LEARNING INTERNAL REPRESENTATIONS

Jonathan Baxter BSc.(Hons)

December 13, 1994

A thesis to be submitted in fulfillment of the requirements of a degree of Doctor of Philosophy at The Flinders University of South Australia.

Author's address: Discipline of Mathematics, Faculty of Science, The Flinders University of South Australia
E-mail address: jon@maths.flinders.edu.au

Contents

Abstract **3**

Chapter 1. Introduction **5**
1.1. Overview of the thesis. 10
1.2. Glossary 11

Chapter 2. Ordinary Learning **15**
2.1. The General Problem 15
2.2. Deviation of Empirical and True Loss 17
2.3. Bounding the Deviation 18
2.4. EC and its incompatibility with PC 19

Chapter 3. Representation Learning **23**
3.1. General Framework 23
3.2. Motivation 25
3.3. The Advantage of Learning Many Tasks 26
3.4. Philosophical Aside 27
3.5. Bounds for PC Representation Learning 28
3.5.1. Computing the Empirical Loss 29
3.5.2. Preliminary Definitions 29
3.5.3. Generalisation on $\boldsymbol{n}$ tasks 30
3.5.4. Generalisation on all tasks. 33
The First Inequality. 34
The Second Inequality. 35
Main Bound. 38
3.5.5. Discussion 39
3.6. General Framework 40
3.6.1. Preliminaries. 40
3.6.2. Generalisation on n tasks. 41
3.6.3. Learning the hypothesis space. 43

Chapter 4. Algorithms and Experiments **45**
4.1. Learning Binary Neural Network Representations 46
4.1.1. The Experiment 47
4.2. Learning Continuous Neural Network Representations 51
4.2.1. Gradient Descent for Representation Learning 51
4.2.2. Representation Learning Using Backpropagation 54
4.2.3. Experiment One: Learning Translation Invariance 55
Representation vs. No Representation. 60
4.2.4. Experiment Two: Learning Symmetric Functions 62
4.2.5. Local Minima. 65
4.3. Zip-Code Network 67

Chapter 5. Vector Quantization **73**
5.1. The Problem 73
5.2. The Solution 74
5.3. Optimal Function Approximation 76
5.4. Learning ρ 79
5.5. Discussion 80

Chapter 6. Learning a Representation Directly **81**
6.1. An error measure for the representation. 81
6.2. Two Example Calculations of $\rho_{\mathcal{G}}$. 82
6.3. Representation Learning for Classification Problems 83
6.4. Experiment 84
6.5. Discussion and Future Directions 86

Appendix A. Measurability and Permissibility **87**

Appendix B. The Fundamental Theroem **91**
B.1. Miscellaneous definitions 91
B.2. Psuedo-metric spaces and ε-covers. 92
B.2.1. Capacity. 93
B.3. The Fundamental Theorem 93

Appendix C. Bounding $\mathcal{C}(\varepsilon, \mathcal{H})$ **99**
C.1. Pseudo-Dimension 99
C.2. Loss Functions 101
C.3. Capacity of Compositions 101
C.4. Capacity of Products 103
C.5. Diagonal Function Spaces 105

Appendix D. Neural Network bounds. **107**

Bibliography **111**

Abstract

Most machine learning theory and practice is concerned with learning a single task. In this thesis it is argued that in general there is insufficient information in a single task for a learner to generalise well and that what is required for good generalisation is information about *many similar learning tasks*. The information about similar learning tasks forms a body of prior information that can be used to constrain the learner and make it generalise better. Examples of learning scenarios in which there are many similar tasks are handwritten character recognition (if one includes the Kanji characters) and spoken word recognition.

After proving that learning without prior information is impossible except in the simplest of situations, the concept of the *environment* of a learner is introduced as a probability measure over the set of learning problems the learner might be expected to learn. It is shown how a sample from the environment may be used to learn a *representation*, or recoding of the input space that is appropriate for the environment. Learning a representation can equivalently be thought of as learning the appropriate features of the environment. Using Haussler's statistical decsision theory framework for machine learning, rigorous bounds are derived on the sample size required to ensure good generalisation from a representation learning process. These bounds show that under certain circumstances learning a representation appropriate for n tasks reduces the number of examples required of each task by a factor of n. It is argued that environments such as character recognition and speech recognition fall into the category of learning problems for which such a reduction is possible.

Once a representation is learnt it can be used to learn *novel* tasks from the same environment, with the result that far fewer examples are required of the new tasks to ensure good generalisation. Rigorous bounds are given on the number of tasks and the number of samples from each task required to ensure that a representation will be a good one for learning novel tasks.

All the results on representation learning are generalised to cover any form of automated hypothesis space bias that utilises information from similar learning problems.

It is shown how gradient-descent based procedures for training Artificial Neural Networks can be generalised to cover representation learning. Two experiments using the new procedure are performed. Both experiments fully support the theoretical results.

The concept of the environment of a learning process is applied to the problem of *vector quantization* with the result that a *canonical* distortion measure for the quantization process emerges. This distortion measure is proved to be optimal if the task is to approximate the functions in the environment.

Finally, the results on vector quantization are reapplied to representation learning to yield an improved error measure for learning in classifier environments. An experiment is presented demonstrating the improvement.

CHAPTER 1

Introduction

Artificial Neural Network (ANN) models have become popular in machine learning primarily because they offer the possibility of *nonparametric* or *model-free* inference. It is thought that a neural network with a sufficiently general architecture should be able to learn any training set whatsoever, thus doing away with the laborious, difficult and possibly unreliable process of model selection. Unfortunately, although it is true that neural networks are *universal* in the sense that they can approximate to arbitrarily high accuracy any continuous function, to do so requires an inordinately large number of parameters, except in the very simplest of cases[1] (of which the *xor* problem is probably the most famous). Ensuring low variance when simultaneously estimating a large number of parameters requires prohibitively large training sets, which typically are not available in practice. Thus the only option available to the neural network researcher in most learning problems is to use their prior knowledge of the learning problem to try and reduce the effective number of parameters. This is basically the same as introducing a model for the problem, thus negating the original aim of model-free inference. This problem is now well recognised in the neural network literature and goes by the name of *the bias/variance dilemma*. A more extensive discussion is presented in Geman et. al. (1992).

In machine learning in general[2], not only in neural network research, it is not the learning problem itself that poses the most difficulty, but the task of identification of appropriate prior information, and how the information should be used to bias the model space. Once a manageable sized model has been found, the task of parameter estimation is relatively trivial. Almost all successes in the application of neural networks to real-world problems can be attributed to an appropriate choice of network architecture and data representation so that the effective number of parameters is small (in this context see Geman et. al. again).

The main conclusion to be drawn from the general failure of neural network techniques to perform truly model-free inference in practice is not that neural networks are a bad model for learning (although that may be the case for other reasons), but

[1] Even for neural networks with only one node the parameter count can be astronomical. For example, a one-node neural network performing classification on 100×100 images will have $10,000$ parameters!

[2] Classical and modern methods of statistical inference should be included as branches of machine learning also, although a statistician would probably prefer to view machine learning as a branch of statistical inference.

that there is simply *insufficient information* in most training sets to specify a solution, no matter what learning technique is used. On face value this appears to be a quite depressing conclusion, for it implies that in practice machine learning is a pretty hopeless task. However, the situation is not quite that bleak, for even though there is, in general, insufficient information in any individual training set to specify a solution to the problem, it appears that in many cases the training set together with the *prior knowledge* of the researcher do contain enough information to solve the problem. If the information source from which researchers derive their prior knowledge could be located, then perhaps this source could be connected directly to an appropriate machine learning algorithm, which would then be able to crunch out its own model biases. This thesis demonstrates one way of doing this.

There are many methods used to bias a large class of models down to a more managable size. Approaches used in neural networks include restrictions on network size, limitations on the number of different weight values, limitations on the number of non-zero weights, and perhaps the most successful method of all—reduction in input dimension through appropriate recoding of the data. This last technique is commonly called "finding a good representation for the data". The work in this thesis is primarily concerned with the problem of finding a good representation, although generalisations of the theory are given at the end of chapter 3 that provide a basis for extension to other kinds of model bias.

