to a particular class of functions F (F will also be referred to at times as the *model*). The choice of F should be governed by beliefs we have about the desired function t. If we know that $t(\mathbf{x})$ possesses certain properties, F should be chosen to contain functions with

these characteristics. In many cases, we may not have much firm knowledge of t but may still have general biases about the properties that tend to arise in applications, e.g., smoothness, mild nonlinearity, etc. In such situations, we may cross our fingers and choose F to have the properties we suspect of t.

The design of F must balance two competing concerns. The larger and more flexible F is, the more likely it is that it contains a good approximation to t. On the other hand, the higher the number of degrees of freedom we give F, the more poorly its parameters will be estimated, assuming our budget of data is fixed in size. This dilemma is known as the *bias-variance* tradeoff [Geman *et al.*, 1992]. Let f be a hypothesis function chosen (usually by minimizing error on the training set) from F. We can consider f to be a random variable with respect to the process of randomly drawing a set of N examples from $P(\mathbf{x}, y)$. Let $\hat{f}$ denote the expectation of f with respect to this process. The expected out-of-sample error of f is given by

$$E_F(N) = \mathcal{E}[(f(\mathbf{x}) - y)^2]$$

The expectation is taken with respect to a random training set of size N and a random test point $\mathbf{x}, y$. It is not hard to show that E_F can be decomposed into three terms:

$$\mathcal{E}[(y - t(\mathbf{x}))^2] + \mathcal{E}[(t(\mathbf{x}) - \hat{f}(\mathbf{x}))^2] + \mathcal{E}[(f(\mathbf{x}) - \hat{f}(\mathbf{x}))^2]$$

$\mathcal{E}[(y - t(\mathbf{x}))^2]$ measures the intrinsic noise in the problem and remains constant no matter what function class is chosen. The two factors we can influence by our choice of F are the *bias* $\mathcal{E}[(t(\mathbf{x}) - \hat{f}(\mathbf{x}))^2]$ and the *variance* $\mathcal{E}[(f(\mathbf{x}) - \hat{f}(\mathbf{x}))^2]$. The bias can be thought of as a measure of how close the closest functions in F come to approximating t. The variance measures the variability of f resulting from the finiteness of the random training set. Typically, adding more free parameters to F will decrease bias but increase variance, while adding restrictions to F will do the opposite. Increasing the training set size usually leaves the bias unaffected but decreases the variance.

The bias-variance formula shows that proper choice of F is not simply a matter of making sure that F is broad and flexible enough that it is sure to contain an exact implementation of t. A more restricted class, capable of only approximating t, may nonetheless have lower expected squared error– the increase in bias may be more than offset by the decrease in variance. This is particularly likely in situations where the training set is small. In order to make sure that the bias is not too large, however, it is crucial to take advantage of all available information about t. Such information allows us to choose a class F which is restricted in flexibility yet has a good chance of approximating t decently.

The No-Free-Lunch theorem and the bias-variance formula provide two powerful theoretical motivations to capitalize on prior information. The incorporation of prior knowledge into the learning process is the motivation behind the first two chapters of the thesis. Each of these chapters presents a technique for enforcing monotonicity constraints in learning models. Such techniques allow us to design models appropriate for situations where we suspect that target function is monotonically increasing in some or all input variables. We formalize the notion of monotonicity as follows.

Consider a pair of input vectors $(\mathbf{x}, \mathbf{x}') \in X$ such that

$$\forall j \neq i, x'_j = x_j \tag{1}$$

$$x'_i > x_i \tag{2}$$

The statement that f is monotonically increasing in input variable x_i means that for all such $\mathbf{x}, \mathbf{x}'$ defined as above

$$f(\mathbf{x}') \geq f(\mathbf{x}) \tag{3}$$

Decreasing monotonicity is defined similarly.

There are a variety of applications where there is good reason to think that the target function is monotonic in some or all of the input variables. Monotonicity con-

straints often suggest themselves in the context of business and finance. For instance, when screening credit card applicants, one might expect creditworthiness to be monotonically increasing in salary but monotonically decreasing in debt. Medical diagnosis is another area in which monotonicity arises. In many situations, the probability of the presence of a disease would be expected to rise monotonically with a particular symptom measurement.

Chapter 2 suggests adding a penalty function measuring "monotonicity error" to the traditional training error measure to form an objective function which strikes a balance between fitting the training data and obeying the monotonicity constraint. The emphasis on the penalty term can be varied in terms of how strong our bias towards monotonicity is. This allows us to enforce monotonicity either quite strictly or only approximately. The method is compatible with a variety of models. Chapter 3 presents a class of models which are exactly monotonic, i.e., monotonic by virtue of functional form. This class is shown to be capable of uniformly approximating any continuous, differentiable monotonic function to an arbitrary degree of accuracy. Real datasets from economic and medical domains are used to test these methods and to compare their performances with each other and with competing techniques.

Chapters 4 and 5 are more theoretical in nature, focusing on the issue of *generalization.* The question of generalization is the following: When can we be confident that our model has truly learned and not simply memorized the examples it has been given? Most learning algorithms attempt to find a candidate function $f \in F$ which has low error on the training set. How many examples do we need in order to be confident that f will continue to perform well on new input vectors not appearing in the training set? Clearly, the answer depends in some way on the flexibility or the degrees of freedom of F. Suppose we wish to learn a real-valued function of one variable and we are given N data points $(x^{(1)}, y_1) \ldots\ldots (x^{(N)}, y_N)$. We can fit the data perfectly using an $(N-1)$-dimensional polynomial, yet it would be foolish to expect that the resulting polynomial would be useful in predicting the output y at other points along the real line. This is so precisely because we are guaranteed to be able to fit the data perfectly, no matter how it was generated. Even if x and y were generated from

independent distributions , i.e., if x contained no information about y, we would still obtain a perfect fit. Therefore, we know that good in-sample performance cannot be a guarantee in and of itself of good out-of-sample performance. On the other hand, if N is large and we find that a linear model fits the in-sample data quite accurately, then we have good reason to believe that the model does a good job of interpolating between the data points. It is unlikely that a linear model will be able to fit a large number of examples unless the process underlying the generation of the examples is actually roughly linear. Hence, the fact that a good fit is obtained is evidence that a linear model captures the relationship between x and y.

Another way to understand the issue of generalization is to contrast learning with simple parameter estimation. Suppose we are working on a binary classification problem and we pick f in such a way that is not influenced at all by the training set. Then basic probability theory tells us that the frequency ν_f of errors on the training set converges to the true probability of error π_f. For instance, Hoeffding's inequality [Hoeffding, 1956] states that

$$Pr\{|\nu_f - \pi_f| > \epsilon\} \leq 2e^{-2\epsilon^2 N}$$

This result cannot be applied, however, in the situation where f has been chosen from F on the basis of having low error on the training set. Here, the training set error is a biased estimate of the true error. These two situations can be compared to coin flipping experiments. If we flip a single coin 10 times and achieve 10 heads, then we can say with some confidence that the coin is unfair. Suppose, however, we flip 1000 coins ten times each and find one coin which comes up heads every time. Now we are considerably less confident about the biasedness of this coin than we were about the first coin. Under the assumption that all the coins are fair, there is a good chance (about 62%, actually) that at least one coin will come up heads 10 times just by sheer chance. This sort of event is what we are worried in the context of machine learning. If F is very flexible and contains a wide variety of functions, then there may be a decent chance that some function in F will have a low training error merely by

chance, while having a true error rate which is unacceptably high. How can we be confident that this sort of event will not occur? We must have enough examples that the error frequency for each function in F converges to the function's true error rate *uniformly* over F. In other words, we require

$$Pr\{\sup_{f \in F} |\nu_f - \pi_f| > \epsilon\} \leq \delta$$

where ϵ and δ are small positive constants.

One might expect that the number of free parameters is the quantity relevant to the determination of the number of examples required for uniform convergence. The number of parameters turns out to be not quite the right measure of flexibility, however. The quantity of interest is the number of *dichotomies* induced by the function class. A dichotomy is simply an N-dimensional binary vector

$$\mathbf{d} = [d_1\ d_2\ \ldots d_N],\quad d_i \in \{0, 1\}$$

We say that a function class F induces a dichotomy $\mathbf{d}$ on a sample of input vectors $\{\mathbf{x^{(1)}},\ \mathbf{x^{(2)}}\ \ldots\ \mathbf{x^{(N)}}\}$ if

$$\exists\ f \in F :\ \forall\ i\ f(\mathbf{x^{(i)}}) = d_i$$

The number of dichotomies $\Delta_F(\mathbf{x^{(1)}},\ \mathbf{x^{(2)}}\ \ldots\ \mathbf{x^{(N)}})$ induced by F has proven to be a very useful measure in the development of bounds on the number of examples required for uniform convergence.

Let

$$H_F(N) = \mathcal{E}[\Delta_F(\mathbf{x^{(1)}},\ \mathbf{x^{(2)}}\ \ldots\ \mathbf{x^{(N)}})]$$

where the expectation is taken over the product distribution resulting from drawing N vectors independently from the input distribution $P(\mathbf{x})$. Define

$$G_F(N) = \max_{\{\mathbf{x^{(1)}},\ \mathbf{x^{(2)}}\ \ldots\ \mathbf{x^{(N)}}\}} \Delta_F(\mathbf{x^{(1)}},\ \mathbf{x^{(2)}}\ \ldots\ \mathbf{x^{(N)}})$$

We refer to $H_F(N)$ as the *VC entropy* of F and $G_F(N)$ as the *growth function* of

F [1]

In their landmark 1971 paper, Vapnik and Chervonenkis [Vapnik and Chervonenkis, 1971] derive uniform convergence bounds in terms of these two quantities.

$$Pr\{\sup_{f\in F}|\nu_f - \pi_f| > \epsilon\} < 4H_F(2N)e^{-\frac{\epsilon^2 N}{8}} \leq 4G_F(2N)e^{-\frac{\epsilon^2 N}{8}}$$

The bounds have since been tightened by Parrondo and Van den Broeck [Parrondo and Van den Broeck, 1993]:

$$Pr\{\sup_{f\in F}|\nu_f - \pi_f| > \epsilon\} < 6H_F(2N)e^{-(\epsilon-\frac{1}{N})^2 N} \leq 6G_F(2N)e^{-(\epsilon-\frac{1}{N})^2 N}$$

Motivated by these results, theorists in machine learning devote a substantial amount of effort to the development of bounds on the number of dichotomies implemented by various function classes. Most of the energy is directed towards bounding $G_F(N)$, the *growth function.* This focus can be explained by a couple of factors. A bound on $G_F(N)$ is useful because it can be employed in bounds which hold over all possible input distributions. In addition to its universal applicability, another appealing property of $G_F(N)$ is the fact that it can be summarized in a single parameter, called the *VC dimension* of F. The VC dimension v is defined as the largest value of N for which $G_F(N) = 2^N$. It turns out that the growth function can be bounded by $(\frac{eN}{v})^v$. This polynomial growth guarantees that for any finite v and any nonzero ϵ,

$$\lim_{N\to\infty} Pr\{\sup_{f\in F}|\nu_f - \pi_f| > \epsilon\} = 0$$

The attraction of these two characteristics of the growth function has resulted in a nearly total emphasis on distribution-independent results. One of the contributions of this thesis, however, is the presentation of a situation of practical importance where it is critical to take the input distribution into account. We accomplish this by con-

[1] Here we use the traditional definiton of the growth function, the one employed in [Vapnik, 1982] and other well known texts such as [Hertz *et al.*, 1991]. In Vapnik's more recent work [Vapnik, 1995], he has defined the growth function as $ln(\max\Delta_F(\mathbf{x^{(1)}}, \mathbf{x^{(2)}} \ldots \mathbf{x^{(N)}}))$. $ln(\mathcal{E}[\Delta_F(\mathbf{x^{(1)}}, \mathbf{x^{(2)}} \ldots \mathbf{x^{(N)}})])$ is referred to in this work as the annealed VC entropy. We find it more convenient to work without the logarithm, and we omit the term *annealed* for brevity.

tinuing our investigation of the monotonicity constraint, this time from a theoretical perspective. Chapter 4 presents an analysis of the class **M** of functions which are monotonically increasing in all input variables. The decision boundaries of monotonic classifiers are characterized and contrasted with those of linear classifiers. **M** is then analyzed in the context of VC theory. We show that distribution-independent analysis fails in this case. It is shown that the VC dimension of **M** is ∞, despite the fact that **M** is powerfully constrained. For most reasonable input distributions, however, the VC entropy of **M** is modest. Monte Carlo techniques are presented for estimating $H_{\mathbf{M}}(N)$ given a model of the input distribution.

Chapter 5 departs from the monotonicity theme and considers broader issues in learning theory. The chapter shows that both $G_F(N)$ and $H_F(N)$ are inadequate characterizations of the expressive power of a function class. The VC inequalities simply count the number of dichotomies a function class induces. A richer description of the flexibility of a function class is possible by considering the diversity of a set of dichotomies in addition to its cardinality. It is shown that many of the learning models employed in practice induce dichotomy sets which are clustered together and have little diversity. By taking this additional characteristic into account, we derive an improvement to the existing VC inequalities which applies to many models used by machine learning practitioners.

Chapter 2 Monotonicity Hints

2.1 Introduction

Researchers in pattern recognition, statistics, and machine learning often draw a contrast between linear models and flexible nonlinear methods. Linear models make very strong assumptions about the function to be modelled. In contrast, many approaches (e.g., k-nearest neighbors, kernel estimators, neural networks) try to make as few assumptions as possible and are attempts to capture arbitrary nonlinear relationships. Between these two extremes, there exists a frequently neglected middle ground of nonlinear models which incorporate strong prior information and obey powerful constraints.

A monotonic model is one example which might occupy this middle area. We argued in the introduction that many machine learning applications arise in which there is good reason to believe the target function is monotonic in some or all input variables. It would be very useful, therefore, to be able to constrain a nonlinear model to obey monotonicity.

The rare previous efforts at developing monotonic models have met with limited success. Archer and Wang [Archer and Wang, 1993] present an algorithm for enforcing monotonicity in two-layer neural networks by an adjustment to the on-line backpropagation algorithm. After each weight update, it is verified that the model is still monotonic. If monotonicity has been violated, then the learning rate for that pattern is decreased until the update no longer causes a violation. This technique restricts the user to one particular model (two-layer neural networks) and one particular optimization technique (on-line backpropagation), each of which may or may not be appropriate for a particular problem. In addition, because the constraint enforcement is coupled together with the optimization, it is unclear whether their method will necessarily find the best possible monotonic fit given the network. Mukarjee and

Stern [Mukarjee and Stern, 1994] present a technique which transforms a standard kernel estimator into a monotonic estimator. Unfortunately, their method is only applicable to cases where monotonicity must be enforced in all the variables. It also suffers from the usual drawbacks of kernel estimators, such as high costs in memory and execution time. Indeed, their method is particularly costly to execute and scales very poorly with the number of inputs because it requires a grid search over the input space.

In contrast, we present the idea of adding a penalty term quantifying a model's violation of monotonicity to the traditional objective function measuring training error. This method provides several advantages over previously developed alternatives. Our method is compatible with any parametrized model and any optimization technique. Monotonicity can be enforced in some inputs while leaving the others unconstrained. In addition to these merits, the characteristic which most distinguishes this method is the ability to adjust the extent to which monotonicity is enforced according to the user's beliefs about the target function. In some applications, the target function may be thought to be only roughly monotonic. The penalty term approach allows this belief to be reflected in practice more accurately than it would be were monotonicity enforced precisely.

The general framework for incorporating prior information into learning via penalty terms is well established and is known as learning from hints [Abu-Mostafa, 1990]. A hint is any piece of information about the target function beyond the available input-output examples. Hints can improve the performance of learning models by reducing capacity without sacrificing approximation ability [Abu-Mostafa, 1993]. Invariances in character recognition [Simard *et al.*, 1993] and symmetries in financial-market forecasting [Abu-Mostafa, 1995] are some of the hints which have proven beneficial in real-world learning applications. This chapter describes the first practical applications of monotonicity hints. The method is tested on three noisy real-world problems: a classification task concerned with credit card applications and two regression problems, one in medical diagnosis and the other in bond rating prediction.

Section 2.2 derives, from Bayesian principles, an appropriate objective function for

simultaneously enforcing monotonicity and fitting the data. Section 2.3 describes the details and results of the experiments. Section 2.4 analyzes the results and discusses possible future work.

2.2 Bayesian Interpretation of Objective Function

We wish to define a single scalar measure of the degree to which a particular candidate function f obeys monotonicity in a set of input variables.

One such natural measure, the one used in the experiments in Section 2.3, is defined in the following way: Let $\mathbf{x}$ be an input vector drawn from the input distribution. Let i be the index of an input variable randomly chosen from a uniform distribution over those variables for which monotonicity holds. Define a perturbation distribution, e.g., U[0,1], and draw δx_i from this distribution. Define $\mathbf{x}'$ such that

$$\forall j \neq i, x'_j = x_j \tag{4}$$

$$x'_i = x_i + sgn(i)\delta x_i \tag{5}$$

where $sgn(i) = 1$ or -1 depending on whether the target function is thought to be monotonically increasing or decreasing in variable i. We will call E_h the *monotonicity error* of f on the input pair $(\mathbf{x}, \mathbf{x}')$.

$$E_h = \begin{cases} 0 & f(\mathbf{x}') \geq f(\mathbf{x}) \\ (f(\mathbf{x}) - f(\mathbf{x}'))^2 & f(\mathbf{x}') < f(\mathbf{x}) \end{cases} \tag{6}$$

Our measure of f's violation of monotonicity is $\mathcal{E}_f[E_h]$, where the expectation is taken with respect to random variables $\mathbf{x}$, i and δx_i.

We believe that the best possible approximation to the target function given the architecture used is probably approximately monotonic. This belief may be quantified in a prior distribution over the candidate functions implementable by the architecture:

$$Pr(f) \propto e^{-\lambda \mathcal{E}_f[E_h]} \tag{7}$$

This distribution represents the *a priori* probability density, or likelihood, assigned to a candidate function with a given level of monotonicity error. The probability that a function is the best possible approximation to the target function decreases exponentially with the increase in monotonicity error. λ is a positive constant which indicates how strong our bias is towards monotonic functions.