There are essentially two sources of prior information about learning problems. The first and most commonly known is knowledge of the physical or mathematical processes underlying the problem. In many cases such knowledge is sufficient to generate a physical model of the problem with few enough parameters that can be estimated using realistically sized training sets. However, for many problems there is very little information available about the physical processes underlying the generation of the data. For example, consider image recognition or character recognition. The data is essentially being generated by our brains—a very complicated physical device about which we have very little information. Thus in these problems another form of prior information is used: *knowledge of similar kinds of learning problems.*

To see how knowledge of similar learning problems provides information constraining the choice of a good representation, consider the problem of learning to recognise the handwritten digit '1'. The most extreme representation possible would be one that completely solves the classification problem, i.e. a representation that outputs 'yes' if its input is an image of a '1', regardless of the position, orientation, noise or writer dependence of the original digit, and 'no' if any other image is presented to it. Learning to recognise the digit '1' using such a representation would require only one example of the digit, for after that the learner would know the image was a '1' if the representation said 'yes' to the image.

Although the representation in this example certainly reduces the complexity of the learning problem, it does not really seem to capture what is meant by the term *representation.* What is wrong is that although the representation is an excellent one for learning to recognise the digit '1', *it could not be used for any other learning task.* A representation that is appropriate for learning to recognise '1' should also be appropriate for other character recognition problems—it should be good for learning other digits, or the letters of the alphabet, or Kanji characters, or Arabic letters, and so on. Thus the information necessary to determine a good representation is not contained in a single learning problem (recognising '1'), but is contained in many examples of similar learning problems. The same argument can be applied to other familiar learning

problems, such as face recognition and speech recognition. A representation appropriate for learning a single face should be appropriate for learning all faces, and similarly a single word representation should be good for all words (maybe even regardless of the language). Thus the information for determining a good representation is not contained in a single learning problem, but in many learning problems of a similar nature. This conclusion is not restricted to model bias through the use of a representation, but would apply to any mechanism of model bias for which information is available regarding similar learning problems.

How can prior information about similar learning problems be used *automatically* to derive a good representation for the learning domain? To illustrate the solution to this problem, suppose the learning problems are all character recognition problems or similar (recognise '1', recognise '2', recognise '$', and so on). Denoting the space of images by X, the character recognition problems can all formally be represented by Boolean functions on X. For example, the function $h_{\text{`1'}}: X \to \{0,1\}$ such that $h_{\text{`1'}}(x) = 1$ if and only if x is an image of the character '1', represents the learning problem "recognise '1'". The set of *similar learning problems* in this case is formally equivalent to a set of Boolean functions $\mathcal{H}: X \to \{0,1\}$.

In an ordinary machine learning situation, for example learning to recognise the character '1', the learner would receive a *training set* $\vec{z} = \{(x_1, h_{\text{`1'}}(x_1)), \ldots, (x_m, h_{\text{`1'}}(x_m))\}$, consisting of sample images $x_i \in X$, and their correct classifications, $h_{\text{`1'}}(x_i)$. The images $x_i$ would be chosen from $X$ according to some fixed (but unknown to the learner) probability distribution $P$ on $X$. The training set is the only information provided to the learner. In order to learn the function $h_{\text{`1'}}$, the learner would have available a *hypothesis space*—a set of functions $\mathcal{H}'$ from which it chooses the best approximation to $h_{\text{`1'}}$ that it can find. If the learner is trying to learn $h_{\text{`1'}}$ using neural networks, $\mathcal{H}'$ would consist of all functions that can be implemented by neural networks (with perhaps a restriction on the networks' architectures). In this case the range of the functions $h' \in \mathcal{H}'$ would normally be the interval $[0,1]$, rather than the set $\{0,1\}$, because continuous neural networks are in general far easier to train than discontinuous ones (gradient descent cannot be used on discontinuous networks). The agreement between a network $h' \in \mathcal{H}'$, and the function $h_{\text{`1'}}$ is typically measured using the *mean-squared* error measure,

$$E(h', \vec{z}) = \frac{1}{m} \sum_{i=1}^{m} \left[h'(x_i) - h_{\text{`1'}}(x_i)\right]^2 . \tag{1}$$

In this thesis $E(h', \vec{z})$ is called the *empirical error of h' on sample $\vec{z}$*. $E(h', \vec{z})$ is usually minimized by first selecting a random member h' from $\mathcal{H}'$ and then performing some form of *gradient descent* on the parameters of h'. Once a satisfactory solution h' has been found, it is then used to classify novel images x by calculating $h'(x)$ and declaring the image to be a '1' if $h'(x) > 0.5$ and not a '1' otherwise. The true utility of h' is measured not by the empirical error, $E(h', \vec{z})$, which after all can be made zero by a procedure that simply rote-learns the entire training set $\vec{z}$, but by the *true error*,

$$E(h', h_{\text{`1'}}) = \int_X \left[h'(x) - h_{\text{`1'}}(x)\right]^2 \, dP(x). \tag{2}$$

The *true error* measures how well the function h' will perform *in practice*, i.e. how it will perform on novel examples not included in its training set. The true error

is more commonly referred to in the neural network literature as the *generalisation error*. The hope is that if the empirical error is small, and the training set is large enough, then with high probability the true error will also be small. However this is clearly dependent on the nature of the learner's hyopthesis space $\mathcal{H}'$, for in the extreme $\mathcal{H}'$ may contain functions that can rote-learn any sample $\vec{z}$, no matter how large. In that case $\vec{z}$ could never be made large enough to ensure with high probability a close agreement between the empirical and true errors. In practice $\mathcal{H}'$ is never that pathological, but if the learner truly does have no knowledge of the problems it is likely to encounter, then for it to be sure that it will have a good $h' \in \mathcal{H}'$ for any problem, it has no choice but to make $\mathcal{H}'$ very large indeed, with very many parameters to be estimated, and hence will require an extraordinarily large training set to ensure a close agreement between the empirical and true errors. Thus, for learning to be feasible the hypothesis space $\mathcal{H}'$ must be biased or reduced in some way.

Recall that the actual learning problem in this case (recognise '1') is but one of many similar learning problems $h \in \mathcal{H}$. To bias the learner's hypothesis space $\mathcal{H}'$ based on prior information about $\mathcal{H}$, suppose that the occurence of similar kinds of learning problems is governed by a probability measure Q on $\mathcal{H}$. Call Q the *environment* of the learner. Any training set $\vec{z} = \{(x_1, h(x_1)), \ldots, (x_m, h(x_m))\}$ supplied to the learner is now assumed to have been generated by first sampling from $\mathcal{H}$ according to Q to generate h, and then sampling m times from X according to P to generate $x_1, \ldots, x_m$ (and hence $\vec{z} = \{(x_1, h(x_1)), \ldots, (x_m, h(x_m))\}$). To provide prior information about similar kinds of learning problems, $\mathcal{H}$ is sampled n times according to Q to generate $h_1, \ldots, h_n$, and then for each i, $1 \leq i \leq n$, X is sampled m times according to P to generate $x_{i1}, \ldots, x_{im}$. The full training set is then an $n \times m$ (n rows, m columns) *matrix*,

$$\mathbf{z} = \begin{matrix} (x_{11}, h_1(x_{11})) & \ldots & (x_{1m}, h_1(x_{1m})) \\ \vdots & \ddots & \vdots \\ (x_{n1}, h_n(x_{n1})) & \ldots & (x_{nm}, h_n(x_{nm})). \end{matrix}$$

This training set is essentially a list of n ordinary training sets, each one consisting of examples of a character to be recognised (h_1 could be '1', h_2 'A', h_3 '%' and so on). The learner uses the information contained in $\mathbf{z}$ to learn a representation appropriate for the environment Q by first splitting its hypothesis space in two: $\mathcal{H}' = \mathcal{G} \circ \mathcal{F}$[3] where $\mathcal{F}$ is a set of functions mapping X into some space V, and $\mathcal{G}$ is a set of functions mapping V into $[0, 1]$. This splitting will be denoted by

$$X \xrightarrow{\mathcal{F}} V \xrightarrow{\mathcal{G}} [0, 1].$$

If $\mathcal{H}'$ consists of neural networks, one way for it to be split is to let $\mathcal{F}$ include all the hidden layers of $\mathcal{H}'$ and let $\mathcal{G}$ consist simply of the last layer of $\mathcal{H}'$. In this case V would be the space $[0, 1]^p$ where p is the number of nodes in the last hidden layer of the networks in $\mathcal{H}'$ (assuming the hidden nodes are $[0, 1]$-valued functions). Clearly $\mathcal{H}'$ could in fact be split at any any layer. The representation is to be chosen from the space $\mathcal{F}$, and hence $\mathcal{F}$ will be called the *representation space*. Each individual $f \in \mathcal{F}$ is a candidate *representation* for the environment Q. The learner wishes to find a representation $f \in \mathcal{F}$ such that a learning problem h selected from $\mathcal{H}$ according to Q, will, with high probability, be well approximated by some function

[3] $\mathcal{G} \circ \mathcal{F} = \{g \circ f : g \in \mathcal{G}, f \in \mathcal{F}\}$.

$g \in \mathcal{G}$ composed with f. In other words, f should be such that if the learner learns within the environment Q, using the hypothesis space $\mathcal{G} \circ f$[4] rather than the full space $\mathcal{H}' = \mathcal{G} \circ \mathcal{F}$, then with high probability it will be successful. Note that learning with $\mathcal{G} \circ f$ will require far fewer examples in general than learning using the full space $\mathcal{G} \circ \mathcal{F}$, because only the parameters of $\mathcal{G}$ need to be estimated, rather than *both* the parameters of $\mathcal{G}$ *and* the parameters of $\mathcal{F}$.

To measure the effectiveness of a representation f on a sample $\mathbf{z}$ as above, define the *empirical error* $E(f, \mathbf{z})$ by

$$E(f, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} \inf_{g \in \mathcal{G}} \frac{1}{m} \sum_{j=1}^{n} \left[g \circ f(x_{ij}) - h_i(x_{ij}) \right]^2 .$$

If each row of $\mathbf{z}$ is denoted by $\vec{z}_i$, so that $\vec{z}_i = \{(x_{i1}, h_i(x_{i1})), \ldots, (x_{im}, h_i(x_{im}))\}$, then $E(f, \mathbf{z})$ can be rewritten as

$$E(f, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} \inf_{g \in \mathcal{G}} E(g \circ f, \vec{z}_i), \tag{3}$$

where $E(g \circ f, \vec{z}_i)$ is the ordinary empirical error of the function $g \circ f$ on sample $\vec{z}_i$, as defined in (1). The empirical error of a representation f is a measure of how well the learner can learn using f, assuming that the learner is able to find the best possible $g \in \mathcal{G}$ for any given sample $\vec{z}$. For example, if the empirical error of f on $\mathbf{z}$ is zero, it is possible for the learner to find a function g_i for each i, $1 \leq i \leq n$, such that the function $g_i \circ f$ agrees exactly with the function h_i on the sample $\vec{z}_i$.

There are two different natural ways of measuring the *true error* (as distinct from the empirical error) of a representation f. The first is a measure of the true performance of f with respect to the n functions $(h_1, \ldots, h_n)$ used to derive the training set $\mathbf{z}$:

$$E(f, h_1, \ldots, h_n) = \frac{1}{n} \sum_{i=1}^{n} \inf_{g \in \mathcal{G}} E(g \circ f, h_i), \tag{4}$$

where $E(g \circ f, h_i)$ is the true error of $g \circ f$ with respect to h_i, as defined in (2). In chapter 3 it is shown that for problem domains in which the number of parameters in $\mathcal{F}$ greatly exceeds the number of parameters in $\mathcal{G}$, the number of examples m required of each function in the sample $\mathbf{z}$ to ensure that (3) and (4) are close is roughly a fraction $\frac{1}{n}$ times the number of examples required to ensure that (1) and (2) are close. This gives the same number of examples in total as the ordinary learning situation ($\frac{1}{n}$ times as many per function, but there are n functions in all) but they are spread across many functions. This is a genuine advantage if the learner can easily obtain data sets corresponding to many similar learning problems, as is the case for character recognition (if sampling includes Kanji characters, the number of different learning tasks available is in the thousands), face recognition (there are about five billion faces in the world at the moment), speech recognition (the number of different words in all the different languages must number in the millions), and so on. Note also that even though the total sample size required for good generalisation is not reduced by sampling from many similar tasks, it is still a good idea to sample in this way because the more tasks that are sampled from, the more likely it is that the resulting representation f will be a good one for the environment concerned, and so

[4] $\mathcal{G} \circ f = \{g \circ f : g \in \mathcal{G}\}$.

will be useful for learning future tasks drawn from the same environment. In fact the second natural definition of the true error of f is a measure of how good f will be for learning tasks from $\mathcal{H}$ in general:

$$E(f,Q) = \int_{\mathcal{H}} \inf_{g \in \mathcal{G}} E(g \circ f, h)\, dQ(h). \tag{5}$$

A representation with a small value for $E(f,Q)$ will, with high probability, be suitable for learning any function $h \in \mathcal{H}$ drawn randomly according to Q, assuming that the learner is able to find the best possible $g \in \mathcal{G}$ in every case.

Clearly, to ensure small deviation between (3) and (5), both n—the number of tasks being learnt, and m—the number of times each task is sampled, must be sufficiently large. Roughly speaking, it turns out that n must exceed the number of examples that would be required for ordinary learning of a single task, if the full hypothesis space $\mathcal{G} \circ \mathcal{F}$ is used. This is a rather depressingly large bound, but it should be noted that it arises from a worst-case analysis, and that in practice it is likely that far fewer learning tasks will be required. This is certainly borne out by the experiments presented in chapter 4.

Although the theoretical and experimental results presented in this thesis show that representation learning goes some of the way towards solving the problem of automatic model bias, it is *not* the claim of this thesis that the entire problem has been solved. To ensure feasible representation learning in practice, the representation space $\mathcal{F}$ will still have to be subject to some kind of preliminary bias. However the bias will be far less than that required in an ordinary learning scenario, and could perhaps be implemented simply through limited knowledge of natural neural structures known to be effective for the problem, or through considerations of computational complexity issues. This is a subject deserving further investigation.

Although this thesis is principally devoted to theoretical and experimental investigations concerning the issue of representation learning and model-bias in general, it turns out that the framework developed also provides a solution to a very different problem: how to choose an appropriate distortion measure for vector quantization. This is discussed in chapter 5.

1.1. Overview of the thesis.

Most of the thesis is devoted to representation learning. All theoretical results relating to representation learning are derived within the *statistical decision theory* framework of machine learning introduced by Haussler (1992), as this is the most refined and generally applicable machine learning framework so far developed.

Chapter 2 begins with a discussion of the main theoretical problems associated with machine learning, namely, how can one be confident that a given model will accurately approximate the data, and how can one be sure of good generalisation. Formal definitions of these concepts are given, under the titles of "EC" and "PC" learning. A theorem of Haussler (1992) bounding the number of examples required for PC learning is then discussed. PC and EC learning together are then shown to be equivalent to a variation of "PAC" learning as introduced by Valiant (1984). It is shown that in the absence of any prior information about the problem domain, PAC learning is impossible, at least if the problem is to learn continous functions on R.

Chapter 3 is devoted to a discussion of a more general nature than the one in this introduction, of how prior information may be introduced through representa-

tion learning, and a subsequent derivation of the number of examples (and learning tasks) required to ensure, at least in the worst case, good generalisation performance from a representation. In the final part of the chapter a mathematical framework is introduced that is applicable to any kind of machine learning procedure that uses knowledge of similar learning problems to bias its hypothesis space (or models). Generalisations of the sample size bounds for representation learning are derived.