In addition to obeying prior information, the model should fit the data well. For classification problems, we take the network output f to represent the probability of class $c = 1$ conditioned on the observation of the input vector (the two possible classes are denoted by 0 and 1). We wish to pick the most probable model given the data. Equivalently, we may choose to maximize $log(P(model|data))$. Using Bayes' Theorem,

$$log(P(model|data)) \propto log(P(data|model) + log(P(model)) \tag{8}$$

$$= \sum_{n=1}^{N} c_n log(f(\mathbf{x^{(n)}}) + (1 - c_n) log(1 - f(\mathbf{x^{(n)}}) - \lambda \mathcal{E}[E_h] \tag{9}$$

For continuous-output regression problems, we interpret f as the conditional mean of the observed output t given the observation of $\mathbf{x}$. If we assume constant-variance gaussian noise, then by the same reasoning as in the classification case, the objective function to be maximized is :

$$-\sum_{n=1}^{N} (f(\mathbf{x^{(n)}}) - t_n)^2 - \lambda \mathcal{E}[E_h] \tag{10}$$

The Bayesian prior leads to a familiar form of objective function, with the first term reflecting the desire to fit the data and a second term penalizing deviation from monotonicity.

2.3 Experimental Results

2.3.1 Background

Two of the three datasets used for experimentation are publicly available. The credit card and medical diagnosis databases were obtained via FTP from the machine learning database repository maintained by UC-Irvine [1]

The credit card task is to predict whether or not an applicant will default. For each of 690 applicant case histories, the database contains 15 features describing the applicant plus the class label indicating whether or not a default ultimately occurred. The meaning of the features is confidential for proprietary reasons. Only the 6 continuous features were used in the experiments reported here. 24 of the case histories had at least one feature missing. These examples were omitted, leaving 666 which were used in the experiments. The two classes occur with almost equal frequency; the split is 55%-45%.

Intuition suggests that the classification should be monotonic in the features. Although the specific meanings of the continuous features are not known, we assume here that they represent various quantities such as salary, assets, debt, number of years at current job, etc. Common sense dictates that the higher the salary or the lower the debt, the less likely a default is, all else being equal. Monotonicity in all features was therefore asserted.

The second application the method was tested on was the prediction of corporate bond ratings. Rating agencies such as Standard & Poors (S & P) issue bond ratings intended to assess the level of risk of default associated with the bond. S & P ratings can range from AAA down to B- or lower.

A model which accurately predicts the S & P rating of a bond given publicly available financial information about the issuer has considerable value. Rating agencies do not rate all bonds, so an investor could use the model to assess the risk associated

[1] They may be obtained as follows: ftp ics.uci.edu. cd pub/machine-learning-databases. The credit data is in the subdirectory /credit-screening, while the liver data is in the subdirectory /liver-disorders.

with a bond which S & P has not rated. The model can also be used to anticipate rating changes before they are announced by the agency.

The dataset, which was donated by a Wall Street firm, is made up of 196 examples. Each training example consists of 10 financial ratios reflecting the fundamental characteristics of the issuing firm, along with an associated rating. The meaning of the financial ratios was not disclosed by the firm for proprietary reasons. The rating labels were converted into integers ranging from 1 to 16. The task was treated as a single-output regression problem rather than a 16-class classification problem.

Monotonicity constraints suggest themselves naturally in this context. Although the meanings of the features are not revealed, it is reasonable to assume that they consist of quantities such as profitability, debt, etc. It seems intuitive that, for instance, the higher the profitability of the firm is , the stronger the firm is, and hence, the higher the bond rating should be. Monotonicity was therefore enforced in all input variables.

The motivation in the medical diagnosis problem is to determine the extent to which various blood tests are sensitive to liver disorders related to excessive drinking. Specifically, the task is to predict the number of drinks a particular patient consumes per day given the results of 5 blood tests. 345 patient histories were collected, each consisting of the 5 test results and the daily number of drinks.

The justification for monotonicity in this case is based on the idea that an abnormal result for each test is indicative of excessive drinking, where abnormal means either abnormally high or abnormally low.

Some readers may be skeptical that monotonicity is strictly satisfied by the target function in these applications. It is important to note that a monotonic model may be a good option even if the target function is not purely monotonic. In such cases, the increase in bias introduced by the monotonicity constraint may be outweighed by the decrease in variance. This is particularly true if the training set is limited in size and noisy.

2.3.2 Experimental Details

In all experiments, batch-mode backpropagation with a simple adaptive learning rate scheme was used [2]. Several methods were tested. The performance of a linear perceptron was observed for benchmark purposes. For the experiments using nonlinear methods, a single hidden layer neural network with direct input-output connections was used. Two methods of training the nonlinear methods were employed. One approach used simply minimized the training error as much as possible. A more sophisticated method known as early stopping (described below) was also applied. Training for all of the above models was performed by maximizing only the first term in the objective function, i.e., by maximizing the log-likelihood of the data (minimizing training error). Finally, training the networks with the monotonicity constraints was performed, using an approximation to (9) and (10).

A leave-k-out procedure was used in order to get statistically significant comparisons of the difference in performance. For each method, the data was randomly partitioned 200 different ways (The split was 550 training, 116 test for the credit data; 150 training, 46 test for the bond rating problem, and 270 training and 75 test for the liver data). The results shown in the tables are averages over the different partitions.

In the early stopping experiments, the training set was further subdivided into a set (450 for the credit data, 110 for the bond data, 200 for the liver data) used for direct training and a second validation set (100 for the credit data, 40 for the bond data, 70 for the liver data). The classification error on the validation set was monitored over the entire course of training, and the values of the network weights at the point of lowest validation error were chosen as the final values.

The process of training the networks with the monotonicity hints was divided into two stages. Since the meanings of the features were unaccessible, the directions of monotonicity were not known *a priori*. These directions were determined by training a linear perceptron on the training data for 300 iterations and observing the result-

[2] If the previous iteration resulted in a increase in likelihood, the learning rate was increased by 3%. If the likelihood decreased, the learning rate was cut in half

ing weights. A positive weight was taken to imply increasing monotonicity, while a negative weight meant decreasing monotonicity.

Once the directions of monotonicity were determined, the networks were trained with the monotonicity hints. For the credit problem, an approximation to the theoretical objective function (10) was maximized:

$$\sum_{n=1}^{N} c_n log(f(x^(n)) + (1 - c_n) log(1 - f(x^(n)) - \frac{\lambda}{L} \sum_{l=1}^{L} E_{h,n} \tag{13}$$

For the regression problems, objective function (12) was approximated by

$$-\sum_{n=1}^{N} (f(x^(n)) - t_n)^2 - \frac{\lambda}{L} \sum_{L=1}^{L} E_{h,n} \tag{14}$$

$E_{h,l}$ represents the network's monotonicity error on a particular pair of input vectors $\mathbf{x}, \mathbf{x}'$. Each pair was generated according to the method described in Section 2.2. The input distribution was modelled as a joint gaussian with a covariance matrix estimated from the training data.

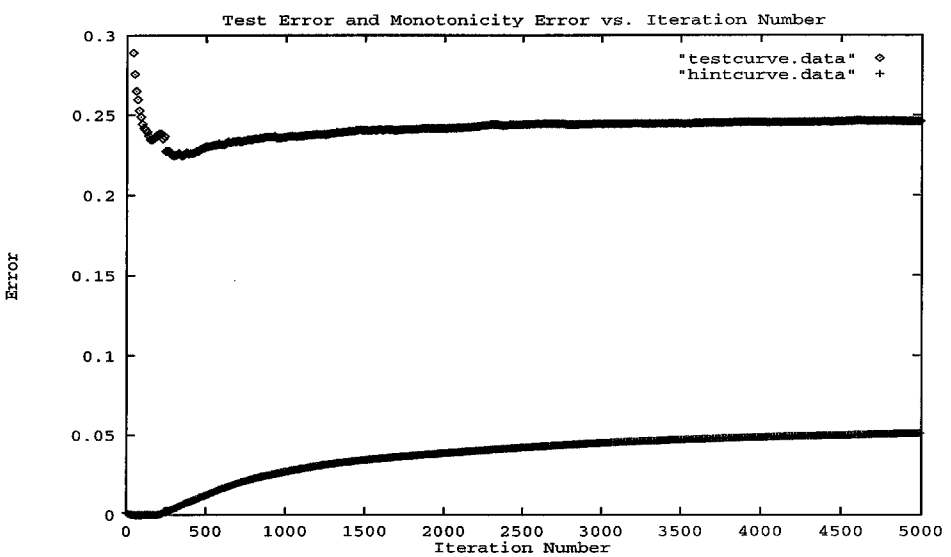

Figure 2.1: The violation of monotonicity tracks the overfitting occurring during training

For each input variable, 500 pairs of vectors representing monotonicity in that variable were generated. λ was chosen to be 5000. This level of λ amounted to a

strong bias towards monotonicity, although the model did have the option to violate monotonicity slightly, as we shall see in the next section.

Since the monotonicity constraint is being taught to the network with examples, the same generalization issues which arise when learning a function apply when learning the constraint. Specifically, there is a risk that the model is flexible enough to fit the monotonicity examples while actually violating monotonicity at other points in the input space. To gauge whether generalization has occurred with respect to the monotonicity property, we need an unbiased, out-of-sample estimate of $\mathcal{E}[E_h]$. Therefore, 100 test pairs per input variable were generated but excluded from the objective function. E_h was monitored on these pairs throughout the course of training, yielding an unbiased measure of the model's adherence to monotonicity.

For contrast, monotonicity test error was also monitored for the two-layer networks trained only on the training data (i.e., no hint). Figure 2.1 shows test error and monotonicity error vs. training time for the credit data for a 6 hidden unit network trained only on the training data (i.e, no hints), averaged over the 200 different data splits. The monotonicity error is multiplied by a factor of 10 in the figure to make it more easily visible. Interestingly, the figure indicates a substantial correlation between overfitting and monotonicity error during the course of training. One plausible interpretation of the figure is that the network's deviations from monotonicity correspond to fitting noise rather than signal.

2.3.3 Results

For all three applications , the nonlinear models which minimized training error as much as possible performed much worse than any of the other methods. Without any variance-reducing measures such as early stopping or monotonicity constraints, the high variance of the nonlinear networks easily outweighs the low bias. The poor performance was so pronounced that it was not deemed worthwhile to perform additional experiments which varied the number of hidden units.

The results for the credit card and liver problems have very similar profiles. Even

Problem	model	training error	test error
Credit	6 h.u.	$15.2\% \pm 0.1\%$	$24.6\% \pm 0.3\%$
Liver	3 h.u.	$0.640 \pm .003$	$0.920 \pm .014$
Bond	4 h.u.	$1.22 \pm .01$	$4.86 \pm .16$

Table 2.1: Performance of nonlinear networks without early stopping

Model	training error	test error	hint test error
Linear	$22.7\% \pm 0.1\%$	$23.7\% \pm 0.2\%$	-
4 h.u.	$20.0\% \pm 0.3\%$	$23.9\% \pm 0.4\%$	.000329
6 h.u.	$19.8\% \pm 0.2\%$	$23.8\% \pm 0.4\%$	.000693
8 h.u.	$19.2\% \pm 0.1\%$	$23.5\% \pm 0.4\%$	.000740

Table 2.2: Performance of standard methods on credit problem

Model	training error	test error	hint test error
4 h.u.	$20.9\% \pm 0.1\%$	$22.0\% \pm 0.3\%$	.000001
6 h.u.	$18.4\% \pm 0.1\%$	$21.7\% \pm 0.3\%$	.000015
8 h.u.	$18.0\% \pm 0.1\%$	$21.6\% \pm 0.3\%$	.000044

Table 2.3: Performance of hint method on credit problem

Model	training error	test error	hint test error
Linear	$0.792 \pm .006$	$0.897 \pm .014$	-
2 h.u.	$0.759 \pm .008$	$0.896 \pm .013$	.000922
3 h.u.	$0.744 \pm .009$	$0.890 \pm .013$	.001062
4 h.u.	$0.760 \pm .009$	$0.897 \pm .014$	.000832
5 h.u.	$0.758 \pm .009$	$0.895 \pm .014$	.000843
6 h.u.	$0.734 \pm .008$	$0.897 \pm .014$	.000939

Table 2.4: Performance of standard methods on liver problem

Model	training error	test error	hint test error
2 h.u.	$0.771 \pm .004$	$0.868 \pm .014$	.000001
3 h.u.	$0.751 \pm .004$	$0.858 \pm .013$	.000000
4 h.u.	$0.734 \pm .003$	$0.863 \pm .014$	.000001
5 h.u.	$0.729 \pm .004$	$0.864 \pm .014$	.000001
6 h.u.	$0.721 \pm .003$	$0.863 \pm .013$	.000002

Table 2.5: Performance of hint methods on liver problem

Method	training error	test error	hint test error
2 h.u.	$2.46 \pm .042$	$3.83 \pm .094$	.007454
4 h.u.	$2.19 \pm .048$	$3.81 \pm .083$	.007056
6 h.u.	$2.14 \pm .054$	$3.77 \pm .072$	.006836

Table 2.6: Performance of standard methods on bond problem

with early stopping, the performances of the two-layer networks are no better than those of the linear models. This similarity in performance is consistent with the thesis of a monotonic target function. A monotonic classifier may be thought of as a mildly nonlinear generalization of a linear classifier. The two-layer network does have the advantage of lower bias, i.e., the capacity to implement some of the nonlinearity present in the target function. However, this advantage is cancelled out (and in other cases could be outweighed) by higher variance, i.e., overfitting resulting from excessive and unnecessary degrees of freedom. When monotonicity hints are introduced, much of this unnecessary freedom is eliminated, although the network is still allowed to implement monotonic nonlinearities. Accordingly, a modest but clearly statistically significant improvement on the credit problem (nearly 2%) results from the introduction of monotonicity hints. Such an improvement could translate into a substantial increase in profit for a bank. Monotonicity hints also significantly improve test error on the liver problem; about 3% more of the target variance is explained. These results are quite consistent over a range of hidden units. The hint test error results support our claim that the improvement is due to the constraint. This measure of monotonicity error is quite low for the models trained with the hint, while it is

Method	training error	test error	hint test error
2 h.u.	$2.87 \pm .027$	$4.09 \pm .150$	.000040
4 h.u.	$2.62 \pm .028$	$3.78 \pm .072$	.000062
6 h.u.	$2.31 \pm .022$	$3.74 \pm .093$	.000080

Table 2.7: Performance of hint methods on bond problem

orders of magnitude higher without the hint. Thus, the models trained with the hint come much closer to being purely monotonic.

It is also important, however, to note that in most cases training with the hint did not result in perfectly monotonic models. This may be a reflection of the target functions' lack of strict adherence to monotonicity. Interestingly, even if the target is perfectly monotonic, it may be beneficial to allow minor deviations by the model. If the model cannot implement the target perfectly, then it is possible that the model's best possible approximation is slightly non-monotonic.

The results for the bond rating prediction problem are less clear cut. In this case, early stopping is enough to enable the nonlinear models to outperform the linear model. The best performance is obtained by a model constrained by monotonicity, but there is not a statistically significant improvement over the models trained only with early stopping. Perhaps monotonicity was not enforced sufficiently strictly- note the non-negligible hint test errors. This theory is supported by the results of the next chapter, where a model which is strictly monotonic is shown to outperform this method on the bond rating problem.

2.4 Conclusion

This chapter has shown that monotonicity hints can significantly improve the performance of a neural network on noisy real-world tasks. The method has virtues, such as compatibility with any parametrized model and any optimization technique, which distinguish it from other methods. Future work may include application of the method to situations where it is clear that the target is only approximately monotonic.

An intelligent choice of λ would become crucial in such situations. An important line of research, therefore, would determine how to set λ to best reflect our prior beliefs about the task to be learned. Another interesting question concerns the number of hint examples pairs required for generalization with respect to the monotonicity property to occur. How does this scale with measures of the flexibility of the model, e.g., the VC dimension?

Chapter 3 Monotonic Networks

3.1 Introduction

This chapter continues the investigation of techniques for enforcing monotonicity constraints in machine learning models. The previous chapter presented a way to enforce monotonicity approximately by adding a second term measuring "monotonicity error" to the usual error measure. This method's approximate enforcement of monotonicity was portrayed as a virtue. This property has a downside, however. If we are confident that the target is strictly monotonic, then we would like to be able to produce a model which also obeys monotonicity precisely. Another problem with the penalty term approach is the computational cost of training the model. A large number of hint example pairs must be generated. It is unclear how the number of pairs scales with the flexibility of the model. It may be that the cost in training time is prohibitive for large models with many free parameters. These problems would be solved if we had a model which obeys monotonicity exactly, i.e., by virtue of functional form.

We present here such a model, which we will refer to as a monotonic network. A monotonic network implements a piecewise-linear surface by taking maximum and minimum operations on groups of hyperplanes. Monotonicity constraints are enforced by constraining the signs of the hyperplane weights. Monotonic networks can be trained using the usual gradient-based optimization methods typically used with other models such as feedforward neural networks. Armstrong [Armstrong *et al.*, 1996] has developed a model called the adaptive logic network which is capable of enforcing monotonicity and appears to have some similarities to the approach presented here. The adaptive logic network, however, is available only through a commercial software package. The training algorithms are proprietary and have not been fully disclosed in academic journals. The monotonic network therefore represents (to the best of our knowledge) the first parametrized model to be presented in an academic setting

which has the ability to enforce monotonicity.

Section 3.2 describes the architecture and training procedure for monotonic networks. Section 3.3 presents a proof that monotonic networks can uniformly approximate any continuous monotonic function with bounded partial derivatives to an arbitrary level of accuracy. Monotonic networks are applied to a real-world problem in bond rating prediction in Section 3.4. In Section 3.5, we discuss the results and consider future directions.

3.2 Architecture and Training Procedure

A monotonic network has a feedforward, three-layer (two hidden-layer) architecture (Fig. 3.1). The first layer of units compute different linear combinations of the input vector. If increasing monotonicity is desired for a particular input, then all the weights connected to that input are constrained to be positive. Similarly, weights connected to an input where decreasing monotonicity is required are constrained to be negative. The first layer units are partitioned into several groups (the number of units in each group is not necessarily the same). Corresponding to each group is a second layer unit, which computes the maximum over all first-layer units within the group. The final output unit computes the minimum over all groups.

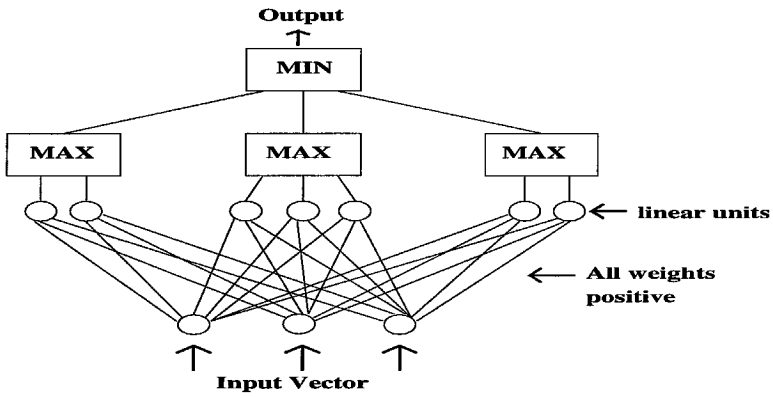

Figure 3.1: This monotonic network obeys increasing monotonicity in all 3 inputs because all weights in the first layer are constrained to be positive.

More formally, if we have K groups with outputs $g_1, g_2, \ldots g_K$, and if group k

consists of h_k hyperplanes $\mathbf{w^{(k,1)}}, \mathbf{w^{(k,2)}}, \ldots \mathbf{w^{(k,h_k)}}$, then

$$g_k(\mathbf{x}) = \max_j \mathbf{w^{(k,j)}} \cdot \mathbf{x} - t^{(k,j)}, 1 \leq j \leq h_k$$

Let y be the final output of the network. Then

$$y = \min_k g_k(\mathbf{x})$$

or, for classification problems,

$$y = \sigma(\min_k g_k(\mathbf{x}))$$

where $\sigma(u) =$ e.g. $\frac{1}{1+e^{-u}}$.

In the discussions which follow, it will be useful to define the term *active*. We will call a group l active at $\mathbf{x}$ if

$$g_l(\mathbf{x}) = \min_k g_k(\mathbf{x})$$

, i.e., if the group determines the output of the network at that point. Similarly, we will say that a hyperplane is active at $\mathbf{x}$ if its group is active at $\mathbf{x}$ and the hyperplane is the maximum over all hyperplanes in the group.

As will be shown in the following section, the three-layer architecture allows a monotonic network to approximate any continuous, differentiable monotonic function arbitrarily well, given sufficiently many groups and sufficiently many hyperplanes within each group. The maximum operation within each group allows the network to approximate convex (positive second derivative) surfaces, while the minimum operation over groups enables the network to implement the concave (negative second derivative) areas of the target function (Fig. 3.2).

Monotonic networks can be trained using many of the standard gradient-based optimization techniques commonly used in machine learning. The gradient for each hyperplane is found by computing the error over all examples for which the hyperplane is active. After the parameter update is made according to the rule of the optimization technique, each training example is reassigned to the hyperplane that is now active at

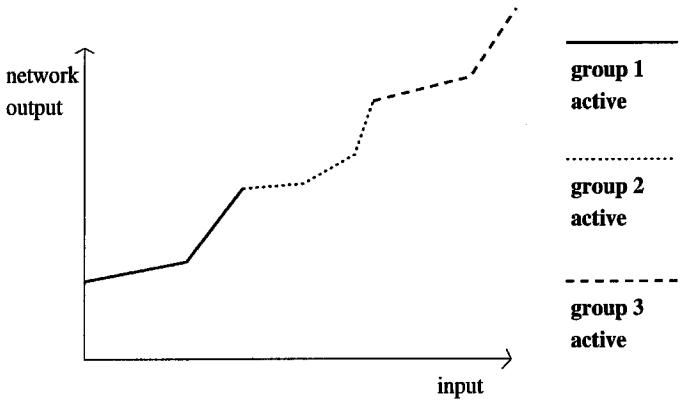

Figure 3.2: This surface is implemented by a monotonic network consisting of three groups. The first and third groups consist of three hyperplanes, while the second group has only two.

that point. The set of examples for which a hyperplane is active can therefore change during the course of training.

The constraints on the signs of the weights are enforced using an exponential transformation. If increasing monotonicity is desired in input variable i, then $\forall j,k$ the weights corresponding to the input are represented as $w_i^{(j,k)} = e^{z_i^{(j,k)}}$. The optimization algorithm can modify $z_i^{(j,k)}$ freely during training while maintaining the constraint. If decreasing monotonicity is required, then $\forall j,k$ we take $w_i^{(j,k)} = -e^{z_i^{(j,k)}}$.

3.3 Universal Approximation Capability

In this section, we demonstrate that monotonic networks have the capacity to approximate uniformly to an arbitrary degree of accuracy any continuous, bounded, differentiable function on the unit hypercube $[0,1]^D$ which is monotonic in all variables and has bounded partial derivatives. We will say that $\mathbf{x}'$ *dominates* $\mathbf{x}$ if $\forall 1 \leq d \leq D$, $x'_d \geq x_d$. A function m is monotonic in all variables if it satisfies the constraint that $\forall \mathbf{x}, \mathbf{x}'$, if $\mathbf{x}'$ dominates $\mathbf{x}$ then $m(\mathbf{x}') \geq m(\mathbf{x})$.

Theorem 3.3.1 Let $m(\mathbf{x})$ be any continuous, bounded monotonic function with bounded partial derivatives, mapping $[0,1]^D$ to $\mathbb{R}$. Then there exists a function $m_{net}(\mathbf{x})$ which can be implemented by a monotonic network and is such that, for any ϵ and any $\mathbf{x} \in [0,1]^D$,$|m(\mathbf{x}) - m_{net}(\mathbf{x})| < \epsilon$.

Proof:

Let b be the maximum value and a be the minimum value which m takes on $[0,1]^D$. Let α bound the magnitude of all partial first derivatives of m on $[0,1]^D$. Define an equispaced grid of points on $[0,1]^D$, where $\delta = \frac{1}{n}$ is the spacing between grid points along each dimension. I.e., the grid is the set S of points $(i_1\delta, i_2\delta, \ldots i_D\delta)$ where $1 \leq i_1 \leq n, 1 \leq i_2 \leq n, \ldots 1 \leq i_D \leq n$. Corresponding to each grid point $\mathbf{x}' = (x'_1, x'_2, \ldots x'_D)$, assign a group consisting of $D+1$ hyperplanes. One hyperplane in the group is the constant output plane $y = m(\mathbf{x}')$. In addition, for each dimension d, place a hyperplane $y = \gamma(x_d - x'_d) + m(\mathbf{x}')$, where $\gamma > \frac{b-a}{\delta}$. This construction ensures that the group associated with $\mathbf{x}'$ cannot be active at any point $\mathbf{x}^*$ where there exists a d such that $x^*_d - x'_d > \delta$, since the group's output at such a point must be greater than b and hence greater than the output of a group associated with another grid point.

Now consider any point $\mathbf{x} \in [0,1]^D$. Let $\mathbf{s^{(1)}}$ be the unique grid point in S such that $\forall d$, $0 \leq x_d - s_d{}^{(1)} < \delta$, i.e., $\mathbf{s^{(1)}}$ is the closest grid point to $\mathbf{x}$ which $\mathbf{x}$ dominates. Then we can show that $m_{net}(\mathbf{x}) \geq m(\mathbf{s^{(1)}})$. Consider an arbitrary grid point $\mathbf{s}' \neq \mathbf{s^{(1)}}$. By the monotonicity of m, if $\mathbf{s}'$ dominates $\mathbf{s^{(1)}}$, then $m(\mathbf{s}') \geq m(\mathbf{s^{(1)}})$, and hence, the group associated with s' has a constant output hyperplane $y = m(\mathbf{s}') \geq m(\mathbf{s^{(1)}})$ and therefore outputs a value $\geq m(\mathbf{s^{(1)}})$ at $\mathbf{x}$. If $\mathbf{s}'$ does not dominate $\mathbf{s^{(1)}}$, then there exists a d such that $s_d{}^{(1)} > s'_d$. Therefore, $x_d - s'_d \geq \delta$, meaning that the output of the group associated with $\mathbf{s}'$ is at least $b \geq m(\mathbf{s^{(1)}})$. All groups have output at least as large as $m(\mathbf{s^{(1)}})$, so we have indeed shown that $m_{net}(\mathbf{x}) \geq m(\mathbf{s^{(1)}})$. Now consider the grid point $\mathbf{s^{(2)}}$ that is obtained by adding δ to each coordinate of $\mathbf{s^{(1)}}$. The group associated with $\mathbf{s^{(2)}}$ outputs $m(\mathbf{s^{(2)}})$ at $\mathbf{x}$, so $m_{net}(\mathbf{x}) \leq m(\mathbf{s^{(2)}})$. Therefore, we have $m(\mathbf{s^{(1)}}) \leq m_{net}(\mathbf{x}) \leq m(\mathbf{s^{(2)}})$. Since $\mathbf{x}$ dominates $\mathbf{s^{(1)}}$ and is dominated by $\mathbf{s^{(2)}}$, by monotonicity we also have $m(\mathbf{s^{(1)}}) \leq m(\mathbf{x}) \leq m(\mathbf{s^{(2)}})$. $|m(\mathbf{x}) - m_{net}(\mathbf{x})|$ is therefore bounded by $|m(\mathbf{s^{(2)}}) - m(\mathbf{s^{(1)}})|$. By Taylor's theorem for multivariate functions, we know that

$$m(\mathbf{s^{(2)}}) - m(\mathbf{s^{(1)}}) = \delta \sum_{d=1}^{D} \frac{\partial m(\mathbf{c})}{\partial x_d}$$

for some point $\mathbf{c}$ on the line segment between $\mathbf{s^{(1)}}$ and $\mathbf{s^{(2)}}$. Given the assumptions made at the outset, $|m(\mathbf{s^{(2)}}) - m(\mathbf{s^{(1)}})|$, and hence, $|m(\mathbf{x}) - m_{net}(\mathbf{x})|$ can be bounded by $d\delta\alpha$. We take $\delta < \frac{\epsilon}{d\alpha}$ to complete the proof ■.

3.4 Experimental Results

The monotonic network was tested on the same set of databases used to evaluate the monotonicity hint technique in the previous chapter. The networks were trained for 1000 batch-mode iterations of gradient descent with momentum and an adaptive learning rate. The parameters of each hyperplane in the network were initialized to be the parameters of the linear model obtained from the training set, plus a small random perturbation. This procedure ensured that the network was able to find a reasonably good fit to the data. Once again, since the meanings of the features were not known, it was not known *a priori* whether increasing or decreasing monotonicity should hold for each feature. The directions of monotonicity were determined by observing the signs of the weights of the linear model obtained from the training data. The results shown are averages over 200 training/test splits for the credit and bond problem and 100 splits for the liver problem.

Model	training error	test error
2 groups, 2 planes per group	$21.3\% \pm 0.1\%$	$22.7\% \pm 0.4\%$
3 groups, 3 planes per group	$20.5\% \pm 0.1\%$	$22.8\% \pm 0.4\%$
4 groups, 4 planes per group	$20.1\% \pm 0.1\%$	$23.0\% \pm 0.4\%$
5 groups, 5 planes per group	$19.7\% \pm 0.1\%$	$22.9\% \pm 0.4\%$

Table 3.1: Performance of monotonic networks on credit problem

The performance of the monotonic network appears to complement the monotonicity hint technique nicely. On all three problems, the monotonic network outperforms the standard methods which do not capitalize on the monotonicity information. The

Model	training error	test error
2 groups, 2 planes per group	$0.785 \pm .006$	$0.880 \pm .020$
3 groups, 3 planes per group	$0.780 \pm .006$	$0.877 \pm .019$
4 groups, 4 planes per group	$0.785 \pm .005$	$0.872 \pm .019$
5 groups, 5 planes per group	$0.786 \pm .005$	$0.867 \pm .019$

Table 3.2: Performance of monotonic networks on liver problem

Model	training error	test error
2 groups, 2 planes per group	$2.78 \pm .05$	$3.71 \pm .07$
3 groups, 3 planes per group	$2.64 \pm .04$	$3.56 \pm .06$
4 groups, 4 planes per group	$2.50 \pm .04$	$3.48 \pm .06$
5 groups, 5 planes per group	$2.44 \pm .03$	$3.43 \pm .06$

Table 3.3: Performance of monotonic networks on bond rating problem

monotonic network does somewhat worse on the credit and liver problems than the monotonicity hint technique. Recall that on the bond rating problem, however, the hint method was unable to achieve a statistically significant improvement over standard methods. In contrast, the monotonic network outperforms other methods by a wide margin which is unquestionably statistically significant.

One can speculate about the reasons for the differences in performance of the two methods of incorporating monotonicity, but it is difficult to say for sure what the correct explanation is. Most likely, there is an element of happenstance behind the differences. Clearly, the class of monotonic functions implementable by a neural network with a small number of hidden units is not exactly the same as the class of functions implementable by a monotonic network with a small number of groups and planes per group. It may be that the neural net happens to be able to approximate the target functions better in the credit and liver applications, while the monotonic network is better suited for the bond rating problem. Perhaps this could have been anticipated with further domain knowledge about the tasks, but it is unlikely. The most important message, however, is that on each of the problems at least one of the two techniques yielded clear benefits.

3.5 Conclusion

We presented a model, the monotonic network, in which monotonicity constraints can be enforced exactly, without adding a second term to the usual objective function. A straightforward method for implementing and training such models was demonstrated, and the method was shown to outperform other methods on a real-world problem.

Several areas of research regarding monotonic networks need to be addressed in the future. One issue concerns the choice of the number of groups and number of planes in each group. In general, the usual bias-variance tradeoff that holds for other models will apply here, and the optimal number of groups and planes will be quite difficult to determine *a priori*. There may be instances where additional prior information regarding the convexity or concavity of the target function can guide the decision, however. Another interesting observation is that a monotonic network could also be implemented by reversing the maximum and minimum operations, i.e., by taking the maximum over groups where each group outputs the minimum over all of its hyperplanes. It will be worthwhile to try to understand when one approach or the other is most appropriate.

Chapter 4 Theory of Monotonicity Constraints

4.1 Introduction

Much of learning theory is concerned with measuring the flexibility and approximating power of various function classes. Concepts such as capacity [Cover, 1965], VC dimension [Vapnik and Chervonenkis, 1971] and effective number of parameters [Moody, 1992] have been developed with this goal in mind.

Most function classes analyzed in these frameworks are explicitly parametrized functional forms such as sigmoidal neural networks of a given architecture. It is also of interest, however, to consider classes of functions which satisfy properties the target function may in some cases be believed to possess. Monotonicity is an example of a constraint the target function is likely to satisfy in some instances. In many application domains, common sense or expert knowledge indicates that the target function is monotonic in some or all input variables. Chapters 2 and 3 demonstrated that constraining models such as neural networks to obey monotonicity can lead to an improvement in performance over both linear models and unconstrained nonlinear models. It would therefore be significant both practically and theoretically to quantify the expressive power of monotonic functions.

We will consider the class $\mathbf{M}$ of monotonically increasing (non-decreasing) functions from $\mathbf{R}^d \rightarrow \{0,1\}$. Let $\mathbf{x} = (x_1, x_2, \ldots x_d)$ and $\mathbf{x}' = (x'_1, x'_2, \ldots x'_d)$ be members of $\mathbf{R}^d$. We will say that $\mathbf{x}'$ *dominates* $\mathbf{x}$, which we denote by $\mathbf{x}' \geq \mathbf{x}$, if $\forall i, 1 \leq i \leq d, x'_i \geq x_i$. Domination defines a partial ordering on $\mathbf{R}^d$.

Definition 4.1 Define the class $\mathbf{M}$ as the set of all functions f such that

$$\mathbf{x}' \geq \mathbf{x} \Rightarrow f(\mathbf{x}') \geq f(\mathbf{x})$$

In many applications, domain knowledge may indicate decreasing monotonicity (i.e., a monotonically decreasing relationship between input and output) in some variables rather than increasing monotonicity. The analysis which follows will also hold for each of the other $2^d - 1$ function classes where some or all variables have a decreasing monotonicity constraint rather than an increasing one. This equivalence is made clear by observing that decreasing monotonicity may be converted to increasing monotonicity by relabelling an input variable as its negation. There are also many situations where monotonicity only holds for some variables, while the relationship of the other variables to the output is completely unknown *a priori*. This case is more complex and will not be addressed here. Note that the class **M** is not explicitly parametrized by weights, unlike classes such as sigmoidal networks with a fixed number of hidden units. When a finite, parametrized model is further constrained to obey monotonicity in all variables, the resulting class of functions will be some subset of **M**. Thus, bounds on the capacity of **M** will upper-bound the capacity of any parametrized model where monotonicity is enforced.

Section 4.2 describes the decision boundaries of monotonic classifiers, comparing and contrasting them to separating hyperplanes. Results are developed about the capacity and VC dimension of **M** in section 4.3. In particular, the capacity is shown to depend almost completely on the input distribution. Section 4.4 presents methods for estimating the capacity and bounding the VC entropy of **M** given a model of the input distribution. These techniques are implemented and shown to yield in some cases much tighter bounds than those which result from bounding the VC dimension of feedforward neural networks with very few hidden units. In section 4.5, analytical results are derived concerning how the capacity of **M** grows with d for independent inputs. Section 4.6 discusses the results and considers future work.

4.2 Decision boundaries

A monotonic classifier may be thought of as a mildly nonlinear generalization of a linear classifier. This relationship is perhaps best demonstrated by considering the

decision boundaries corresponding to the two models. It is well known that the decision boundary implemented by a linear perceptron is simply a $d-1$ dimensional hyperplane splitting input space into two regions. In two dimensions, this boundary consists of a straight line dividing the input plane.

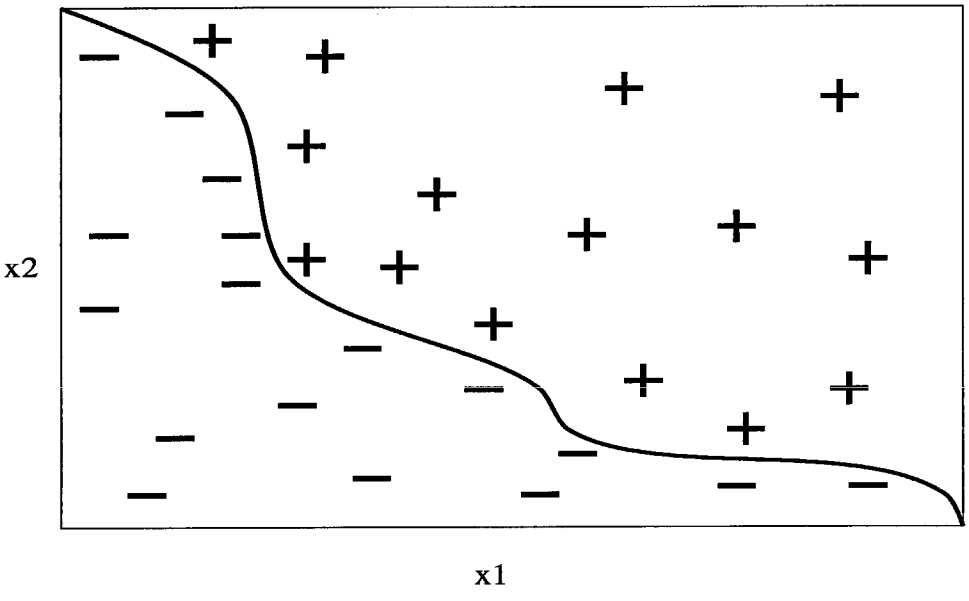

Figure 4.1: A monotonic classifier splits the input plane with a decision boundary consisting of a one-to-one mapping between the two input variables. Monotonic classifiers are analogous to but slightly more flexible than linear classifiers, which split the input plane with a straight line.