In chapter 4 an algorithm is presented for training neural networks to learn representations. It is based on gradient-descent procedures for ordinary learning problems. Three separate experiments are given, two of which use the gradient procedure and one which uses exhaustive search. In all experiments the theoretical results of chapter 3 are well supported. In the first gradient-based experiment, a representation appropriate for learning a simple kind of translationally-invariant Boolean function is learnt, and in the second a representation for learning symmetric Boolean functions is learnt.

The next chapter, chapter 5, shows how the idea of an *environment* of a learning process can be applied to the problem of vector quantization, with the result that a natural definition of the distortion measure between two signals emerges. This distortion measure is shown to be optimal if the task is to approximate functions from the environment. A very brief discussion is given showing how the distortion measure may be estimated using representation learning techniques.

The ideas of the quantization chapter are then reapplied to representation learning in the final chapter, chapter 6, and as a result a superior measure of the error of a representation is derived. An experiment is presented showing the effectiveness of this improved error measure.

Most of the more technical mathematical results have been relegated to the appendices. This is not because they are trivial or irrelevant, but because to have included them in the main text would have upset the flow of the discussion too much. The appendices have been written so as to be self-contained and hence can be read through in their own right, although for much of the motivation the reader may need to refer to the main text. Appendix A contains definitions and lemmas relating to measurability criterion needed elsewhere in the work. Appendix B is devoted to a proof of the *fundamental theorem* which is crucial in bounding the number of examples needed to ensure good generalisation from a representation learner. Appendix C contains definitions and results that allow the *capacity* of a complex class of functions to be bounded in terms of its simpler constituents. Finally, appendix D applies the results of appendix C to the problem of bounding the capacity of *feedforward neural networks*.

1.2. Glossary

Most of the notation used in this thesis is summarised below, along with a brief description and the page on which it is defined.

X: Input Space 15
Y: Outcome or Output Space 15
Z: Sample Space ($= X \times Y$) 15
P: Probability measure on Z 15
l: Loss function 15
A: Action space 15
$\mathcal{H}$: Hypothesis space 15

h: Hypothesis 15
l_h: Loss function for hypothesis h 15
$\langle l_h \rangle_P$: Expected value of l_h 16
$l_{\mathcal{H}}$: Set of loss functions 16
$\langle l_h \rangle_{\vec{z}}$: Empirical estimate of l_h 16
$\mathcal{A}$: Learner 16
d_ν: Metric on R^+ 17
PC: Probably Consistent 17
d_P: Pseudo-metric on functions 18
$(\mathcal{F}, d_P)$: Pseudo-metric space 19
$\mathcal{N}(\varepsilon, \mathcal{F}, d_P)$: Smallest ε-cover 19
$\mathcal{M}(\varepsilon, \mathcal{F}, d_P)$: Packing number 19
$\sigma_{\mathcal{F}}$: σ-algebra on Z induced by $\mathcal{F}$ 19
$\mathcal{P}_{\mathcal{F}}$: Probability measures on Z induced by $\mathcal{F}$ 19
$\mathcal{C}(\varepsilon, \mathcal{F})$: ε-capacity of $\mathcal{F}$ 19
$r^*(P)$: Best possible loss 19
(α, ν)-**EC**: Empirically Correct 20
(α, ν)-**PAC**: Probably Approximately Correct 20
Q: Environmental Measure 23
$\mathbf{z}$: (n, m)-sample 24
$Z^{(n,m)}$: $n \times m$ matrices over Z 24
(n, m)-**sample**: (n, m)-sample 24
$E^*_{\mathcal{G}}(f, \mathbf{z})$: Empirical representation loss 24
$E^*_{\mathcal{G}}(f, Q)$: True representation loss 25
$E(\vec{h}, \mathbf{z})$: Empirical loss of sequence of hypotheses $\vec{h}$ 29
$E(\vec{h}, \vec{P})$: True loss of sequence of hypotheses $\vec{h}$ 29
$\mathcal{H}^n$: Set of sequences of hypothesis 29
$\overline{\mathcal{H}}$: Diagonal hypothesis space 29
$\mathcal{G}^n \circ \overline{\mathcal{F}}$: Space for representation learning 29
$\vec{g} \circ \bar{f}$: Element of $\mathcal{G}^n \circ \overline{\mathcal{F}}$ 29
$E^*_{\mathcal{G}}(f, \vec{P})$: True loss of representation (2) 29
$\langle f \rangle_{\vec{P}}$: Expected value of $f : Z^n \to [0, M]$ 31
$\langle f \rangle_{\mathbf{z}}$: Empirical estimate of $f : Z^n \to [0, M]$ 31
$\bar{l}_{g \circ f}$: Induced function on probability measures 36
$\bar{l}_{\mathcal{G} \circ \mathcal{F}}$: Space of induced functions on probability measures 36
l^*_f: Function on probability measures 36
$l^*_{\mathcal{F}}$: Space of functions on probability measures 36
d_Q: Metric on functions on probability measures 36
H: Hypothesis space family 40
$E^*(\mathcal{H}, \mathbf{z})$: Empirical loss of hypothesis space 40
$E^*(\mathcal{H}, Q)$: True loss of hypothesis space 40
$E_d(\vec{x})$: Reconstruction error 73
$\rho(x, y)$: Canonical distortion measure 74
$E_{\mathcal{F}}(\vec{x}, \vec{X})$: Reconstruction error of $\mathcal{F}$ 77
$\rho_{\mathcal{G}}(v, w)$: Distortion measure 81
$\sigma_{\mathcal{H}}$: Sigma algebra on Z induced by $\mathcal{H}$ 87
$\mathcal{P}_{\mathcal{H}}$: Probability measures on $\sigma_{\mathcal{H}}$ 87
$h_1 \oplus \cdots \oplus h_n$: Average of n functions 87

$\mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_n$: Space of average functions 88
H_σ: Union of hypotheses in family H 88
H^n: Cartesian product of hypothesis spaces in family H 88
(X, ρ): General metric space 92
$\mathcal{N}(\varepsilon, X, \rho)$: Smallest ε-cover 92
$\mathcal{M}(\varepsilon, X, \rho)$: Packing number 92
$\mathrm{dimp}(\mathcal{H})$: Pseudo-dimension 100
$d_{[P,l_\mathcal{G}]}$: Metric on a component of a composition 101
$C_{l_\mathcal{G}}(\varepsilon, \mathcal{F})$: ε-capacity of a component of a composition 102
$d^*_{[P,l_\mathcal{G}]}$: Metric on component of a composition (2) 105
$\mathcal{C}^*_{l_\mathcal{G}}(\varepsilon, \mathcal{F})$: ε-capacity of a component of a composition (2) 105
$d_{L^1(P)}$: Metric on functions 108
$\mathcal{C}(\varepsilon, \mathcal{H}, d_{L^1})$: Capacity of function space 108

CHAPTER 2

Ordinary Learning

Haussler's (1992) *statistical decision theory* formulation of machine learning is introduced and the notion of a *Probably Consistent* or PC learning procedure is defined. Haussler's treatment is then reproduced to give a theorem bounding the number of examples required for PC learning. A probably consistent learner needs only to perform at the same level in practice as it does in training, there is no requirement that it actually perform well. For the learner to be useful it must be both probably consistent (PC) and *Empirically Correct* (EC). An empirically correct learner is one that performs well in training. Both PC and EC together imply a version of the well-known PAC (Probably Approximately Correct) criterion introduced by Valiant (1984). However it is shown that for very general learning scenarios, *in the absence of prior information*, it is impossible for a learner to be PAC.

2.1. The General Problem

Under the statistical decision theory formulation of machine learning the learner is supplied with a *training set* or *sample* $\vec{z} = \{z_1, z_2, \ldots, z_m\}$, where each element $z_i = (x_i, y_i)$ consists of an *input* $x_i \in X$ and an *outcome* $y_i \in Y$, X and Y being arbitrary sets known as the *input* and *outcome spaces* respectively. The sample is generated by m independent trials from $Z = X \times Y$ according to an unknown joint probability distribution P. In addition the learner is provided with an *action* space A, a loss function $l\colon Y \times A \to [0, M]$ and a *hypothesis*[1] space $\mathcal{H}$ containing functions $h\colon X \to A$. Defining the *expected* or *true loss* of hypothesis h with respect to distribution P as

$$E(h, P) = \int_{X \times Y} l\,(y, h(x))\; dP(x, y), \tag{6}$$

the goal of the learner is to produce a hypothesis $h \in \mathcal{H}$ that has expected loss close to zero. $l(y, h(x))$ is designed to give a measure of the *loss* the learner suffers, when, given an input $x \in X$, it produces an action $h(x) \in A$ and is subsequently shown the outcome $y \in Y$.

If, for each $h \in \mathcal{H}$, a function $l_h : Z \to [0, M]$ is defined by $l_h(z) = l(y, h(x))$ for all $z = (x, y) \in Z$, then $E(h, P)$ can be expressed as the expectation of l_h with respect

[1] In statistical decision theory the hypotheses are more commonly called *decision rules*.

to P,

$$E(h,P) = \langle l_h \rangle_P = \int_Z l_h(z)\, dP(z).$$

Let $l_{\mathcal{H}} = \{l_h : h \in \mathcal{H}\}$. The measure P and the σ-algebra on Z are assumed to be such that all the $l_h \in l_{\mathcal{H}}$ are P-measurable[2].

The learner does not know the exact distribution P, it is only given a sample $\vec{z} = \{z_1, \ldots, z_m\}$ drawn from Z according to P. Hence the learner does not have enough information to find a function $h \in \mathcal{H}$ minimizing the true loss, $\langle l_h \rangle_P$. The best the learner can do is search for a function that minimizes the *empirical loss* on the sample $\vec{z}$,

$$E(h,\vec{z}) = \frac{1}{m}\sum_{i=1}^{m} l_h(z_i). \tag{7}$$

Let $\langle l_h \rangle_{\vec{z}}$ be a synonym for $E(h,\vec{z})$.

Translating, for example, the problem of training a neural network to recognize the hand-written character "A" into this framework: the input space X would be the space of all possible images, the outcome space Y would be $\{0,1\}$—"1" for an example of "A" and "0" otherwise. P would be the environmental distribution over characters, with $P(y|x)$ being either 1 or 0 according to whether the classification y is correct for x or not. $\mathcal{H}$ would be a class of feed-forward neural networks mapping X into some interval of the real line, say $[0,1]$, which would be the action space A. Typically, in these problems the loss function l is the squared error: $l(y,a) = (y-a)^2$ for all $(y,a) \in Y \times A$. The true loss of a hypothesis $h \in \mathcal{H}$ is then the expected mean-squared difference between $h(x)$ and y on a sample (x,y) drawn at random according to P, and the empirical loss on a training sample $\vec{z} = (z_1, \ldots, z_m), z_i = (x_i, y_i)$ is the *mean-squared error* of the hypothesis with respect to the sample: $\langle l_h \rangle_{\vec{z}} = 1/m \sum_{i=1}^{m} (h(x_i) - y_i)^2$. A hypothesis h with zero empirical loss correctly classifies all examples in the training set and if h also has zero true loss it will correctly classify all examples in practice.

The purpose of the loss function in this formulation is that it allows the learner to use the same hypothesis space $\mathcal{H}$ under many different circumstances, simply by changing the definition of the loss function. This permits results derived under specific learning scenarios to apply in a wider variety of situations. In Haussler (1992), section 1.1, many more quite diverse examples of machine learning and classical statistical problems are translated into this framework.

The behaviour of the learner is to take in samples $\vec{z} \in Z^m$ and produce as output hypotheses in $\mathcal{H}$. Thus the learner can formally be denoted as a map $\mathcal{A}$ from the space of all samples into $\mathcal{H}$,

$$\mathcal{A} : \bigcup_{m \geq 1} Z^m \to \mathcal{H}.$$

This notation implies that $\mathcal{A}$ is a function, however it is a trivial matter to also treat stochastic learners within this framework.

The two main questions to ask of any learner $\mathcal{A}$ are under what conditions will the learner be able to produce a hypothesis h with small empirical loss (7), and with what confidence will the empirical loss be close to the true loss (6)? The latter problem is

[2] The space of measures $\mathcal{P}_{l_{\mathcal{H}}}$ and the σ-algebra, $\sigma_{l_{\mathcal{H}}}$ in definition A.1 have been designed explicitly to fulfill this purpose.

tackled over the next two sections, while discussion of the former is postponed until section 2.4.

2.2. Deviation of Empirical and True Loss

Before the conditions ensuring a close agreement between the empirical loss and true loss can be analysed, it first needs to be decide what "close" means in this context. The empirical loss (7) and true loss (6) are both positive numbers so deciding how to measure their difference is a matter of choosing a metric on R^+. Haussler (1992) introduced the one parameter family of metrics, d_ν, defined as follows. For all $x, y \in \mathrm{R}^+$ and all $\nu > 0$, let

$$d_\nu(x, y) = \frac{|x - y|}{\nu + x + y}.$$

Note that the range of d_ν is $[0, 1)$. The advantage of this metric over the usual Euclidean distance $d(x, y) = |x - y|$ is that it is a relative, rather than absolute measure of distance, and so will be more appropriate when x and y are large. However, note that when either x or y is zero the condition $|x - y| < \varepsilon$ is equivalent to $d_\varepsilon(x, y) < \frac{1}{2}$ and so results on the deviation between the true and empirical loss under the d_ν metric can be translated into results under the normal Euclidean metric if the empirical loss is known to be zero. Some other useful properties of the d_ν metric are given in lemma B.1.

It is now possible to define precisely what is meant by PC learning within this machine learning context.

DEFINITION 2.1. A learning algorithm $\mathcal{A} : \cup_{m \geq 1} Z^m \to \mathcal{H}$ is *Probably Consistent* or *PC* with respect to the set of probability measures $\mathcal{P}$ on Z if for all $\nu > 0$, $0 < \alpha < 1$, $0 < \delta < 1$ and $P \in \mathcal{P}$, there exists a finite integer N such that

$$\Pr\left\{ \vec{z} \in Z^m : d_\nu\left(\langle l_{\mathcal{A}(\vec{z})} \rangle_P, \langle l_{\mathcal{A}(\vec{z})} \rangle_{\vec{z}} \right) > \alpha \right\} < \delta$$

for all $m \geq N$, where $\vec{z}$ is generated by m independent trials from Z according to P[3].

To see how a learner $\mathcal{A}$ could fail to be PC, consider the situation in which $\mathcal{A}$ simply rote-learns any sample it is given. That is, it produces a hypothesis h that functions by comparing any input x with its training examples $x_1, \ldots, x_m$ and if it finds a match x_i outputs the action a minimizing $l(y_i, a)$. If it doesn't find a match it simply chooses a random action. The behaviour of such a learner off the sample, no matter how large the sample is, will clearly bear no relation to its behaviour on the sample and so in general such a learner would not be PC. To prevent such a scenario the learner must be provided with a restricted hypothesis space $\mathcal{H}$ from which to choose its hypotheses. The next two sections are devoted to demonstrating exactly how the PC-ness of a learner is critically governed by the nature of $\mathcal{H}$.