Consider a monotonic classifier $\theta(m(x_1, x_2))$, where m maps $\mathbf{R}^2$ to $\mathbf{R}$ and $\theta(u)$ is the Heaviside step function, i.e., $\theta(u) = 1$ if $u \geq 0$ and $\theta(u) = 0$ otherwise. Let $m(x_1, x_2)$ be a continuous, differentiable, strictly increasing function of both variables, i.e., $\forall x_1, x_2$ we have $\frac{\partial m}{\partial x_1} > 0$ and $\frac{\partial m}{\partial x_2} > 0$. Assume that for any value of x_1, there exists an x_2 for which $m(x_1, x_2) > 0$ and an x_2 for which $m(x_1, x_2) < 0$. Similarly, assume that for any value of x_2, there exists an x_1 for which $m(x_1, x_2) > 0$ and an x_1 for which $m(x_1, x_2) < 0$. Define the decision boundary of a classifier $\theta(g(\mathbf{x}))$ to be the set of points $B = \{\mathbf{x} : g(\mathbf{x}) = 0\}$. If the above conditions hold, then we have the following theorem.

Theorem 4.2.1 The decision boundary of a monotonic classifier in $\mathbf{R}^2$ is a one-to-one mapping between the two input variables, i.e., an invertible function.

Proof: Fix $x_1 = a$, and let $x_2^{min}(a)$ be the smallest value x_2 can take such that

$m = 1$. By continuity in m, $m(a, x_2^{min}(a)) = 0$, and by the strictly increasing nature of m, $\forall x_2 > x_2^{min}, m(a, x_2) > 0$. Therefore, x_2^{min} is the only value of x_2 for $x_1 = a$ which lies on the decision boundary. By an analogous argument, for each value of x_2, there is a unique value of x_1 such that the point lies on the decision boundary. Therefore, the boundary must be an invertible function from one input variable to the other ■.

This theorem is in agreement with the intuition that a monotonic model is more flexible than a linear one, but is still very severely constrained.

If we make analogous assumptions in the d-dimensional case, then the boundary must be a single, somewhat flexible sheet such that specifying the values of any subset of $d-1$ input variables defines a unique value of the dth variable at which the classification changes.

4.3 Basic Capacity Results

To make the idea of capacity precise, we must define a few auxiliary concepts. Define a *dichotomy* to be a set of d-dimensional input vectors, each of which have an associated class label of either 0 or 1. Define a *positive* example as an input vector labelled 1 and a *negative* example as an input vector labelled 0. We say that a dichotomy is *separable* by a function class if there exists at least one function in the class which maps each of the input vectors to its correct class label. A *random dichotomy* is a dichotomy where the label for each example is assigned randomly with equal probability for either class. Let $P(N)$ be the probability that a random dichotomy of N examples can be separated by the function class. The *capacity* of a function class is the integer N^* for which $P(N)$ is closest to 0.5. Capacity (unlike VC dimension) is therefore a quantity which *depends on the input distribution.* The importance of this point will become clear below.

The following theorem provides a polynomial time test for monotonic separability:

Theorem 4.3.1 *A dichotomy is separable by* **M** *if and only if there exists no negative example which dominates some positive example*

Proof: The necessity of this condition is obvious. Sufficiency may be demonstrated by constructing a function belonging to **M** which implements the dichotomy. Consider the function f which classifies as 1 only those input vectors which dominate some positive example in the dichotomy. By construction, f obviously separates the dichotomy correctly. It is also clear that f belongs to **M**. Suppose $f(\mathbf{x}) = 1$ and $\mathbf{x}' \geq \mathbf{x}$. $\mathbf{x}$ must dominate some vector $\mathbf{x}^*$ in the dichotomy. Any vector $\mathbf{x}'$ which dominates $\mathbf{x}$ must also dominate $\mathbf{x}^*$. Thus, $f(\mathbf{x}') = 1$, hence, $f(\mathbf{x}') \geq f(\mathbf{x})$. Therefore, $f \in \mathbf{M}$ ■.

It follows immediately from the theorem that a dichotomy may be checked for monotonic separability in time at most quadratic in the number of examples.

Capacity results for the perceptron (i.e., the class of linear threshold functions) are well known: the capacity is $2d$, where d is the dimensionality of the input space [Cover, 1965]. This result is true for any smooth distribution of continuous input variables, because the number of linearly separable dichotomies of a set of input vectors is independent of how those input vectors are arranged, provided they are in general position. This lack of dependence on input distribution is in sharp contrast to the case of **M**. Here, the number of dichotomies depends *heavily* on how the input vectors are arranged.

Consider figure 4.2. A little inspection will reveal that this dichotomy is not monotonically separable, even if we are free to take decreasing monotonicity to hold for one or both of the input variables. Figure 4.3 depicts a drastically different situation. In this case, the input distribution is such that there exists a monotonically *decreasing* relationship between the two input variables, i.e., for all $\mathbf{x} = (x_1, x_2)$ and $\mathbf{x}' = (x'_1, x'_2)$, $x_1 > x'_1 \Rightarrow x_2 < x'_2$. When N input points are drawn from such a distribution, any of the 2^N possible dichotomies are separable by **M**, the class of monotonically *increasing* functions. To see why, consider that for a dichotomy not to be separable by **M**, there must be a negative example which is dominated by some positive example. But if the two input variables are related in a monotonically decreasing way, no example can dominate any other example. All dichotomies are therefore separable given such an input distribution. Note that even if the input

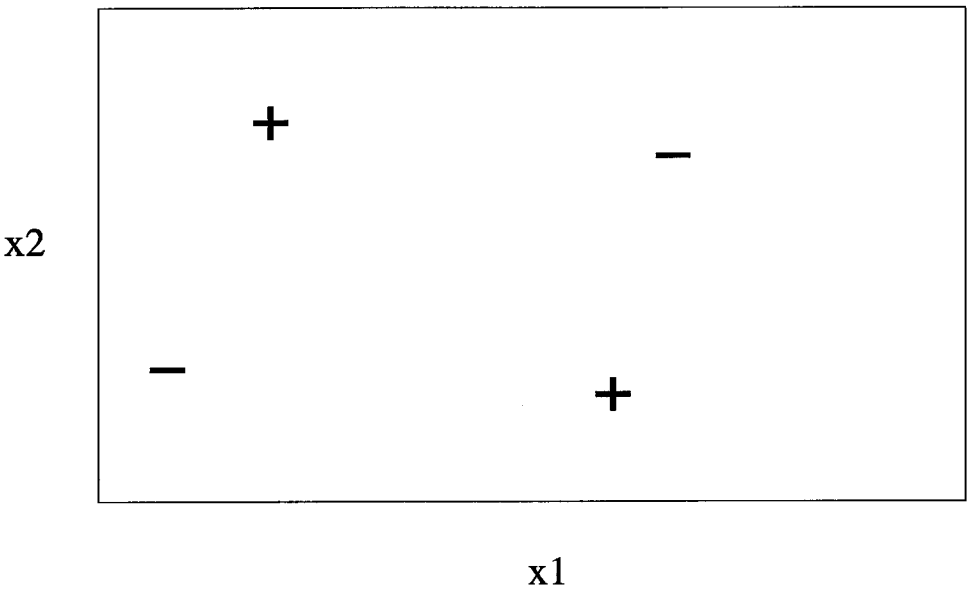

Figure 4.2: This dichotomy of four points cannot be implemented by any monotonic function, regardless of whether the monotonicity constraint in each variable is increasing or decreasing. +'s indicate positive examples, -'s negative examples.

dimensionality is greater than 2, a monotonically decreasing relationship between two of the input variables is sufficient to make domination impossible, and hence, to make all dichotomies separable by **M**.

The example depicted in figure 4.3 establishes a result which explains why input distribution-dependent notions such as capacity are more useful than the concept of VC dimension for the analysis of monotonicity. Recall that the VC dimension is the maximum value of N for which the growth function $= 2^N$. The growth function is defined as the maximum number of separable dichotomies of N points, where this maximum is taken over all possible choices of the N points. Figure 4.3 demonstrates that we may always choose input points in such a way that all 2^N possible dichotomies may be separated monotonically. It is granted that the figure 4.3 example is not very realistic- it would be extremely odd to find a problem where the target is believed to increase monotonically with two input variables which appear to be related to each other in a monotonically decreasing way. Nonetheless, such an example is permitted by the definition of the growth function. Thus, the VC dimension of the class of monotonic functions $= \infty$! This result is misleading, however, since monotonicity is

still a very powerful constraint in most cases.

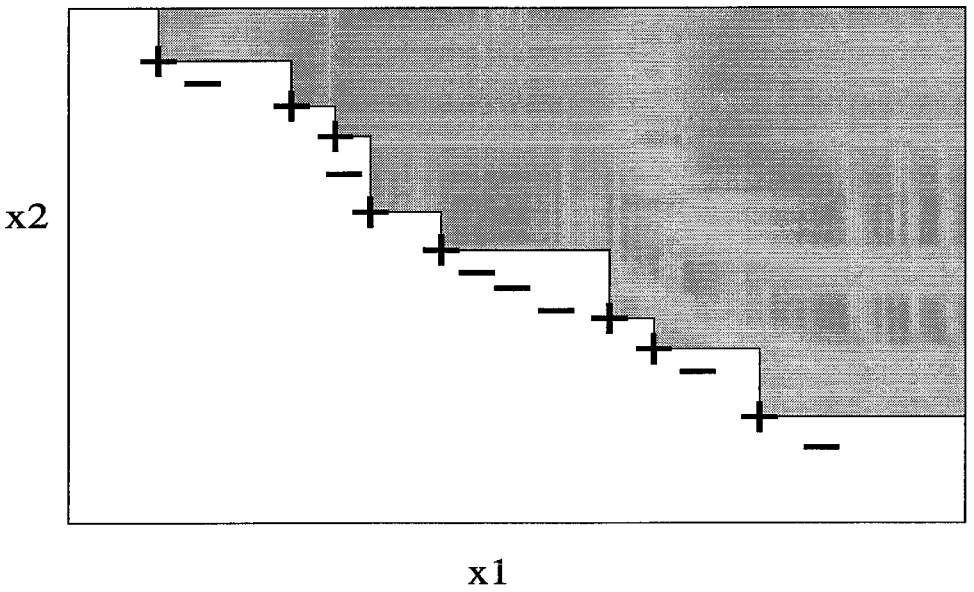

Figure 4.3: If two input variables x_1 and x_2 are related in a monotonically decreasing way, then any dichotomy may be separated by M. A particular, arbitrary dichotomy is shown here. The shaded area indicates the region of input space which, by monotonicity, must be classified positively.

Figure 4.4 depicts a situation opposite to that of figure 4.3. In this case, there exists a monotonically *increasing* relationship between the two input variables. The points are totally ordered, i.e., every point either dominates or is dominated by every other point. The class **M** has very little separating power in such a situation. **M** is free only to choose a dividing point from among the N input vectors and classify positively those vectors dominating the dividing vector. Thus, only $N + 1$ of the 2^N possible dichotomies are separable by **M**. The number of separable dichotomies remains $N + 1$ if we have d input variables all of which are related to each other in a monotonically increasing way, i.e., all of which rise and fall together.

The preceding examples demonstrate that the number of dichotomies separable by **M**, and hence, the capacity of **M** can be arbitrarily large or small depending on the particular input distribution. The second and third examples- especially the third- cannot be dismissed as merely irrelevant, degenerate cases which will never occur in real life. These two examples are the extreme versions of possible real world situations where the input variables do not have strict monotonic relationships but

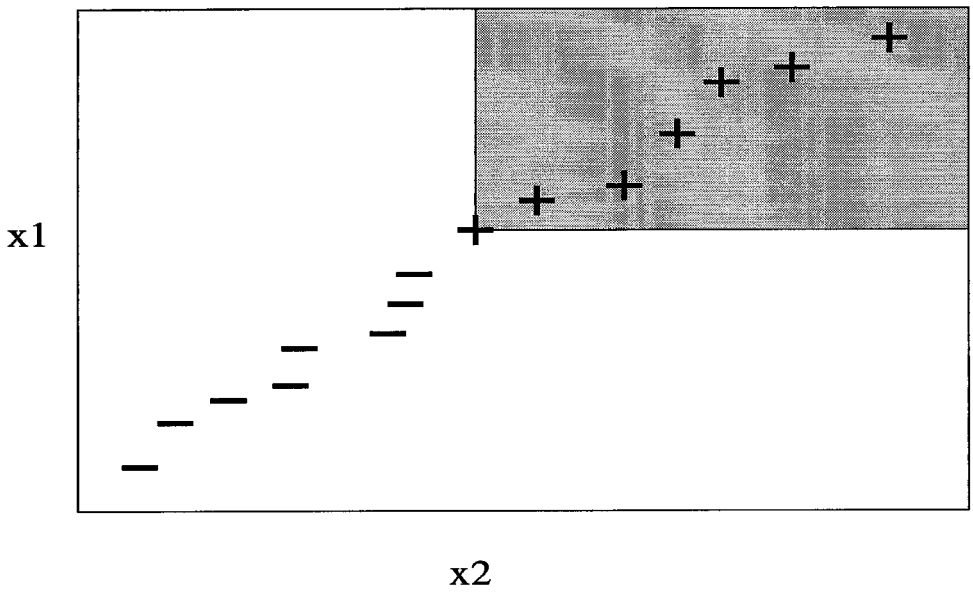

Figure 4.4: If two input variables x_1 and x_2 are related in a monotonically increasing way, then only $n+1$ of the 2^n possible dichotomies are separable by **M**. The shaded area indicates the region of input space which, by monotonicity, must be classified positively.

are correlated significantly. It should be clear from figure 4.4 that if we have two input variables which are highly but not perfectly positively correlated, then the number of separable dichotomies of N points is still likely to be low, although somewhat higher than $N+1$. Likewise, if we have two input variables which have a substantial negative correlation, then the number of separable dichotomies is likely to be quite high, although somewhat less than 2^N. The effect of correlation will be demonstrated numerically in the next section.

4.4 Capacity and VC Entropy Estimation

If we have a good model for the input distribution for a given problem, the capacity of **M** may be estimated computationally. N input vectors may be drawn from the model of the input distribution and labelled randomly as positive or negative with equal probability. Theorem 4.3.1 tells us how to check efficiently whether or not the dichotomy generated is separable by **M**. This procedure may be repeated many times to get an estimate of $P(N)$, the probability that N randomly labelled points

are separable by **M**. The estimate of the capacity of **M** is then that N^* for which $P(N^*) \approx 0.5$.

This procedure was used to estimate the capacity of **M** for various d for the case of independent $\mathcal{N}(0,1)$ input variables. The number of examples N was varied over a wide range. For each N, 1000 random dichotomies were generated and checked for monotonic separability. The capacity estimate was taken to be the N^* for which the estimate of $P(N)$ was closest to 0.5. The results are shown in Table 4.1. The capacity of **M** is modest for low d, but grows more quickly than the capacity of the perceptron. This behavior agrees with our intuition that **M** is a highly constrained function class, but nonetheless more flexible than the class of perceptrons.

input dimension	capacity of perceptron	estimated capacity of M, independent inputs
2	4	4
3	6	6
4	8	8
5	10	12
6	12	17
7	14	23
8	16	33
9	18	45
10	20	64
11	22	85
12	24	126

Table 4.1: Capacity of M and of the perceptron given independent gaussian inputs

The effect of correlation between input variables was also explored for $d = 10$. We generated 10 $\mathcal{N}(0,1)$ input variables $x_1, ..., x_{10}$ according to a covariance matrix with 1s along the diagonal and ρ elsewhere. As the theory of the previous section predicts, the capacity decreases drastically with increasing ρ (Table 4.2).

In addition to estimating capacity, it would be useful to make explicit statements about the generalization of a monotonic model. The VC bounds on generalization, based on the growth function, are well known. Such a bound is of no use to us here, since $\forall N$, the growth function of $\mathbf{M} = 2^N$. An analogous, distribution-dependent

covariance	capacity of M, $d = 10$
0.	64
0.1	26
0.2	16
0.3	11
0.4	9
0.5	7

Table 4.2: Capacity of M, d=10 for various levels of correlation between inputs

bound also holds, however. The growth function can be replaced in the bound with the VC entropy $H(N)$. If we denote by π_m the true error rate of a classifier m within the class **M** and by ν_m the observed error rate, then we have the following theorem [Parrondo and Van den Broeck, 1993]:

$$Pr\{\sup_m |\pi_m - \nu_m| > \epsilon\} \leq 6H(2N)e^{-(\epsilon - \frac{1}{n})^2 n} \quad \textbf{(4.1)}$$

In theory, one could estimate $P(2N)$ using the technique outlined above for estimating capacity. Such a procedure is computationally infeasible for even modestly large N, however. Since $H(2N) = (2^{2N}P(2N))$, the bound may be written as $4(4e^{-\epsilon^2})^N P(2N)$. Substituting in $N = 5000$, $\epsilon = 0.25$, we find that $P(2N)$ would need to be the order of 10^{-2870} in order for the bound to be non-trivial. In order to demonstrate that $P(2N)$ is this low, at least the order of 10^{2870} (!) dichotomies would have to be generated and checked for monotonic separability. The number of dichotomies that need to be generated grows exponentially with N, so direct estimation of $P(2N)$ is not an efficient way to obtain a bound.

Fortunately, we can bound $H(2N)$ using a polynomial time algorithm. We appeal to a lemma in [Vapnik and Chervonenkis, 1971]. Let $\mathbf{x^{(1)}}, \mathbf{x^{(2)}}, \ldots \mathbf{x^{(N)}}$ be a set of input vectors and let $\Delta^{\mathbf{G}}(\mathbf{x^{(1)}}, \mathbf{x^{(2)}}, \ldots \mathbf{x^{(N)}})$ be the number of dichotomies induced by a function class **G** on this sample. Define $\binom{N}{i} = 0$ for $i > N$. If for all subsamples $\mathbf{x^{(i_1)}}, \mathbf{x^{(i_2)}}, \ldots \mathbf{x^{(i_s)}}$ of cardinality s we have the inequality

$$\Delta^{\mathbf{G}}(\mathbf{x^{(i_1)}}, \mathbf{x^{(i_2)}}, \ldots \mathbf{x^{(i_s)}}) < 2^s$$

then the following bound holds:

$$\Delta^{\mathbf{G}}(\mathbf{x^{(1)}}, \mathbf{x^{(2)}}, \ldots \mathbf{x^{(N)}}) < \Phi(s, N) = \sum_{k=0}^{s} \binom{N}{k}$$

If a function class induces all 2^r dichotomies on a sample of size r, then it is said to *shatter* the sample. Hence, we can upper-bound the number of dichotomies induced by $\mathbf{M}$ on a set of input vectors by finding the cardinality of the largest subset which $\mathbf{M}$ can shatter. It follows immediately from Theorem 4.3.1 that $\mathbf{M}$ can shatter a set of points if and only if no point dominates any other point in the set. We therefore need to find the cardinality r of the largest totally unordered subset of a partially ordered set, where two elements are ordered if one dominates the other. A totally unordered subset of a set is called an *antichain* [Stanley, 1986], and an antichain at least as large as all other antichains of the set is called a *maximal* antichain (note that a maximal antichain is not necessarily unique). A 17-element maximal antichain subset of 100 points in $\mathbf{R}^2$ generated from a pair of independent $\mathcal{N}(0,1)$ distributions is shown in Figure 5. r can be thought of as the "effective" VC dimension of $\mathbf{M}$ on a particular set of N points, and indeed, it would be the VC dimension if the input space were restricted to be only those N points.

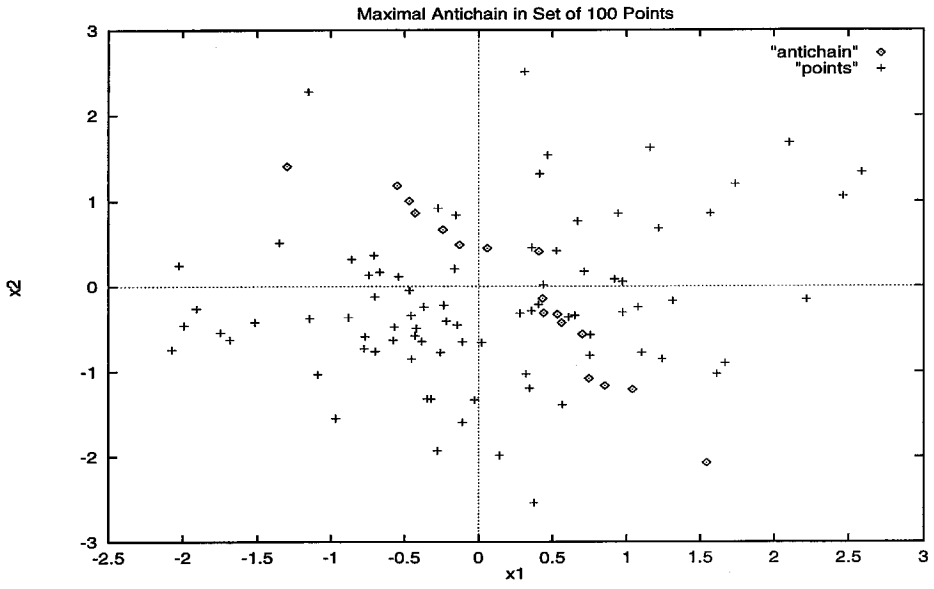

Figure 4.5: This set of 100 points has a maximal antichain of size 17.

Ford and Fulkerson show in [Ford and Fulkerson, 1962] that finding the cardinality of a maximal antichain may be solved by posing the problem in terms of an undirected bipartite graph. Let the graph consist of $2N$ vertices $a_1, a_2, \ldots a_N, b_1, b_2, \ldots b_N$, where an edge exists between a_i and b_j if and only if $\mathbf{x_i} \geq \mathbf{x_j}$. Define an *independent set* of edges to be a set where no pair of edges are incident to the same vertex. Ford and Fulkerson show that if U is a maximal antichain and E is a maximal independent set of edges, then

$$|U| + |E| = N$$

An $O(N^3)$ algorithm is given in [McHugh, 1990] for finding a maximal independent set of edges in a bipartite graph. This algorithm, known as the alternating path method, was implemented and used to find the the cardinality r of a maximal antichain for samples of size 5000 and $10,000$ in $\mathbf{R}^2$ generated from a joint normal distribution with 1 on the diagonal of the covariance matrix and ρ elsewhere. Since (**4.1**) involves $H(2N)$, this procedure gives us bounds for training sets of size $N = 2500$ and $N = 5000$, respectively. For each (N, ρ) pair, 10 samples of cardinality $2N$ were generated and the corresponding r was determined. For each sample, $\Delta^{\mathbf{M}}(\mathbf{x^{(1)}}, \mathbf{x^{(2)}}, \ldots \mathbf{x^{(2N)}})$ is bounded by $\Phi(r+1, 2N)$. $H(2N)$ is therefore bounded by $(\mathcal{E}[\Phi(r+1, 2N)])$. Table 4.3 displays the sample mean and standard deviation of r. The standard deviation of r is quite low, meaning that 10 samples suffice to estimate $\mathcal{E}[\Phi(r+1, 2N)]$ quite accurately.

n	ρ	mean r	ϵ
2500	0.0	134.8 ± 2.5	-
2500	0.5	101.7 ± 2.5	0.46
2500	0.9	66.5 ± 2.4	0.39
5000	0.0	190.4 ± 3.2	0.45
5000	0.5	147.0 ± 3.1	0.4
5000	0.9	93.5 ± 1.71	0.34

Table 4.3: Average size of largest antichain and smallest ϵ for which the VC bound (**4.1**) is less than 0.01.

This procedure yielded non-trivial bounds for correlated inputs in $\mathbf{R}^2$. For the sake of simplicity, we bound $H(2N)$ by the sample mean of $\Phi(r+1, 2N)$ plus $\frac{2\sigma}{\sqrt{10}}$, where σ is the sample standard deviation of $\Phi(r+1, 2N)$[1]. The last column in Table 4.3 shows the largest value of ϵ for which (**4.1**) evaluates to less than .01 (the exact choice of confidence level makes little difference, since the bound decreases very sharply from above unity to very small values over a very small range of ϵ). Note that high levels of covariance are not unrealistic given the types of real-world problems where the monotonicity constraint arises. Consider, for instance, the problem of approving credit card applicants on the basis of their salaries and current savings. One would expect salary and savings to be highly correlated.

These bounds may appear rather loose at first, but they are impressive compared with what would be obtained by bounding the VC dimension of a feedforward neural network. Let W be the total number of parameters (weights and thresholds) in a neural network and let U be the total number of units (hidden plus output). Then Baum and Haussler show in [Baum and Haussler, 1989] that $2Wlog_2(eU)$ is an upper-bound on the VC dimension of feedforward networks consisting of linear threshold units[2]. This bound is shown in table 4.4 for networks of 2 inputs and various numbers of hidden units.

hidden units	$2Wlog_2(eU)$
2	55
3	90
4	128
5	170
10	402
20	946

Table 4.4: Upper bound on VC dimension of feedforward neural networks of linear threshold units with 2 inputs.

[1]To be entirely rigorous, one would form a confidence interval for $\mathcal{E}[\Phi(r+1, 2N)]$ which would then be used in conjunction with the confidence interval provided by the VC bound.

[2]The use of this bound in practice may be optimistic, because the result treats hidden nodes as threshold units and therefore neglects the analog nature of the sigmoidal units most commonly used in real applications.

If a network of even fairly modest size is employed, many more examples would be needed to get the same bounds from bounding the VC dimension of the network than we get by bounding the VC entropy of **M**. A comparison of table 4.3 and table 4. indicates that for $N = 2500, \rho = 0.5$, we get a better bound by bounding **M** than we would by bounding the VC dimension of a network of 4 or more hidden units. For $\rho = 0.9$, the bound is better than it would be for a network of only 3 hidden units.

Consider a 10 hidden unit network. For $\epsilon = 0.46$, 9200 examples would be needed to get the same bound we get with 2500 examples and $\rho = 0.5$ here. 13,000 examples would be needed to get the same bound at $\epsilon = 0.4$ that we get for $n = 5000$, $\rho = 0.5$. The bound we obtain for $N = 5000$, $\rho = 0.9$ at $\epsilon = 0.34$ would require 19,400 examples.

If a neural network is constrained to obey monotonicity, as in chapters 2 and 3, then a bound on the entropy of **M** upper-bounds the entropy of the network with the constraint. Tables 4.3 and 4.4 demonstrate that this bound can be much tighter than the one obtained by employing the bound on growth function of the network.

The comparison of tables 4.3 and 4.4 also confirms our intuition that the flexibility and expressive power of **M** are fairly modest. This suspicion is supported by noting that the "effective" VC dimension of **M** for reasonable input distributions in $\mathbf{R}^2$ is comparable to the VC dimension of a small neural network with only a few hidden units.

4.5 Exponential Behavior of Capacity for Independent Inputs

The arguments in section 4.3 make it clear that analytical results regarding the capacity of **M** which apply independent of input distribution cannot be obtained, since capacity is highly distribution-dependent. If we assume independence between input variables, however, we can say something about how the capacity of **M** grows with d. Table 1 shows that capacity for independent inputs is low for low d but appears

to increase quickly as d becomes larger. An exponential relationship between d and capacity is suggested by the results. In this section, we prove that capacity is indeed exponential in d for independent inputs [3].

Define a sequence $q_1(x_1), q_2(x_1, x_2), q_3(x_1, x_2, x_3), \ldots$ of input densities on $\mathbf{R}^1, \mathbf{R}^2, \mathbf{R}^3, \ldots$ such that $q_i(x_1, \ldots, x_i) = q_{i-1}(x_1, \ldots, x_{i-1})p_i(x_i), i > 1$, where $p_1(x_1) = q_1(x_1)$ and $p_i(x_i)$ is the marginal density of input variable x_i. We will also require that for all i, $p_i(x_i)$ satisfy the condition that if x_{i_1} and x_{i_2} are two samples drawn independently from p_i, then $Pr\{x_{i_1} \geq x_{i_2}\} = Pr\{x_{i_2} \geq x_{i_1}\} = 0.5$. This condition is trivially satisfied by any smooth, continuous distribution (discrete distributions are ruled out, however). An example of a sequence satisfying these conditions is a sequence of i.i.d $\mathcal{N}(0, 1)$ variables. Note that we do not require the inputs to be identically distributed, however.

If $\mathbf{x}$ and $\mathbf{x}'$ are two input vectors in $\mathbf{R}^d$ drawn from $q_d(x_1, \ldots, x_d)$ then $Pr\{\mathbf{x} \geq \mathbf{x}' = 2^{-d}\}$. Now consider the probability $\Psi(N, d)$ that in a sample $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N})$ of size N drawn from q_d there exists a pair $(\mathbf{x_i}, \mathbf{x_j}), i \neq j$ such that $\mathbf{x_i} \geq \mathbf{x_j}$. In other words, $\Psi(N, d)$ is the probability that a sample of size N cannot be shattered by $\mathbf{M}$. Since there are $N^2 - N$ ordered pairs of distinct input vectors, we can use a union bound to find that $\Psi(N, d)$ is no greater than $N^2 2^{-d}$. Let $C(d)$ denote the capacity of $\mathbf{M}$ as a function of d for a sequence of input distributions satisfying the above requirements. Then the following theorem holds:

Theorem 4.5.1 The capacity $C(d)$ of $\mathbf{M}$ is $\Omega((\sqrt{2})^d)$ for independent inputs.

Proof Clearly, $P(N, d)$, the probability that a random dichotomy of N points drawn from q_d is monotonically separable, is lower-bounded by $1 - \Psi(N, d)$, the probability that all 2^N dichotomies are monotonically separable. Now suppose the theorem is false, i.e., suppose that $\forall c > 0$, there exists an D such that $\forall d \geq D$, $C(d) < c(\sqrt{2})^d$. From the definition of capacity we know that $P(C(d), d) = 0.5$. From the union bound, $\Psi(C(d), d) < C^2(d)2^{-d}$. Given any $\delta > 0$, we can choose c small enough and d large enough such that $\Psi(C(d), d) < \delta$. Take $\delta < 0.5$. Then we have $P(C(d), d) \geq 1 - \Psi(C(d), d) > 0.5$. This contradictions the assumption that

[3] The result presented in this section is due mostly to an anonymous referee.

$P(C(d), d) = 0.5$. Therefore, $C(d)$ must be $\Omega((\sqrt{2})^d)$ ■.

Capacity grows exponentially in d for independent inputs. This result should not be interpreted too pessimistically, however. The sorts of applications where monotonicity constraints arise (e.g. economic and medical diagnosis problems) typically involve relatively few inputs. In addition, there is often strong correlation between the variables, so the independence assumption does not hold in these cases.

4.6 Conclusion

We have shown that the capacity and entropy of **M** can be estimated computationally given a model of the input distribution. The bounds on the entropy lead to bounds on out-of-sample error which are tighter than those which would otherwise be obtained with neural networks with a very low number of hidden units. This led to the conclusion that monotonicity can be a very powerful constraint.

Future work may include extensions of the analytical results in section 4.5. Can we say something about how quickly the expected size of the maximal antichain increases with N and d? Correlated inputs also need to be considered. What level of correlation suffices for the capacity to grow polynomially with d?

Chapter 5 Generalization Bounds for Connected Function Classes

5.1 Introduction

Although the Vapnik-Chervonenkis analysis of learning systems has provided valuable insight into the process of learning, the VC bounds are rarely useful when evaluated numerically. The bounds typically require an unreasonably large number of examples in order to evaluate to less than unity, particularly when the VC dimension is high, as is typically the case when a model with many parameters is being employed in a high dimensional input space. Yet the out-of-sample error of a learning system is often observed to be quite low in situations where the VC bound promises nothing.

One possible explanation for this discrepancy is that the VC analysis is not a sufficiently detailed characterization of the flexibility of a function class. The VC bound is computed from the function class's growth function $G(N)$, which is defined as the maximum number of dichotomies of N input vectors. Thus, through the eyes of the VC bound, all function classes with the same growth function are equivalent in terms of generalization behavior. Aspects of a function class other than its growth function may play a role, however. It may be that the function classes used in practice have an additional property not accounted for by the growth function which results in good generalization with fewer examples than the VC analysis would suggest. If this conjecture is correct, it would help explain the apparent looseness of the VC bound. The VC bound must hold for all function classes which implement at most $G(N)$ dichotomies on N input vectors. Consider a hypothetical pair of function classes, each of which induce $G(N)$ dichotomies on N examples. If one function class possesses an additional characteristic which leads it to require fewer examples for good generalization than the other class needs, then clearly the VC bound will be

unnecessarily pessimistic for the first class, since it must hold true for the second class as well as the first. In this chapter, such a characteristic is described and utilized in deriving an improved bound which holds for most function classes of practical interest.

5.2 The Connectedness Theorem

The general idea of examining the set of dichotomies induced by a function class on a set of points seems like an intuitively valid way of measuring the power of the class. The growth function, however, characterizes a set of dichotomies in a very simple way- it merely counts the number of elements in the set. More detailed descriptions of dichotomy sets can easily be imagined. One might also take into account the variety or diversity of the dichotomies, for instance. A function class which induces a set of dichotomies which is clustered together might be thought to be less flexible than another class which implements a set of dichotomies which are dissimilar and dispersed, even if the cardinality of the two sets is equal. Consider a hypothetical situation, displayed in Figure 5.1, where we have function classes F_1 and F_2, each of which induce 5 dichotomies of 30 input vectors. The VC analysis does not distinguish between F_1 and F_2, since they induce the same number of dichotomies. Intuitively, however, it would seem that F_2 is considerably more flexible than F_1. F_1 has no flexibility at all on 26 of the 30 input vectors- it must always label these inputs the same way. F_2, on the other hand, can choose between 5 completely dissimilar sets of classifications. The contrast between similar and dissimilar sets of dichotomies becomes even more stark as the number of data points N becomes large. Consider a pair of dichotomies of $N = 10,000$ points which agree on $9,999$ points and differ on only 1. The two dichotomies are virtually the same and it would seem that they should only count as effectively 1 dichotomy (or perhaps $1 + \epsilon$ dichotomies). Yet the VC bounds count this pair as two dichotomies, just the same as if they disagreed on $5,000$ points. Of course, these comments are relevant only if it is true that the dichotomies induced by function classes used by practitioners are somehow clustered together. If it could be demonstrated that function classes in common usage induce

sets of dichotomies which are not very diverse, then this property can perhaps be used to tighten the VC bound.

Dichotomies induced by $\mathbf{F}_1$:

$\mathbf{d}_1$: 0 1 1 0 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 1 1 1 0 1 0 1 0 1 1
$\mathbf{d}_2$: 0 1 1 0 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 1 1 1 0 1 0 1 0 1 0
$\mathbf{d}_3$: 0 1 1 0 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 1 1 1 0 1 0 1 0 0 0
$\mathbf{d}_4$: 0 1 1 0 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 1 1 1 0 1 0 1 1 0 0
$\mathbf{d}_5$: 0 1 1 0 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 1 1 1 0 1 0 0 1 0 0

Dichotomies induced by $\mathbf{F}_2$:

$\mathbf{d}_1$: 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 1 1 0 1 0 1
$\mathbf{d}_2$: 1 1 1 0 1 1 0 1 0 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 1
$\mathbf{d}_3$: 0 0 1 0 1 0 0 0 1 1 0 1 1 1 0 1 1 0 1 1 1 0 0 1 0 0 0 1 1 0
$\mathbf{d}_4$: 1 0 0 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 1 0 1 0 1 1
$\mathbf{d}_5$: 1 0 1 0 0 0 1 1 1 1 0 1 0 1 0 1 1 1 0 0 1 0 0 1 0 0 1 1 1 1

Figure 5.1: F_1 induces a very clustered set of dichotomies, while the dichotomy set induced by F_2 is very diverse.