[3] It is being assumed that $\mathcal{A}$ is not so pathological in its behaviour that the set in definition 2.1 is unmeasurable

2.3. Bounding the Deviation

Making no assumptions about the nature of $\mathcal{A}$ or P, the probability in definition 2.1 can be bounded by:

$$\begin{aligned}(8)\quad &\Pr\left\{\vec{z}\in Z^m : d_\nu\left(\langle l_{\mathcal{A}(\vec{z})}\rangle_P, \langle l_{\mathcal{A}(\vec{z})}\rangle_{\vec{z}}\right) > \alpha\right\} \\ &\qquad\qquad \leq \Pr\left\{\vec{z}\in Z^m : \exists l_h \in l_{\mathcal{H}} : d_\nu\left(\langle l_h\rangle_P, \langle l_h\rangle_{\vec{z}}\right) > \alpha\right\}.\end{aligned}$$

This bound is quite crude, as it is simply being said that the deviation between the true and empirical loss on $\vec{z}$ of the learner's hypothesis $\mathcal{A}(\vec{z})$ can't be greater than α if there are in fact no functions in the hypothesis space $\mathcal{H}$ with such a deviation. This is effectively a worst-case scenario; the inequality above becomes an equality if the learner always chooses the hypothesis with the largest deviation between empirical and true loss. Clearly no learner will ever achieve this, except possibly by chance, as the learner doesn't know the true loss and hence cannot bias its search towards regions of the hypothesis space with high deviation between empirical and true loss (not that it would want to in any case). Typically the learner is in fact stochastic: given a sample $\vec{z}$ the learner produces a *probability measure* $\mathcal{A}(\vec{z})$ on the function space $\mathcal{H}$, and then selects a function $h \in \mathcal{H}$ according to $\mathcal{A}(\vec{z})$. For example, in neural networks trained with simple gradient descent, the learner starts with an initial distribution (gaussian, uniform, etc) over the weights in the network, and then iteratively updates them in the direction of steepest descent of the error function, which, in this framework, is the empirical loss of the network on the sample $\vec{z}$. This process usually proceeds until no further improvement in the error is seen i.e. the weights are at a local minimum in the error surface. Thus the network (hypothesis) chosen by the learner is selected from a distribution over the local minima in the error surface for sample $\vec{z}$, where the probability of a local minima is the probability of its attracting basin in weight space under the initial distribution on the weights. In the absence of any other information there is no reason to suppose that the hypotheses with large deviation between empirical loss on $\vec{z}$ and true loss are more concentrated in the local minima of the error surface than elsewhere in the surface. Hence, it can be assumed that the probability of the learner choosing a hypothesis with large deviation is simply equal to the measure (under the initial distribution on the weights) of the hypotheses with large deviation. Unfortunately it is not clear how to even begin calculating such a measure and so, rather than weighting each sample $\vec{z}$ by the measure of the region of weight space with high deviation between empirical loss on $\vec{z}$ and true loss, it is weighted by one if the set of hypotheses with high deviation on $\vec{z}$ is non-empty, and zero otherwise.

To state the main result bounding the right hand side of (8), some definitions from pseudo-metric space theory are required. These are given in definitions B.2, B.3 and B.4 but for continuity are reproduced here.

Firstly a *pseudo-metric* is just a metric without the condition that points zero distance apart necessarily be identical. For the present discussion the following pseudo-metric is significant.

DEFINITION 2.2. Given any probability measure P on Z and any space of functions $\mathcal{F}: Z \to [0, M]$, define the pseudo-metric d_P on $\mathcal{F}$ by

$$d_P(f, g) = \int_Z |f(z) - g(z)|\, dP(z),$$

for all $f, g \in \mathcal{F}$.

Note that d_P is only a pseudo-metric because f and g could differ on a set of measure zero and still be equal under d_P. The pair $(\mathcal{F}, d_P)$ is called a *pseudo-metric space*. An *ε-cover* for $(\mathcal{F}, d_P)$ is any subset $F \subseteq \mathcal{F}$ such that for all $f \in \mathcal{F}$ there exists an $f' \in F$ such that $d_P(f, f') \leq \varepsilon$. F is called an ε-separated set if $\rho(f, f') > \varepsilon$ for all distinct $f, f' \in F$. Denote the size of the smallest ε-cover of $(\mathcal{F}, d_P)$ by $\mathcal{N}(\varepsilon, \mathcal{F}, d_P)$ and the size of the largest ε-separated subset by $\mathcal{M}(\varepsilon, \mathcal{F}, d_P)$. Define $\sigma_{\mathcal{F}}$ to be the σ-algebra on Z generated by all inverse images under any $f \in \mathcal{F}$ of any Borel set in $[0, M]$. Define $\mathcal{P}_{\mathcal{F}}$ to be the set of all probability measures P on Z such that all functions $f \in \mathcal{F}$ are P-measurable. The *ε-capacity* of $\mathcal{F}$ is defined by

$$\mathcal{C}(\varepsilon, \mathcal{F}) = \sup_{P \in \mathcal{P}_{\mathcal{F}}} \mathcal{N}(\varepsilon, \mathcal{F}, d_P).$$

If the supremum does not exist then the capacity is defined to be ∞.

The main theorem bounding the probability of deviation between true and empirical loss in equation (8) can now be stated.

THEOREM 2.1. *Let $\mathcal{H}$ be a family of maps from X into A and l be a loss function $l: Y \times A \to [0, M]$. Assume that $\mathcal{H}$ and l are such that $l_{\mathcal{H}}$ is permissible*[4]*. Let $\vec{z}$ be generated by m independent draws from Z according to any probability measure P. Then for all $\nu > 0$, $0 < \alpha < 1$*

$$\Pr\{\vec{z} \in Z^m : \exists l_h \in l_{\mathcal{H}} : d_\nu(\langle l_h \rangle_P, \langle l_h \rangle_{\vec{z}}) > \alpha\} \leq 4\mathcal{C}(\alpha\nu/8, l_{\mathcal{H}}) e^{-\frac{\alpha^2 \nu m}{8M}}.$$

This theorem is essentially due to Haussler (1992), Pollard (1984) and others going back even further, but it has been derived in this work as corollary B.7 of a more general result (theorem B.6) in section B.3.

Using inequality 8, theorem 2.1 implies that to guarantee with probability $1 - \delta$ a d_ν deviation of less than α between a learner's true and empirical loss, it must be provided with a sample of size m where

$$m > \frac{8M}{\alpha^2 \nu} \ln\left(\frac{4\mathcal{C}(\alpha\nu/8, l_{\mathcal{H}})}{\delta}\right). \tag{9}$$

Thus, as long as $\mathcal{C}(\alpha\nu/8, l_{\mathcal{H}}) < \infty$ the learner is guaranteed to be PC. Thus the capacity of a hypothesis space $\mathcal{H}$ determines critically PC-learnability using $\mathcal{H}$. However, be aware that many approximations go into establishing theorem 2.1 (see the proof in appendix B) and so in general the above bound on m will not be tight.

2.4. EC and its incompatibility with PC

PC learning ensures consistency between empirical estimates and true errors, but does not have anything to say about exactly how good the learner's performance is. This is covered by the *EC* criterion.

To begin with, for any probability measure P on Z define $r^*(P)$ to be the infimum of $\langle l_h \rangle_P$ over *all* functions $h: X \to A$ such that l_h is P-measurable. Note that the function h need not come from the learner's hypothesis space $\mathcal{H}$.

[4] This relatively benign measure theoretic condition is needed to ensure that sets such as the one appearing in the right hand side of (8) are measurable. The definition of permissibility is given in appendix A.

DEFINITION 2.3. Let $\nu > 0$ and $0 < \alpha < 1$ be two real parameters. A learning algorithm $\mathcal{A} \colon \cup_{m \geq 1} Z^m \to \mathcal{H}$ is (α, ν)-*Empirically Correct* or (α, ν)-EC (or just "EC" for short) with respect to the set of probability measures $\mathcal{P}$ if for all $P \in \mathcal{P}$ and $0 < \delta < 1$ there exists a positive integer N such that for all $m \geq N$,

$$\Pr\{\vec{z} \in Z^m \colon d_\nu \left(\langle l_{\mathcal{A}(\vec{z})} \rangle_{\vec{z}}, r^*(P) \right) > \alpha\} < \delta.$$

Thus for an algorithm $\mathcal{A}$ to be (α, ν)-EC it must, for sufficiently large samples and with high probability, produce hypotheses with empirical loss within α (under the d_ν metric) of the true loss of the *best possible hypothesis*. Requiring the learner to match closely the best possible performance may seem somewhat harsh, as often in machine learning one requires only that the learner approximates the performance of the best possible hypothesis chosen from within its own hypothesis space $\mathcal{H}$. Clearly that requirement tells nothing about how well the learner is actually doing—for example if the learner was performing image recognition with a hypothesis space containing only the hypothesis that "everything is a picture of a dog" then the learner would always do brilliantly in its own terms but would be of no practical use.

The parameterisation (α, ν) of the definition of EC allows a level of accuracy to be specified for the learner's hypotheses—the learner is not necessarily required to produce arbitrarily good hypotheses, reflecting the fact that often in practice approximate solutions are sufficient.

To show how EC and PC learning come into conflict "Probably Approximately Correct" or PAC learning is defined, PC and EC together are shown to imply PAC and then a simple but representative situation is given in which PAC implies not PC.

DEFINITION 2.4. Let $\nu > 0$ and $0 < \alpha < 1$ be two real parameters. A learning algorithm $\mathcal{A} \colon \cup_{m \geq 1} Z^m \to \mathcal{H}$ is (α, ν)-*Probably Approximately Correct* or (α, ν)-PAC or just PAC for short with respect to the set of probability measures $\mathcal{P}$ if for all $P \in \mathcal{P}$ and all $0 < \delta < 1$, there exists a positive integer N such that for all $m \geq N$,

$$\Pr\{\vec{z} \in Z^m \colon d_\nu \left(\langle l_{\mathcal{A}(\vec{z})} \rangle_P, r^*(P) \right) > \alpha\} < \delta.$$

So in PAC learning the learner is required to eventually start producing hypotheses that have true loss arbitrarily close to the best possible. In Valiant's (1984) original definition of PAC he essentially required that the sample size N grow at most polynomially in the relevant parameters $1/\delta$, $1/\alpha$ and $1/\nu$. However, this approach conflates the problems of *computational complexity* and *sample complexity* and for the purposes of this work the issue of sample complexity is far more important. Thus restrictions on the rate of growth of the sample size are not given in this definition of PAC.

That EC learning and PC learning together imply PAC is easily seen through the triangle inequality for d_ν. If the learner is PC then it is possible to sample sufficiently many times so that the probability of

$$d_\nu \left(\langle l_{\mathcal{A}(\vec{z})} \rangle_{\vec{z}}, \langle l_{\mathcal{A}(\vec{z})} \rangle_P \right) > \alpha/2$$

is arbitrarily small. If the learner is $(\alpha/2, \nu)$-EC then the probability

$$d_\nu \left(\langle l_{\mathcal{A}(\vec{z})} \rangle_{\vec{z}}, r^*(P) \right) > \alpha/2$$

may also be made arbitrarily small and hence by the triangle inequality for d_ν the probability of

$$d_\nu \left(\langle l_{\mathcal{A}(\vec{z})} \rangle_P, r^*(P) \right) > \alpha,$$

can be made arbitrarily small. Thus, PC and $(\alpha/2, \nu)$-EC together imply (α, ν)-PAC.

To satisfy the PAC condition the learner's hypothesis space must contain (α, ν) approximations to the best possible for any $P \in \mathcal{P}$. The following argument shows how, at least for measurable functions on R and for $\frac{\alpha\nu}{1-\alpha} \leq \frac{1}{8}$, this is incompatible with PC learning.

Suppose the learner is trying to learn Borel measurable functions from $X = [0,1]$ into $Y = [0,1]$. Denote the set of all such functions by $\mathcal{F}$. Assume the hypotheses it produces are also Borel measurable functions $[0,1] \to [0,1]$, so that $A = Y = [0,1]$. Denote as usual the learner's hypothesis space by $\mathcal{H}$. Let the loss function, $l\colon [0,1] \times [0,1] \to [0,1]$ be $l(y,y') = |y - y'|$ for all $y, y' \in Y$. Let P be the uniform probability measure on $[0,1]$ and given $f \in \mathcal{F}$ denote by P_f the probability measure induced on $X \times Y$ by generating samples $(x, f(x)) \in X \times Y$ with x generated according to P. This is what would normally be called "sampling from f". Let $\mathcal{P}_{\mathcal{F}} = \{P_f : f \in \mathcal{F}\}$. By defining $d_P(f,h) = \int_X |f(x) - h(x)|\, dP(x)$, for all P-measurable f and h, observe that

$$\begin{aligned}
\langle l_h \rangle_{P_f} &= \int\limits_{X \times Y} l_h(x,y)\, dP_f(x,y) \\
&= \int_X l(h(x), f(x))\, dP(x) \\
&= \int_X |h(x) - f(x)|\, dP(x) \\
&= d_P(h,f).
\end{aligned}$$

Note also that for all $P_f \in \mathcal{P}_{\mathcal{F}}$, $r^*(P_f) = 0$.

What is required of the learner's hypothesis space $\mathcal{H}$ to ensure (α, ν)-PAC learning? To start with, for all $P_f \in \mathcal{P}_{\mathcal{F}}$ there must exist at least one $h \in \mathcal{H}$ such that $d_\nu\left(\langle l_h \rangle_{P_f}, r^*(P_f)\right) \leq \alpha$. But as $r^*(P_f) = 0$,

$$d_\nu\left(\langle l_h \rangle_{P_f}, r^*(P_f)\right) \leq \alpha \quad \Rightarrow \quad \langle l_h \rangle_{P_f} \leq \frac{\alpha\nu}{1-\alpha} \quad \Rightarrow \quad d_P(h,f) \leq \frac{\alpha\nu}{1-\alpha}$$

Set $\varepsilon = \frac{\alpha\nu}{1-\alpha}$ and it has been shown that for (α, ν)-PAC learning, $\mathcal{H}$ must be an ε-cover for $(\mathcal{F}, d_P)$. Now suppose H is an ε-cover of $(\mathcal{H}, d_P)$. By the triangle inequality for d_P H must also be a 2ε-cover for $\mathcal{F}$ and so $\mathcal{C}(\varepsilon, \mathcal{H}) \geq \mathcal{C}(2\varepsilon, \mathcal{F})$.

To be certain that the learner is (α, ν)-PAC, in the absence of any information to the contrary, it must be assumed that $\mathcal{F}$ contains all Borel measurable functions $f\colon X \to A$. This implies something about the capacity of $\mathcal{F}$ as follows. For all $n = 1, 2, \ldots$ define the subset I_n of $[0,1]$ by

$$I_n = \bigcup_{m=1}^{2^n - 1} \left[\frac{2m-1}{2^n}, \frac{2m}{2^n}\right].$$

For any $m, n \geq 0$ let $I_n \triangle I_m$ be the symmetric difference of I_n and I_m (the set of points $x \in [0,1]$ such that $x \in I_n$ and $x \notin I_m$ or $x \in I_m$ and $x \notin I_n$.) By abuse of notation denote the characteristic function of I_n by I_n also, and the characteristic function of $I_n \triangle I_m$ by $I_n \triangle I_m$. Let $F = \{I_n : n = 1, 2, \ldots\}$. Clearly all $f \in F$ are

Borel functions and so $F \subset \mathcal{F}$. For any $n, m \geq 1$ and $n \neq m$,

$$\begin{aligned} d_P(I_n, I_m) &= \int_X |I_n(x) - I_m(x)| \, dP(x) \\ &= \int_0^1 |I_n \triangle I_m(x)| \, dx \\ &= \frac{1}{2}. \end{aligned}$$

Thus F is a $\frac{1}{2}$-separated subset of $(\mathcal{F}, d_P)$ and so for all $\delta \leq \frac{1}{2}$, as $|F| = \infty$, $\mathcal{M}(\delta, \mathcal{F}, d_P) = \infty$ (recall that $\mathcal{M}(\delta, \mathcal{F}, d_P)$ is the size of the largest δ-separated subset of $\mathcal{F}$). So by lemma B.3, for all $\delta \leq \frac{1}{4}$, $\mathcal{N}(\delta, \mathcal{F}, d_P) = \infty$ and so $\mathcal{C}(\delta, \mathcal{F}) = \infty$. Hence for all $\varepsilon \leq \frac{1}{8}$, $\mathcal{C}(\varepsilon, \mathcal{H}) = \infty$. Thus if α and ν are such that $\frac{\alpha\nu}{1-\alpha} \leq \frac{1}{8}$ there is no learning algorithm that is (α, ν)-PAC.