We will show here that most function classes used in practice do indeed induce dichotomy sets which are clustered. Our theorem will apply to function classes of the form

$$\theta(u(\mathbf{w}, \mathbf{x}))$$

where

$$\mathbf{w} \in \mathbb{R}^p, \quad \mathbf{x} \in \mathbb{R}^d$$

$$u(\mathbf{w}, \mathbf{x}) : \mathbb{R}^p \times \mathbb{R}^d \to \mathbb{R}$$

$$\theta(u) = \begin{cases} 1 & u \geq 0 \\ 0 & u < 0 \end{cases}$$

The vector $\mathbf{w}$ is the set of adjustable parameters of the model, while $\mathbf{x}$ is the input vector or feature vector. We require u to be a continuous function of $\mathbf{w}$ and $\mathbf{x}$ and that all first and second derivatives of u with respect to $\mathbf{w}$ and $\mathbf{x}$ exist. One additional property is also needed for the theorem below to hold. Let input vectors $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N)}$ be drawn from a density $P(\mathbf{x})$. Let $u_n(\mathbf{w})$ denote $u(\mathbf{w}, \mathbf{x}^{(n)})$. The

gradient of u with respect to $\mathbf{w}$ at the point $\mathbf{x^{(n)}}$ is given by

$$\nabla u_n(\mathbf{w}) = \left[\frac{\partial u(\mathbf{w}, \mathbf{x^{(n)}})}{\partial w_1} \quad \frac{\partial u(\mathbf{w}, \mathbf{x^{(n)}})}{\partial w_2} \cdots \frac{\partial u(\mathbf{w}, \mathbf{x^{(n)}})}{\partial w_p}\right]^T$$

Then we require u and P to be such that

$$Pr\{\exists \mathbf{w},\ n_1,\ n_2 \mid \nabla u_{n_1}(\mathbf{w}) \propto \nabla u_{n_2}(\mathbf{w})\} = 0$$

This condition is likely to be satisfied by most smooth input distributions, assuming that u depends smoothly on both the inputs and the parameters. Later in the chapter, we provide short proofs which show that this condition is satisfied by many common machine learning models given smooth input distributions over continuous spaces.

We also need some basic definitions from graph theory in order to state the theorem. An undirected graph is simply a set of vertices V connected by a set of edges E, where the edges do not have directions associated with them. A *path* is a sequence $\{v_1, v_2, \ldots v_m\}$ of vertices such that $\forall 1 \leq i \leq m-1$, there is an edge between v_i and v_{i+1}. We say that a graph is *connected* if every pair of vertices is connected by a path. Let $D = \{\mathbf{d^{(1)}}, \mathbf{d^{(2)}}, \ldots \mathbf{d^{(\Delta)}}\}$ be a set of dichotomies. Define the *dichotomy graph* of D to be an undirected graph where each vertex v_m corresponds to the dichotomy $\mathbf{d^{(m)}}$ and an edge exists between v_{m_1} and v_{m_2} if and only if the Hamming distance (denoted by $||\mathbf{d^{(m_1)}} - \mathbf{d^{(m_2)}}||$) between $\mathbf{d^{(m_1)}}$ and $\mathbf{d^{(m_2)}}$ is 1. In other words, an edge exists between two vertices if and only if the corresponding dichotomies classify exactly $N-1$ of the N input vectors the same way. We say that D is connected if its dichotomy graph is connected.

We are now in a position to state the theorem which quantifies the sense in which dichotomy sets induced by most function classes used in practice are clustered.

Theorem 5.2.1 Let $\theta(u(\mathbf{w}, \mathbf{x}))$ be a function class and $P(\mathbf{x})$ be an input distribution such that u and P satisfy the above requirements. Let $D = \{\mathbf{d^{(1)}}, \mathbf{d^{(2)}}, \ldots \mathbf{d^{(\Delta)}}\}$ be the set of dichotomies induced by $\theta(u(\mathbf{w}, \mathbf{x}))$ on a set of input vectors $\mathbf{x^{(1)}}, \mathbf{x^{(2)}}, \ldots \mathbf{x^{(N)}}$ drawn independently from P. Then with probability 1, D is connected.

$\mathbf{d_1}$: 1 1 1 1 1
$\mathbf{d_2}$: 1 1 1 1 0
$\mathbf{d_3}$: 1 1 1 0 0
$\mathbf{d_4}$: 1 1 1 0 1
$\mathbf{d_5}$: 0 1 1 0 1
$\mathbf{d_6}$: 0 1 1 1 1

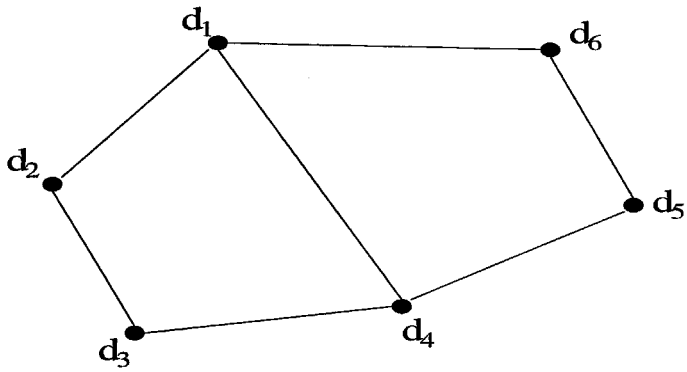

Figure 5.2: A set of 6 dichotomies of 5 points and its corresponding dichotomy graph

Proof: To say that the dichotomy graph of D is connected is to say that given any pair of dichotomies $\mathbf{d}^{(\mathbf{m_1})}, \mathbf{d}^{(\mathbf{m_2})} \in D$, there exists a sequence of dichotomies $\mathbf{d}^{(\mathbf{k_1})}, \mathbf{d}^{(\mathbf{k_2})}, \ldots \mathbf{d}^{(\mathbf{k_K})} \in D$ such that

$$||\mathbf{d}^{(\mathbf{m_1})} - \mathbf{d}^{(\mathbf{k_1})}|| = 1$$

$$\forall 1 \leq i \leq K-1 \quad ||\mathbf{d}^{(\mathbf{k_i})} - \mathbf{d}^{(\mathbf{k_{i+1}})}|| = 1$$

$$||\mathbf{d}^{(\mathbf{k_K})} - \mathbf{d}^{(\mathbf{m_2})}|| = 1$$

Let $\mathbf{w}^{(\mathbf{m_1})}$ be a parameter vector which implements dichotomy $\mathbf{d}^{(\mathbf{m_1})}$, i.e.,

$$d_j{}^{(m_1)} = 1 \Rightarrow u(\mathbf{w}^{(\mathbf{m_1})}, \mathbf{x}^{(\mathbf{j})}) > 0$$

$$d_j{}^{(m_1)} = 0 \Rightarrow u(\mathbf{w}^{(\mathbf{m_1})}, \mathbf{x}^{(\mathbf{j})}) < 0$$

Similarly, let $\mathbf{w}^{(\mathbf{m_2})}$ implement dichotomy $\mathbf{d}^{(\mathbf{m_2})}$. Consider the convex combination of these two parameter vectors:

$$\mathbf{w}^{(\alpha)} = (1-\alpha)\mathbf{w}^{(\mathbf{m_1})} + \alpha \mathbf{w}^{(\mathbf{m_2})}$$

Define

$$u_n(\mathbf{w}^{(\alpha)}) = u(\mathbf{w}^{(\alpha)}, \mathbf{x}^{(\mathbf{n})})$$

We know from the continuity of u that any time the classification of $\mathbf{x}^{(\mathbf{n})}$ changes as α is varied, we must pass through a point at which $u_n(\mathbf{w}^{(\alpha)}) = 0$. If for all values of α, $u_n(\mathbf{w}^{(\alpha)}) = 0$ for at most one integer n, then the theorem follows immediately- the sequence of dichotomies generated by varying α corresponds to a path from the vertex v_{m_1} to the vertex v_{m_2} in the dichotomy graph of D. Thus, we need only consider the case where there exist parameter vectors $\mathbf{w}^{(\alpha^*)}$ along the line between $\mathbf{w}^{(\mathbf{m_1})}$ and $\mathbf{w}^{(\mathbf{m_2})}$ such that the classification of multiple input vectors changes simultaneously, i.e., such that $u_n(\mathbf{w}^{(\alpha^*)}) = 0$ for more than one value of n. Given any such α^*, we will construct an alternate path through parameter space which avoids $\mathbf{w}^{(\alpha^*)}$. Assume that at α^* there are L examples whose classification changes, i.e.

$$u_{n_1}(\mathbf{w}^{(\alpha^*)}) = 0, \ldots u_{n_L}(\mathbf{w}^{(\alpha^*)}) = 0$$

Given an α^*, we must construct an alternate path through parameter space such that the classifications of $\mathbf{x}^{(\mathbf{n_1})}, \ldots \mathbf{x}^{(\mathbf{n_L})}$ change one at a time. Consider $\mathbf{w}^{(\alpha^* - \delta_\mathbf{a})}$, where δ_a is chosen to be positive but small enough that there are no classification changes between $\alpha^* - \delta_a$ and α. Define $\mathbf{w}^{(\alpha^* + \delta_\mathbf{b})}$ similarly. Let $\mathbf{d}^{(\mathbf{a})}$ denote the dichotomy implemented by $\theta(u(\mathbf{w}^{(\alpha^* - \delta_\mathbf{a})}, \mathbf{x}))$ and let $\mathbf{d}^{(\mathbf{b})}$ denote the dichotomy implemented by $\theta(u(\mathbf{w}^{(\alpha^* + \delta_\mathbf{b})}, \mathbf{x}))$. Hence

$$||\mathbf{d}^{(\mathbf{a})} - \mathbf{d}^{(\mathbf{b})}|| = L$$

We can conclude from the continuity of u that there exist spheres W_a and W_b of non-zero volume around $\mathbf{w}^{(\alpha^* - \delta_\mathbf{a})}$ and $\mathbf{w}^{(\alpha^* + \delta_\mathbf{b})}$, respectively, such that every parameter vector within W_a implements $\mathbf{d}^{(\mathbf{a})}$ and every parameter vector within W_b implements $\mathbf{d}^{(\mathbf{b})}$. Let r_a be the radius of W_a and r_b be the radius of W_b.

The strategy of the construction will be to show that there must exist a pair of points $\mathbf{w_a} \in W_a$ and $\mathbf{w_b} \in W_b$ such that along the line from $\mathbf{w_a}$ to $\mathbf{w_b}$, the dichotomies generated change one bit at a time. To demonstrate this claim, we rely

on the fact that for each n_l the manifold in parameter space defined by $u_{n_l}(\mathbf{w}) = 0$ is approximately planar, locally around $\mathbf{w}^{(\alpha^*)}$. A second-order Taylor expansion gives

$$u_{n_l}(\mathbf{w}^{(\alpha^*)} + \Delta\mathbf{w}) = u_{n_l}(\mathbf{w}^{(\alpha^*)}) + \nabla u_{n_l}(\mathbf{w}^{(\alpha^*)})^T \Delta\mathbf{w} + \frac{1}{2}\Delta\mathbf{w}^T H \Delta\mathbf{w}$$

where H is the Hessian of u_{n_l} at some point between $\mathbf{w}^{(\alpha^*)}$ and $\mathbf{w}^{(\alpha^*)} + \Delta\mathbf{w}$. For sufficiently small $||\Delta\mathbf{w}||$, the decision boundary for $\mathbf{x}^{(\mathbf{n_l})}$ is approximately given by the $p-1$ dimensional hyperplane

$$\nabla u_{n_l}(\mathbf{w}^{(\alpha^*)})^T \Delta\mathbf{w} = 0$$

Recall that one of the assumptions made at the outset of the proof is that the probability that the gradient vectors for two different input vectors are collinear is 0. Therefore, with probability 1 each example $\mathbf{x}^{(\mathbf{n_l})}$ is associated with a distinct hyperplane $\nabla {u_{n_l}}^T \Delta\mathbf{w} = 0$ and the intersection of any two such hyperplanes is a $p-2$ dimensional hyperplane.

Lemma 5.2.1: There exist points $\mathbf{w_a} \in W_a$ and $\mathbf{w_b} \in W_b$ such that $\forall 0 \leq \beta \leq 1$, $\beta\mathbf{w_a} + (1-\beta)\mathbf{w_b}$ satisfies $\nabla {u_{n_l}}^T \Delta\mathbf{w} = 0$ for at most 1 value of l, where l ranges from 1 to L.

Proof: We will first show that there exist non-zero volumes ${W_a}^{(1)}$ and ${W_b}^{(1)}$ such that any line from a point in ${W_a}^{(1)}$ to a point in ${W_b}^{(1)}$ avoids the intersection of the two hyperplanes

$$\nabla {u_{n_1}}^T \Delta\mathbf{w} = 0, \quad \nabla {u_{n_2}}^T \Delta\mathbf{w} = 0$$

First, we need to find points $\mathbf{w_a^{(1)}}$ and $\mathbf{w_b^{(1)}}$ such that the line between them avoids the intersection of the two hyperplanes. Define $\Delta\mathbf{w_a^{(1)}} = \mathbf{w_a^{(1)}} - \mathbf{w}^{\alpha^*}$ and $\Delta\mathbf{w_b^{(1)}} = \mathbf{w_b^{(1)}} - \mathbf{w}^{\alpha^*}$. Then simple algebra reveals that the line $\beta{\mathbf{w_a}}^{(1)} + (1-\beta){\mathbf{w_b}}^{(1)}$ crosses the hyperplane $\nabla {u_{n_1}}^T \Delta\mathbf{w} = 0$ at

$$\beta_1 = \frac{-\nabla {u_{n_1}}^T \Delta\mathbf{w_b^{(1)}}}{\nabla {u_{n_1}}^T \Delta\mathbf{w_a^{(1)}} - \nabla {u_{n_1}}^T \Delta\mathbf{w_b^{(1)}}}$$

while it crosses $\nabla {u_{n_2}}^T \Delta \mathbf{w} = 0$ at

$$\beta_2 = \frac{-\nabla {u_{n_2}}^T \Delta \mathbf{w_b^{(1)}}}{\nabla {u_{n_2}}^T \Delta \mathbf{w_a^{(1)}} - \nabla {u_{n_2}}^T \Delta \mathbf{w_b^{(1)}}}$$

If we were to pick

$$\mathbf{w_a}^{(1)} = \mathbf{w}^{(\alpha^* - \delta_a)}$$

$$\mathbf{w_b}^{(1)} = \mathbf{w}^{(\alpha^* + \delta_b)}$$

then the line between the two points would coincide with the original line between $\mathbf{w_{m_1}}$ and $\mathbf{w_{m_2}}$, meaning that $\beta_1 = \beta_2$ and that the line crosses both hyperplanes at $\mathbf{w}^{(\alpha^*)}$, by assumption. Therefore, this is not an acceptable choice. Suppose, however, we add a perturbation to $\mathbf{w_b}^{(1)}$, i.e.

$$\mathbf{w_b}^{(1)} = \mathbf{w}^{(\alpha^* + \delta_b)} + \eta \mathbf{\Delta v}$$

where

$$\mathbf{\Delta v} = \nabla u_{n_2} - \frac{{\nabla u_{n_1}}^T \nabla u_{n_2}}{{\nabla u_{n_1}}^T \nabla u_{n_1}} \nabla u_{n_1}$$

and η is chosen such that $||\eta \mathbf{\Delta v}|| = \frac{r_b}{2}$. $\mathbf{\Delta v}$ is orthogonal to ∇u_{n_1}, which means that β_1 remains unchanged while β_2 is altered by a non-zero amount, i.e., the two hyperplanes are crossed at different points along the line. Because β_1 and β_2 depend continuously on $\mathbf{w_a}^{(1)}$ and $\mathbf{w_b}^{(1)}$, there must exist non-zero volumes ${W_a}^{(1)}$ and ${W_b}^{(1)}$ around $\mathbf{w_a}^{(1)}$ and $\mathbf{w_b}^{(1)}$, respectively, such that a line from any point in ${W_a}^{(1)}$ to any point in ${W_b}^{(1)}$ intersects the two hyperplanes at different points.

The same argument may be used again to show that there exist volumes ${W_a}^{(2)} \subseteq {W_a}^{(1)}$ and ${W_b}^{(2)} \subseteq {W_b}^{(1)}$ such that a line from any point in ${W_a}^{(2)}$ to any point in ${W_b}^{(2)}$ avoids the intersection between $\nabla {u_{n_1}}^T \Delta \mathbf{w} = 0$ and $\nabla {u_{n_3}}^T \Delta \mathbf{w} = 0$. In fact, the argument may be repeatedly applied $\binom{L}{2}$ times for each pair of distinct hyperplanes, yielding two volumes such that any line drawn between them avoids all intersections of hyperplanes. The lemma has therefore been demonstrated ■.

Let $\mathbf{w^{(s)}} \in W_a$ and $\mathbf{w^{(f)}} \in W_b$ be the starting and final points of a line which

crosses the hyperplanes one at a time. We will now show that if δ_a, δ_b, r_a and r_b are chosen to be sufficiently small, then the classification of each input vector $\mathbf{x^{(n_l)}}$ is determined by the function $\theta(\nabla {u_{n_l}}^T \Delta \mathbf{w}^{(\beta)})$, where

$$\Delta \mathbf{w}^{(\beta)} = \mathbf{w}^{(\beta)} - \mathbf{w}^{(\alpha^*)}$$

$$\mathbf{w}^{(\beta)} = \beta \mathbf{w^{(s)}} + (1-\beta)\mathbf{w^{(f)}}$$

Recall again Taylor's theorem:

$$u_{n_l}(\mathbf{w}^{(\beta)}) = u_{n_l}(\mathbf{w}^{(\alpha^*)}) + \nabla u_{n_l}^T \Delta \mathbf{w}^{(\beta)} + \frac{1}{2} \Delta \mathbf{w}^{(\beta)^T} H_{n_l} \Delta \mathbf{w}^{(\beta)}$$

where H_{n_l} is the p by p Hessian matrix of u_{n_l} evaluated at some point between $\mathbf{w}^{(\alpha^*)}$ and $\mathbf{w}^{(\beta)}$. Define $\lambda_{\beta,l}$ to be the largest magnitude of any eigenvalue of H_{n_l} at any point between $\mathbf{w}^{(\alpha^*)}$ and $\mathbf{w}^{(\beta)}$. Then it is not hard to show that

$$|\Delta \mathbf{w}^{(\beta)^T} H_{n_l} \Delta \mathbf{w}^{(\beta)}| \leq \lambda_{\beta,l} ||\Delta \mathbf{w}^{(\beta)}||^2$$

We also have the identity

$$\nabla u_{n_l}^T \Delta \mathbf{w}^{(\beta)} = ||\nabla u_{n_l}|| ||\Delta \mathbf{w}^{(\beta)}|| cos(\phi)$$

where ϕ is the angle between ∇u_{n_l} and $\Delta \mathbf{w}^{(\beta)}$.

It will be convenient to define some auxiliary variables. Let $y_{n_l} = 1$ if ${d_{n_l}}^{(a)} = 1$ and let $y_{n_l} = -1$ if ${d_{n_l}}^{(a)} = 0$. Then, as we move along a line from a point in $\mathbf{W_a}$ to a point in $\mathbf{W_b}$, the classification of $\mathbf{x_{(n_l)}}$ changes when $y_{n_l} u_{n_l}(\mathbf{w}) < 0$.

The numbering of the examples $\mathbf{x}^{(n_1)}, \ldots, \mathbf{x}^{(n_L)}$ is arbitrary, so assume for the sake of simplicity that they are numbered in the order in which $\Delta \mathbf{w}^{(\beta)}$ passes through their respective hyperplanes. From the lemma above, we know that

$$\forall 0 \leq i \leq L, \quad \exists \beta_i$$

such that

$$y_{n_l} \nabla u_{n_l}{}^T \Delta \mathbf{w}^{(\beta_i)} < 0, \quad l \leq i$$

$$y_{n_l} \nabla u_{n_l}{}^T \Delta \mathbf{w}^{(\beta_i)} > 0, \quad l > i$$

Define

$$c_{min} = \min_{i,l} cos(\phi_{i,l})$$

where $\phi_{i,l}$ is the angle between $\Delta \mathbf{w}^{(\beta_i)}$ and ∇u_{n_l}. Let

$$\lambda_{max} = \max_{i,l} \lambda_{\beta_i,l}$$

and let

$$\nabla_{min} = \min_{l} ||\nabla u_{n_l}||$$

Then if

$$\forall i, \ ||\Delta \mathbf{w}^{(\beta_i)}|| < \frac{\nabla_{min} c_{min}}{\lambda_{max}}$$

then

$$\forall i, \ l, \ \theta(\nabla u_{n_l}{}^T \Delta \mathbf{w}^{(\beta_i)}) = \theta(u(\mathbf{w}_{\beta_i}, \mathbf{x}^{(\mathbf{n_l})}))$$

and we have established that the line between $\mathbf{w}^{(\mathbf{s})}$ and $\mathbf{w}^{(\mathbf{f})}$ generates a sequence of dichotomies which changes one bit at a time. It remains only to show that we can choose $||\Delta \mathbf{w}^{(\beta_i)}||$ to be sufficiently small. Recall that $\mathbf{w}^{(s)} \in W_a$ and $\mathbf{w}^{(f)} \in W_b$. W_a has center $\mathbf{w}^{(\alpha^* - \delta_a)}$ and radius r_a and W_b has center $\mathbf{w}^{(\alpha^* + \delta_b)}$ and radius r_b. Then the distances from $\mathbf{w}^{(s)}$ and $\mathbf{w}^{(f)}$ to $\mathbf{w}^{(\alpha^*)}$ are at most

$$(\max(\delta_a, \delta_b) + \max(r_a, r_b))||\mathbf{w}^{(m_1)} - \mathbf{w}^{(m_2)}||$$

We are free to choose δ_a, δ_b, r_a, and r_b to be as small as we like. $\mathbf{w}^{(\beta)}$ is a convex combination of $\mathbf{w}^{(s)}$ and $\mathbf{w}^{(f)}$ and therefore, for any β, it can be no further from $\mathbf{w}^{(\alpha^*)}$ than the further of $\mathbf{w}^{(s)}$ and $\mathbf{w}^{(f)}$. Hence we can make $||\Delta \mathbf{w}^{(\beta)}||$ as small as is necessary to guarantee that the path from $\mathbf{w}^{(s)}$ to $\mathbf{w}^{(f)}$ generates a sequence of dichotomies which change one bit at a time.

There are potentially many places along the line from $\mathbf{w}^{(\mathbf{m_1})}$ to $\mathbf{w}^{(\mathbf{m_2})}$ where the classifications of several input vectors change simultaneously. The above construction can be repeated in each such instance, resulting in a path through parameter space from $\mathbf{w}^{(\mathbf{m_1})}$ to $\mathbf{w}^{(\mathbf{m_2})}$ generating a sequence of dichotomies which corresponds to a path from v_{m_1} to v_{m_2} in the dichotomy graph of D. Hence, the proof of the theorem is complete ■.

5.3 Connected Function Classes

We now show that the conditions required for Theorem 5.2.1 to hold are satisfied by some well-known and useful machine learning models. The assumption will be made in each case that the probability distribution $P(\mathbf{x})$ is such that with probability 1, any finite set of input vectors are in *general position*, i.e., any subset of d or fewer vectors is linearly independent. We will refer to a model which induces a connected dichotomy set with probability 1 under these conditions as a *connected function class.*

Perhaps the simplest model in usage is the linear threshold model, a.k.a the perceptron.

Proposition 5.3.1 *The linear threshold model is a connected function class.*

Proof A perceptron is a model of the form $\theta(u(\mathbf{w}, \mathbf{x}))$ where $u(\mathbf{w}, \mathbf{x})) = \mathbf{w}^{\mathbf{T}}\mathbf{x} - b$. $\nabla u(\mathbf{w})$ is simply $\mathbf{x}$. Thus, the satisfaction of the requirements for Theorem 5.2.1 follow immediately from the general position assumption ■.

Similar reasoning can be employed to show that the connectedness theorem applies to 2-layer sigmoidal neural networks.

Proposition 5.3.2 *A 2-layer sigmoidal feedforward network with at least 2 hidden units is a connected function class .*

Proof: A 2-layer network computes a function of the form

$$\theta(u(\mathbf{x}))$$

where

$$u(\mathbf{x}) = \mathbf{v}^T\mathbf{h}(\mathbf{x}) - \beta$$

and

$$h_i = s(\mathbf{w_i}^T\mathbf{x} - b_i)$$

s is a monotonically increasing function, e.g., the hyperbolic tangent function. $\frac{\partial u}{\partial w_{ij}}$ is given by

$$v_i s'(\mathbf{w_i}^T\mathbf{x})x_j$$

The monotonicity of s implies that s' is always non-zero. Assume for some i that $v_i \neq 0$. Suppose that the theorem is false. Define $u_n = u(\mathbf{x^{(n)}})$. Then for some pair $(\mathbf{x^{(1)}}, \mathbf{x^{(2)}})$ and some constant α we have

$$\frac{\partial u_1}{\partial \mathbf{w_i}} = \alpha \frac{\partial u_2}{\partial \mathbf{w_i}}$$

$$v_i s'(\mathbf{w_i}^T\mathbf{x^{(1)}})\mathbf{x^{(1)}} = \alpha v_i s'(\mathbf{w_i}^T\mathbf{x^{(2)}})\mathbf{x^{(2)}}$$

$$\mathbf{x^{(1)}} \propto \mathbf{x^{(2)}}$$

which contradicts the general position assumption. The only case not covered by this argument is the situation where $\forall i,\ v_i = 0$. This area of parameter space is a hyperplane of dimension at most $p-2$, since we have assumed at least 2 hidden units. By Lemma 5.2.1, it can be avoided in the construction of a path through parameter space corresponding to a path in the dichotomy graph ■.

Radial basis function networks [Bishop, 1995] are another popular machine learning model.

Proposition 5.3.3 *A radial basis function network with at least two hidden units is a connected function class.*

Proof: A radial basis function network computes the function $\theta(u(\mathbf{x}))$ where

$$u(\mathbf{x}) = \sum_{j=1}^{J} w_j e^{-\frac{||\mathbf{x}-\mu_j||^2}{2\sigma_j{}^2}}$$

where w_j, μ_j and σ_j are all adjustable parameters. Define $\mathbf{p_j}$ as the vector of all parameters associated with unit j of the network, i.e.,

$$\mathbf{p_j} = [w_j \ \mu_{j1} \ \ldots \mu_{jd} \ldots \sigma_j]$$

Define $u_1(\mathbf{x}) = u(\mathbf{x^{(1)}})$ and $u_2(\mathbf{x}) = u(\mathbf{x^{(2)}})$. Now suppose the proposition is false. Then with nonzero probability the gradient of u_1 with respect to $\mathbf{p_j}$ is equal to the gradient of u_2 with respect to $\mathbf{p_j}$. We have

$$\frac{\partial u_1}{\partial \mu_\mathbf{j}} = -\frac{w_j}{{\sigma_j}^2} e^{-\frac{||\mathbf{x^{(1)}} - \mu_j||^2}{2{\sigma_j}^2}} (\mathbf{x^{(1)}} - \mu_j)$$

$$\frac{\partial u_2}{\partial \mu_j} = -\frac{w_j}{{\sigma_j}^2} e^{-\frac{||\mathbf{x^{(2)}} - \mu_j||^2}{2{\sigma_j}^2}} (\mathbf{x^{(2)}} - \mu_j)$$

Clearly we need $\mathbf{x^{(1)}} - \mu_j = \alpha(\mathbf{x^{(2)}} - \mu_j)$ for some constant α. Then

$$\frac{\partial u_2}{\partial \mu_j} = \alpha e^{\frac{||\mathbf{x^{(1)}} - \mu_j||^2}{2{\sigma_j}^2} - \frac{||\mathbf{x^{(2)}} - \mu_j||^2}{2{\sigma_j}^2}} \frac{\partial u_1}{\partial \mu_j}$$

Now consider the partial derivative with respect to the parameter w_j.

$$\frac{\partial u_2}{\partial w_j} = e^{-\frac{||\mathbf{x^{(2)}} - \mu_j||^2}{2{\sigma_j}^2}} = e^{\frac{||\mathbf{x^{(1)}} - \mu_j||^2}{2{\sigma_j}^2} - \frac{||\mathbf{x^{(2)}} - \mu_j||^2}{2{\sigma_j}^2}} \frac{\partial u_1}{\partial w_j}$$

In order for the gradient of u_1 with respect to $\mathbf{p_j}$ to be proportional to the gradient of u_2 with respect to $\mathbf{p_j}$, the ratio of $\frac{\partial u_1}{\partial w_j}$ to $\frac{\partial u_2}{\partial w_j}$ must be the same as the ratio of $\frac{\partial u_1}{\partial \mu_j}$ to $\frac{\partial u_2}{\partial \mu_j}$. Then $\alpha = 1$, implying that $\mathbf{x^{(1)}} = \mathbf{x^{(2)}}$. This contradicts the general position assumption.

As was the case for sigmoidal neural nets, we need to address the situation where $\forall j, \ w_j = 0$ separately. Once again, we know that area of parameter space can be avoided from Lemma 5.2.1. since there are at least two hidden units and therefore the area is a hyperplane of dimension at most $p - 2$ ■.

It can also be shown that the class **M** of monotonically increasing functions is

connected. Connectedness does not follow from Theorem 5.2.1, though. In this case, a different line of reasoning can be used

Proposition 5.3.4 *The class* **M** *of monotonically increasing functions is a connected function class.*

Proof: Let $\mathbf{d}$ and $\mathbf{d}'$ be two monotonically separable dichotomies of N points $\{\mathbf{x}^{(1)} \ldots \mathbf{x}^{(N)}\}$. Define X as the set of input vectors labelled class 1 by $\mathbf{d}$ and define X' as the set labelled class 1 by $\mathbf{d}'$. Recall that it was shown in Chapter 4 that any monotonically separable dichotomy can be implemented by a function $m \in M$ defined as follows: Classify as class 1 only those points in input space which dominate at least one input vector labelled positively in the dichotomy. Let m be the function which classifies as 1 only those input vectors which dominate some vector in X, and define m' analogously. Then m implements $\mathbf{d}$ and m' implements $\mathbf{d}'$.

Consider the set of input vectors which belong to X but not to X'. Because domination is a partial ordering, there must exist some input vector $\mathbf{x_{min}}$ in this set which does not dominate any other vector in the set. Suppose we remove this vector from X. Now m classifies $\mathbf{x_{min}}$ as 0 but otherwise the dichotomy it implements is the same as $\mathbf{d}$. Now we repeat this removal procedure by finding a new $\mathbf{x_{min}}$ from the remaining vectors in X. This procedure can be iterated to produce a sequence of monotonically separable dichotomies which change one bit at a time, terminating in a dichotomy $\mathbf{d_{mid}}$ which classifies as 1 only those vectors classified as 1 by both $\mathbf{d}$ and $\mathbf{d}'$. Now consider the set of vectors which belong to X' but not to X. There must exist some vector $\mathbf{x_{max}}$ in this set which is not dominated by any other vector. If we add this vector to X, then m now classifies $\mathbf{x_{max}}$ as 1 but otherwise the dichotomy it implements is the same as $\mathbf{d_{mid}}$. We can repeat this process, generating a sequence of monotonically separable dichotomies which change one bit at a time , terminating in $\mathbf{d}'$ ■.

5.4 VC Bound for Connected Function Classes

In this section, we derive an improvement to the VC bound which holds for all connected function classes. Let F be a set of functions mapping $\mathbf{R}^d$ to $\{0,1\}$. We draw N examples $(\mathbf{x^{(1)}}, y_1) \ldots (\mathbf{x^{(N)}}, y_N)$ independently from some probability distribution on $\mathbf{R}^d \times \{0,1\}$. For $f \in F$, define

$$\pi_f \equiv Pr[y \neq f(\mathbf{x})]$$

and

$$\nu_f \equiv \frac{1}{N} \sum_{n=1}^{N} |y_n - f(\mathbf{x^{(n)}})|$$

π_f is the true error rate of f, while ν_f is the error rate on the n examples at hand. We wish to bound the quantity

$$Pr\{\sup_{f \in F} |\pi_f - \nu_f| > \epsilon\},$$

the probability that the worst case discrepancy between π_f and ν_f is greater than ϵ.

5.4.1 Review of the Standard VC bound

This section gives a sketch of the proof of the currently existing VC bound. This review will help make clear the reason that the connectedness theorem should enable us to tighten the bound.

The distribution-independent Vapnik-Chervonenkis bound is given by

$$Pr\{\sup_{f \in F} |\pi_f - \nu_f| > \epsilon\} < 6G(2N)e^{-(\epsilon - \frac{1}{N})^2 N}$$

The derivation of this result relies upon a lemma [Parrondo and Van den Broeck, 1993]:

$$Pr\{\sup_{f \in F} |\pi_f - \nu_f| > \epsilon\} < 2Pr\{\sup_{f \in F} |\nu_f - \nu'_f| > \epsilon - \frac{1}{N}\}$$

where ν_f and ν'_f are the error rates on two distinct sets of N examples, each drawn

independently. If $f_1, f_2 \in F$ agree on all $2N$ points, i.e., if they induce the same dichotomy, then $|\nu_{f_1} - \nu'_{f_1}| = |\nu_{f_2} - \nu'_{f_2}|$. Therefore, we need only consider the set D of dichotomies induced on the sample of $2N$ points, and can write

$$Pr\{\sup_{f \in F} |\nu_f - \nu'_f| > \epsilon - \frac{1}{N}\} = Pr\{\max_{\mathbf{d^{(i)}} \in D} |\nu_{\mathbf{d^{(i)}}} - \nu'_{\mathbf{d^{(i)}}}| > \epsilon - \frac{1}{N}\}$$

Let X be the $2N$ by d matrix of input vectors such that $\nu_{\mathbf{d^{(i)}}}$ is computed over rows 1 through N and $\nu'_{\mathbf{d^{(i)}}}$ is computed over rows $N+1$ through $2N$. Define $\rho_i(X)$ as follows:

$$\rho_i(X) = |\nu_{\mathbf{d^{(i)}}} - \nu'_{\mathbf{d^{(i)}}}|$$

Define T to be a permutation matrix which permutes the $2N$ rows of X. It is clear that for any T

$$Pr\{\max_{\mathbf{d^{(i)}} \in D} \rho_i(X) > \epsilon - 1/N\} = Pr\{\max_{\mathbf{d^{(i)}} \in D} \rho_i(TX) > \epsilon - 1/N\}$$

since the $2N$ examples are drawn independently and therefore all orderings are equally likely. Hence we can write

$$Pr\{\max_{\mathbf{d^{(i)}} \in D} \rho_i(X) > \epsilon - 1/N\} = \int_X \frac{1}{2N!} \sum_{j=1}^{2N!} \theta(\max_{\mathbf{d^{(i)}} \in D} \rho_i(T_j X) - (\epsilon - \frac{1}{N}))dP(X)$$

where the index i runs over all $2N!$ possible permutations. Define $\Delta(\mathbf{x^{(1)}}, \ldots, \mathbf{x^{(2N)}})$ as the number of dichotomies induced by F on the $2N$ input vectors. Then the dichotomies induced by F may be represented in a binary matrix Ω with Δ rows and $2N$ columns. $\Omega_{ij} = 0$ if $d_{ij} = y_j$ and 1 otherwise, where d_{ij} is the element j of dichotomy $\mathbf{d^{(i)}}$ and y_j is the class label associated with the input vector $\mathbf{x^{(j)}}$. For each dichotomy $\mathbf{d^{(i)}}, 1 \leq i \leq \Delta$, ν_d is the frequency of 1s among the first N entries in the row, while ν'_d is the frequency of 1s among the second N entries. Permuting the rows of X corresponds to permuting the columns of Ω. The integrand of the integral in the preceding expression is therefore the probability that a random permutation of the columns of Ω will result in a matrix with at least one row $\mathbf{d^{(i)}}$

for which $\rho_i(X) > \epsilon - 1/N$. All permutations are equally likely, so for a given Ω this calculation amounts to counting how many of the $2N!$ permutations result in at least 1 row for which the inequality holds.

partition:	**1**	**2**	**1**	**1**	**2**	**2**	**2**	**1**	**2**	**1**
	$\mathbf{x}^{(1)}$	$\mathbf{x}^{(2)}$	$\mathbf{x}^{(3)}$	$\mathbf{x}^{(4)}$	$\mathbf{x}^{(5)}$	$\mathbf{x}^{(6)}$	$\mathbf{x}^{(7)}$	$\mathbf{x}^{(8)}$	$\mathbf{x}^{(9)}$	$\mathbf{x}^{(10)}$
$\mathbf{d}^{(1)}$	0	1	0	0	1	0	1	1	0	1
$\mathbf{d}^{(2)}$	1	1	0	0	1	0	1	1	0	1
$\mathbf{d}^{(3)}$	1	0	0	0	1	0	1	1	0	1
$\mathbf{d}^{(4)}$	1	0	1	0	1	0	1	1	0	1
$\mathbf{d}^{(5)}$	1	0	1	1	1	0	1	1	0	1
$\mathbf{d}^{(6)}$	1	0	1	1	0	0	1	1	0	1
$\vdots$					$\vdots$					
$\mathbf{d}^{(\infty)}$	1	0	1	1	0	1	0	0	1	0

Figure 5.3: The columns of the dichotomy matrix Ω are randomly partitioned into two groups. For each row, the frequency of 1's is computed for each of the two groups. The VC bound upper bounds the number of column partitions which result in at least one row where the difference in frequencies is greater than $\epsilon - \frac{1}{N}$.

The VC bound is derived first by considering a single row (e.g., row 1) in isolation and bounding the number of column permutations which satisfy the inequality for that row. It suffices to count the number of partitionings of the $2N$ elements of the row into two groups of N, since $\rho_1(X)$ is invariant to permutations which leave the partitioning unchanged. If there are N_1 1s among the $2N$ elements of the row, then $Pr\{\rho_1(X) > \epsilon - 1/N\}$ is given by

$$\Gamma = \sum_k \frac{\binom{N_1}{k}\binom{2N-N_1}{N-k}}{\binom{2N}{N}}$$

where the index k runs over all values satisfying the inequalities

$$|\frac{k}{N} - \frac{N_1 - k}{N}| > \epsilon - \frac{1}{N}$$

The following bound is derived in [Vapnik, 1982]:

$$\Gamma < 3e^{-(\epsilon-\frac{1}{N})^2 N}$$

From the definition of the growth function $G(N)$, we know Δ is bounded above by $G(2N)$. The total number of partitions of the columns of Ω which result in at least one row satisfying $\rho(X) > (\epsilon - \frac{1}{N})$ can be no greater than the maximum number of partitions for which the inequality is satisfied for a single row, multiplied by the number of rows. This union bound argument is used to obtain the final VC result.

$$2Pr\{\max_{d\in D} \rho_d(X) > (\epsilon - \frac{1}{N})\} < 2\Delta\Gamma < 6G(2N)e^{-(\epsilon-\frac{1}{N})^2 N}$$

5.4.2 Derivation of the Connected Function Class Bound

It should be obvious that when the dichotomies are connected, the union bound technique will result in a loose bound. Any partition of the columns for which the inequality is satisfied simultaneously in more than one row is being unnecessarily counted more than once. A connected set of dichotomies is such that many dichotomies (rows) are very similar to each other. At the very least, we know that for every row, there exists at least one other row which differs in only 1 of the $2N$ entries. Clearly, the vast majority of partitions for which the inequality holds in a given row will also result in the inequality being satisfied for the row which differs in only 1 entry. This observation is at the heart of the improved bound being derived here.

Theorem 5.4.1 *For any connected function class, the following bound holds for* $N > \frac{1}{2(\epsilon-\frac{1}{N})^2}$:

$$Pr\{\sup_{f\in F}|\pi_f - \nu_f| > \epsilon\} < 12(\frac{(G(2N)-1)}{\sqrt{\pi N}} + 1)e^{(-(\epsilon-\frac{1}{N})^2 N)}$$

Proof:

Suppose we define some ordering over the dichotomies $\mathbf{d^{(1)}}, \ldots, \mathbf{d^{(\Delta)}}$ induced by

F on a sample of size $2N$. Then we have

$$Pr\{\max_{\mathbf{d^{(i)}}\in\mathbf{D}} \rho_i(X) > \epsilon - \frac{1}{N}\} = \sum_{i=1}^{\Delta} Pr\{\rho_i(X) > \epsilon - \frac{1}{N}, \rho_j(X) \leq \epsilon - \frac{1}{N}\ \forall j < i\ \}$$

Lemma 5.4.1: *For any connected set of dichotomies, there exists an ordering* ($\mathbf{d^{(1)}}, \mathbf{d^{(2)}}, \ldots \mathbf{d^{(\Delta)}}$) *such that the following property holds*:

$$\forall\ i > 1\ \ \exists l(i) < i$$

such that

$$||\mathbf{d^{(i)}} - \mathbf{d^{(l(i))}}|| = 1$$

Proof: The proof is by construction- we describe an iterative procedure for finding an ordering. Let S represent the current ordered set of dichotomies. Pick $\mathbf{d^{(1)}}$ arbitrarily from D and place it in S. Now repeat the following procedure $\Delta - 1$ times: Let i be the current number of dichotomies in S. From the definition of connectedness, there must exist a pair of dichotomies $\mathbf{d^{(a)}} \in S, \mathbf{d^{(b)}} \notin S$ such that $||\mathbf{d^{(a)}} - \mathbf{d^{(b)}}|| = \mathbf{1}$. Set $\mathbf{d^{(i+1)}} = \mathbf{d^{(b)}}$. Then $\mathbf{d^{(l(i))}} = \mathbf{d^{(a)}}$. ■ .

This lemma allows us to bound the quantity

$$Pr\{\rho_i(X) > \epsilon - \frac{1}{N},\ \rho_j(X) <= \epsilon - \frac{1}{N}\ \forall j < i\ \}$$

For $i > 1$, we know $\exists$ a dichotomy $\mathbf{d^{(l(i))}}$ which differs by only 1 bit from $\mathbf{d^{(i)}}$. It is clear that $|\rho_i(X) - \rho_{l(i)}(X)| = \frac{1}{N}$. Let $n_{min} = \lfloor N\epsilon \rfloor$. Then clearly

$$\rho_i(X) > \epsilon - \frac{1}{N},\ \rho_{l(i)}(X) <= \epsilon - \frac{1}{N} \Rightarrow \rho_i(X) = \frac{n_{min}}{N},\ \rho_{l(i)}(X) = \frac{n_{min} - 1}{N}$$

The following bound therefore holds:

$$Pr\{\max_{\mathbf{d^{(i)}}inD} \rho_{\mathbf{i}}(X) > \epsilon - \frac{1}{N}\} \leq Pr\{\rho_{\mathbf{d^{(1)}}}(X) > \epsilon - \frac{1}{N}\} + \sum_{i=2}^{\Delta} Pr\{\rho_i(X) = \frac{n_{min}}{N}\}$$

Define $p(k)$ and $q(k)$ as follows:

$$p(k) = \frac{\binom{N_1}{k}\binom{2N-N_1}{N-k}}{\binom{2N}{N}}$$

$$q(k) = \frac{p(k+1)}{p(k)} = \frac{(N_1-k)(N-k)}{(k+1)(N+k+1-N_1)}$$

$p(k)$ is the probability of k ones appearing in the first of two partitions of the $2N$ elements of a single row. Define k_{min} to be the smallest k such that

$$k - \frac{N_1}{2} > \frac{n_{min}}{2}$$

Then, using the symmetry of $p(k)$, we know that

$$Pr\{\rho_i(X) = \frac{n_{min}}{N}\} = 2p(k_{min})$$

Using Stirling's approximation [Feller, 1950] :

$$\sqrt{2\pi}N^{N+\frac{1}{2}}e^{-N+\frac{1}{12N+1}} < N! < \sqrt{2\pi}N^{N+\frac{1}{2}}e^{-N+\frac{1}{12N}}$$

one can easily derive upper and lower bounds for $\binom{N}{\frac{N}{2}}$:

$$\frac{2^{N+1}}{\sqrt{2\pi N}}e^{\frac{-1}{3N}} < \binom{N}{\frac{N}{2}} < \frac{2^{N+1}}{\sqrt{2\pi N}}e^{\frac{-1}{5N}}$$

Note that the upper bound is also a bound on $\binom{N}{k}$ for all k. This implies that

$$\forall k \;\; p(k) < \frac{2\sqrt{N}}{\sqrt{\pi N_1(2N-N_1)}}$$

We have defined $q(k)$ such that

$$p(k_{min}) = p(\lfloor 0.5n \rfloor) \prod_{i=\lfloor 0.5n \rfloor}^{k_{min}-1} q(i)$$

In [Vapnik, 1982] it is shown that

$$\prod_{i=\lfloor 0.5n \rfloor}^{k_{min}-1} q(i) < \quad exp\Big(\frac{-(N+1)((\epsilon - \frac{1}{N})N)^2}{(N_1+1)(2N-N_1+1)}\Big)$$

Returning to $p(k)$, we have

$$p(k_{min}) < \frac{2\sqrt{N}}{\sqrt{\pi N_1(2N-N_1)}} exp\Big(\frac{-(N+1)((\epsilon - \frac{1}{N})N)^2}{(N_1+1)(2N-N_1+1)}\Big)$$

Differentiating twice with respect to N_1 leads to the conclusion that this expression is maximized at $N_1 = N$ provided that $2(\epsilon - \frac{1}{N})^2 N > 1$, which is required in order for any bound of this nature to evaluate to below unity. Therefore

$$p(k_{min}) < \frac{2}{\sqrt{\pi N}} exp\Big(\frac{-((\epsilon - \frac{1}{N})N)^2}{(N+1)}\Big) < \frac{3}{\sqrt{\pi N}} e^{(-(\epsilon - \frac{1}{N})^2 N)}$$

Thus, we obtain

$$Pr\{\max_{\mathbf{d}^{(i)} \in D} \rho_i(X) > \epsilon - \frac{1}{N}\} = Pr\{\rho_1(X) > \epsilon - \frac{1}{N}\} + \frac{6(G(2N)-1)}{\sqrt{\pi N}} e^{(-(\epsilon - \frac{1}{N})^2 N)}$$

which in turn is bounded by

$$6\Big(\frac{(G(2N)-1)}{\sqrt{\pi N}} + 1\Big) e^{(-(\epsilon - \frac{1}{N})^2 N)}$$

We multiply by the required factor of 2 to complete the proof ■.

The impact of the improvement in the bound will be most substantial for learning models with very low VC dimension, becoming more negligible as the VC dimension v grows. Recall that

$$G(2N) \leq (2Ne/v)^v$$

Hence we can view $\frac{1}{\sqrt{(N)}}$ factor as (roughly speaking) reducing the VC dimension by $\frac{1}{2}$. This reduction is significant when the VC dimension is small but quite minor when the VC dimension is large. In table 5.1, we show the number of examples required to be 90% confident that the true error rate is within 25% of the observed error rate, as a function of the VC dimension. The reduction in the number of examples is more than 10% for a VC dimension of 3, but less than 1 % for a VC dimension of 100.

VCdim	connected class bound	standard bound
3	337	388
10	1032	1095
100	10,090	10,180

Table 5.1: Comparison of standard bounds and connected class bounds.

5.5 Simulations

One might expect that the connectedness theorem should imply a larger improvement in the bound then the one obtained by the result of the previous section. Theorem 5.4.1 only capitalizes on the fact that for each dichotomy there exists another dichotomy which differs by only 1 bit. Connectedness actually implies much more than this- for any dichotomy, there will be many other dichotomies which differ on only a small fraction of the 2N points. Hence, the overcounting done by the standard VC bound is probably even more severe than Theorem 5.4.1 would indicate.

How much more could the bound be tightened? A lower bound is given in [Ehrenfeuch *et al.*, 1993]. The bound applies when we want to have confidence of greater than 99% that uniform convergence at accuracy level $\epsilon \leq \frac{1}{8}$ has occurred. Then the number of examples required is bounded below by

$$N \geq \frac{v-1}{32\epsilon}$$

where v is the VC dimension of the function class. For instance, setting $v = 50$ and

$\epsilon = 0.1$, we arrive at a lower bound of $N = 16$ examples. Upper bounds such as the one given in Theorem 5.4.1 are transcendental equations, meaning that N cannot be expressed analytically as a function of ϵ and v. We can solve numerically for the required N, however. The connected class bound yields $N = 42,300$! It is therefore hardly an exaggeration to say that a considerable gap exists between upper and lower bounds.

Interestingly, theoretical lower bounds are not the only way to assess the potential for tightening the upper bound. This section will show how Monte Carlo simulations can be used to gauge how much room for improvement there remains.

Recall that the critical intermediate step in the derivation of Theorem 5.4.1 is the bounding of the quantity

$$p = Pr\{\max_{\mathbf{d}^{(i)} \in \mathbf{D}} |\nu_{\mathbf{d}^{(i)}} - \nu'_{\mathbf{d}^{(i)}}| > \epsilon - \frac{1}{N}\}$$

p is bounded by bounding the number of partitions of the columns of the dichotomy matrix Ω which result in at least one row j for which

$$|\nu_{\mathbf{d}^{(j)}} - \nu'_{\mathbf{d}^{(j)}}| > \epsilon - \frac{1}{N}\}$$

If Ω were known explicitly, then p could be estimated by randomly partitioning the columns of Ω many times and observing how frequently the statement

$$\{\max_{\mathbf{d}^{(i)} \in D} |\nu_{\mathbf{d}^{(i)}} - \nu'_{\mathbf{d}^{(i)}}| > \epsilon - \frac{1}{N}\}$$

held true. For certain simple models, however, we do know Ω. Perhaps the simplest model which still contains an infinite number of hypotheses is the one-dimensional threshold model

$$f(x) = \theta(x - b)$$

It is easy to see that the threshold model induces $N + 1$ dichotomies- the threshold b can be positioned to be larger than k of the data points, for any $0 \leq k \leq N$. Suppose

we form an $N+1$ by N matrix A with row k corresponding to the dichotomy which classifies $k-1$ inputs as 0. The first N rows of A correspond to a square matrix with 1's on and above the diagonal and $0's$ below, while the last row is all 0's. Then Ω can be obtained immediately by comparing each row of A to the target dichotomy and setting $\Omega_{ij} = 1$ if there is agreement between A_{ij} and the corresponding entry in the target dichotomy.

This procedure was implemented to estimate p for various N and ϵ for the threshold model. Table 5.2 shows number of examples required for the estimate of p to fall below 0.5, as a function of ϵ. The number of examples predicted by the standard VC bound and the connected function class bound are shown as well.

ϵ	Monte Carlo	connected class bound	standard bound
0.1	290	780	1140
0.05	1100	3380	5120
0.02	6800	23500	36800

Table 5.2: Comparison of simulation results and theory for threshold model

The simulations demonstrate that there is still considerable room for improvement. It should be emphasized, however, that these simulation results apply only for the threshold model. The gap between theory and simulation could be either larger or smaller for more complex models. We also need to note that the simulations only bound the room for improvement via techniques which bound

$$p = Pr\{\max_{\mathbf{d^{(i)}} \in \mathbf{D}} |\nu_{\mathbf{d^{(i)}}} - \nu'_{\mathbf{d^{(i)}}}| > \epsilon - \frac{1}{N}\}$$

as an intermediate step. It is conceivable that better bounds could be obtained through another approach.

5.6 Conclusion

The significance of the results presented in this chapter lies less in the improvement in the bounds than in having introduced the notion that the diversity of the dichotomies of a function class plays a role in its generalization behavior. Opportunities abound for further investigations of this notion. One could develop more detailed characterizations of dichotomy graphs. For instance, the average number of edges incident to a node in the graph could be calculated for various function classes. A dichotomy graph with many edges per node would correspond to a less diverse dichotomy set than a graph with only a few edges per node. A second line of investigation could consider alternative measures of the diversity of a set of a dichotomies. One promising concept is the number of dichotomies a function class can *approximate.* We say that two dichotomies δ-approximate each other if the fraction of input vectors they disagree on is less than δ. Then a function class δ-approximates a dichotomy $\mathbf{d}$ if it implements a dichotomy $\mathbf{d}'$ which δ-approximates $\mathbf{d}$. A function class which implements Δ clustered dichotomies will approximate fewer dichotomies than a second class which implements Δ dispersed dichotomies. Perhaps bounds can be obtained in terms of the number of dichotomies a function class can approximate.

Chapter 6 Conclusion

A general conclusion to a thesis like this one, which consists of two loosely related parts, can often seem forced. It does seem, though, that an overarching lesson about learning systems can be derived from the work presented here: the importance of capitalizing on all available information. Each chapter of the thesis can be interpreted as an illustration of this point.

The lesson is most easily derived from the chapters on monotonicity constraints. This research demonstrated the value of using prior knowledge about the target function. Chapters 2 and 3 showed that models which obey monotonicity constraints can outperform other models that fail to adhere to this property when it is known to hold for the target. Chapter 4 supported this idea theoretically with results about the relatively low data requirements of monotonic models.

The connection between the research in Chapter 5 and the value of information may be less obvious, but no less important. Here we saw the need to take advantage of all available information about the *models* we use. Previous learning theory characterized a function class only through the number of dichotomies it induces. We showed that there was more information available about many common function classes, namely, the connectedness property. This information turned out to be relevant as well, enabling us to obtain an improved bound. Future work on learning theory should bear this lesson in mind. Progress is most likely to be made by incorporating more detailed descriptions of learning systems into the theory.

Bibliography

[Abu-Mostafa, 1990] Y. Abu-Mostafa. Learning from hints in neural networks. *Journal of Complexity*, 6:192–198, 1990.

[Abu-Mostafa, 1993] Y. Abu-Mostafa. Hints and the vc dimension. *Neural Computation*, 4:278–288, 1993.

[Abu-Mostafa, 1995] Y. Abu-Mostafa. Financial market applications of learning from hints. In A. Refenes, editor, *Neural Networks in the Capital Markets*, pages 278–288. Wiley, 1995.

[Archer and Wang, 1993] N.P. Archer and S. Wang. Application of the back propagation neural network algorithm with monotonicity conditions for two group classification problems. *Decision Sciences*, 24:60–75, 1993.

[Armstrong *et al.*, 1996] W. W. Armstrong, C. Chu, and M. M. Thomas. Feasibility of using adaptive logic networks to predict compressor unit failure. In P. Keller, S. Hashem, L. Kangas, and R. Kouzes, editors, *Applications of Neural Networks in Environment, Energy, and Health*, page Chapter 12. World Scientific Publishing Company, 1996.

[Baum and Haussler, 1989] E. B. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1:151–160, 1989.

[Bishop, 1995] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[Cover, 1965] T.M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14:326–334, 1965.

[Ehrenfeuch *et al.*, 1993] A. Ehrenfeuch, D. Haussler, M. Kearns, and L. Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82:247–267, 1993.

[Feller, 1950] W. Feller. *An Introduction to Probability Theory and Its Applications.* John Wiley & Sons, New York, 1950.

[Ford and Fulkerson, 1962] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1962.

[Geman *et al.*, 1992] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias-variance dilemma. *Neural Computation*, 4:1–58, 1992.

[Hertz *et al.*, 1991] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, 1991.

[Hoeffding, 1956] W. Hoeffding. *Ann. Math. Statist*, 27:713–721, 1956.

[McHugh, 1990] J. A. McHugh. *Algorithmic Graph Theory.* Prentice Hall, Englewood Cliffs, NJ, 1990.

[Moody, 1992] J. E. Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, volume 4, pages 847–854. Morgan Kaufmann, 1992.

[Mukarjee and Stern, 1994] H. Mukarjee and S. Stern. Feasible nonparametric estimation of multiargument monotone functions. *Journal of the American Statistical Association*, 89:77–80, 1994.

[Parrondo and Van den Broeck, 1993] J.M.R. Parrondo and C. Van den Broeck. Vapnik-chervonenkis bounds for generalization. *Journal of Physics A: Mathematical and General*, 26:2211–2233, 1993.

[Simard *et al.*, 1993] P. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, volume 4, pages 50–58. Morgan Kaufmann, 1993.

[Stanley, 1986] R. P. Stanley. *Enumerative Combinatorics.* Wadswroth & Brooks/Cole, Monterey, CA, 1986.

[Vapnik and Chervonenkis, 1971] V. Vapnik and A. Y. 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. Vapnik. *Estimation of Dependencies Based on Empirical Data.* Springer-Verlag, 1982.

[Vapnik, 1995] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

[Wolpert, 1996] D. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8:1341–1390, 1996.