Is this example a realistic one? In a sense, the answer is no. The space of all measurable functions is extremely large and contains many functions that are very unlikely to be seen in practice, particularly the more "wiggly" members of F above. In fact by assuming that the functions in $\mathcal{F}$ are limited in their wiggliness (perhaps by putting a bound on the integral of the square of their second derivative) then it is fairly easy to see that the capacity of $\mathcal{F}$ will be finite for all $\varepsilon > 0$, hence the learner could get away with using a hypothesis space which is also finite in capacity. This is an example of using *prior information* of the kinds of functions likely to be seen by the learner to turn a non-PAC problem into a PAC one. However that is not the end of the story. Assumptions about smoothness only serve to bias the learners hypothesis space in an absolutely minimal way. The capacity of the learner's hypothesis space would still have to be prohibitively large to ensure EC learning in all situations and hence learning, although not technically impossible, would still be infeasible. This is where the "art" of machine learning comes in—it is the art of using prior knowledge about the problem domain to further bias the hypothesis space so that sample sizes required for PAC learning become managable. The problem with this approach is that it is an "art" and hence it is very much up to the skill and insights of the researcher whether sufficiently many correct biases can be found.

In the next chapter it is shown how, at least for some kinds of learning problems, it is possible to *learn* the bias by learning an appropriate *representation* for the learner's environment.

CHAPTER 3

Representation Learning

The previous chapter demonstrated that in the absence of any prior information PAC learning is impossible. In this chapter the concept of the *environment* of a learning process is defined and it is shown how the environment may be sampled to automatically generate prior information for the learner. Such information can be used to learn an appropriate *representation* for the environment, which can then be used to bias the learner's hypothesis space and greatly improve learning of future tasks drawn from the same environment. Lower bounds for PC learning of representations are given, similar to the ones given for ordinary learning in the previous chapter. These bounds show that for learning environments consisting of a large number of similar tasks, such as character recognition and speech recognition, the number of examples of each task required for PC learning can be greatly reduced compared to that required for ordinary learning. In the final section the results on representation learning are generalised to cover any procedure for learning hypothesis space bias.

3.1. General Framework

The ordinary learning scenario of the previous chapter is summarised by the following diagram ("id" just means the identity function).

$$\begin{array}{ccccc} \mathcal{P} \ni P & & & & \\ \downarrow & & & & \\ Z = & X & \xrightarrow{h \in \mathcal{H}} & A & \\ & \times & & \times & \xrightarrow{l} [0, M] \\ & Y & \xrightarrow{\mathrm{id}} & Y & \end{array}$$

If the learner receives no information about the kinds of tasks $P \in \mathcal{P}$ it is required to learn, it must assume that $\mathcal{P}$ is the most general possible and it was shown in the previous chapter how in general this causes the learner to use a hypothesis space $\mathcal{H}$ so large that PAC-learning using $\mathcal{H}$ is impossible. To enable the learner to use a smaller hypothesis space it must be given some information about $\mathcal{P}$. This can be modelled by placing a probability measure Q on $\mathcal{P}$ so that when the learner receives a task $P \in \mathcal{P}$ to learn, it is assumed to have been drawn from $\mathcal{P}$ according to Q. Q will be referred to as the *environment* of the learning process. For Q to be well defined there needs to be a σ-algebra on $\mathcal{P}$. The appropriate one to use will become clear later in the chapter.

To enable the learner to get some idea of the environment in which it is learning, and hopefully then extract some of the bias inherent in the environment, it is provided not just with a single sample $\vec{z} = (z_1, \ldots, z_m)$, sampled according to some probability measure $P \in \mathcal{P}$, but with n such samples $(\vec{z}_1, \ldots, \vec{z}_n)$. Each sample $\vec{z}_i = (z_{i1}, \ldots, z_{im})$, for $1 \leq i \leq n$, is generated by first sampling from $\mathcal{P}$ according to Q to generate P_i, and then sampling m times from P_i to generate $\vec{z}_i = (z_{i1}, \ldots, z_{im})$. Thus the entire sample is generated by sampling n times from $\mathcal{P}$ according to Q to generate $P_1, \ldots, P_n$, and then sampling m times from Z according to each P_i. Denote the entire sample by $\mathbf{z}$ and write it as an $n \times m$ (n rows, m columns) matrix over Z:

$$\mathbf{z} = \begin{matrix} z_{11} & \ldots & z_{1m} \\ \vdots & \ddots & \vdots \\ z_{n1} & \ldots & z_{nm} \end{matrix}$$

Denote the $n \times m$ matrices over Z by $Z^{(n,m)}$ and call a sample $\mathbf{z} \in Z^{(n,m)}$ generated by the above process an (n, m) *sample*.

To enable the learner to take advantage of the prior information contained in $\mathbf{z}$, the hypothesis space $\mathcal{H}\colon X \to A$ is split into two sections: $\mathcal{H} = \mathcal{G} \circ \mathcal{F}$ where $\mathcal{F}\colon X \to V$ and $\mathcal{G}\colon V \to A$, where V is an arbitrary set[1]. To simplify the notation this will be written in future as

$$X \xrightarrow{\mathcal{F}} V \xrightarrow{\mathcal{G}} A.$$

$\mathcal{F}$ is called the *representation space* and an individual member f of $\mathcal{F}$ is called a *representation*. The representation learning scenario is summarised by the following diagram.

$$\begin{array}{ccccc} Q & & & & \\ \downarrow & & & & \\ \mathcal{P} \ni P & & & & \\ \downarrow & & & & \\ Z = & X & \xrightarrow{h \in \mathcal{G} \circ \mathcal{F}} & A & \\ & \times & & \times & \xrightarrow{l} [0, M] \\ & Y & \xrightarrow{\mathrm{id}} & Y & \end{array}$$

Based on the information about the environment Q, contained in $\mathbf{z}$, the learner searches for a good representation $f \in \mathcal{F}$. A good representation is one with a small *empirical loss* $E^*_{\mathcal{G}}(f, \mathbf{z})$ on $\mathbf{z}$, where this is defined by

$$E^*_{\mathcal{G}}(f, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} \inf_{g \in \mathcal{G}} \langle l_{g \circ f} \rangle_{\vec{z}_i}, \tag{10}$$

where $\vec{z}_i = (z_{i1}, \ldots, z_{im})$ denotes the ith row of $\mathbf{z}$. The empirical loss of f with respect to $\mathbf{z} \in Z^{(n,m)}$ is a measure of how well the learner can learn $\mathbf{z}$ using f, assuming that the learner is able to find the best possible $g \in \mathcal{G}$ for any given sample $\vec{z} \in Z^m$. For example, if the empirical loss of f on $\mathbf{z} = (\vec{z}_1, \ldots, \vec{z}_n)$ is zero then it is possible for the learner to find a function $g_i \in \mathcal{G}$, for each i, $1 \leq i \leq n$, such that the ordinary

[1] That is, $\mathcal{H} = \{g \circ f : g \in \mathcal{G}, f \in \mathcal{F}\}$.

empirical loss, $\langle l_{g_i \circ f} \rangle_{\vec{z}_i}$, is zero. The empirical loss of f is an estimate of the *true loss* of f (with respect to Q), where this is defined by

$$E^*_{\mathcal{G}}(f, Q) = \int_{\mathcal{P}} \inf_{g \in \mathcal{G}} \langle l_{g \circ f} \rangle_P \, dQ(P). \tag{11}$$

The true loss of f with respect to Q is the expected best possible performance of $g \circ f$—over all $g \in \mathcal{G}$—on a distribution chosen at random from $\mathcal{P}$ according to Q. If f has a small true loss then learning using f on a random "task" P, drawn according to Q, will with high probability be successful.

Once the learner has found the best representation it can with respect to the sample $\mathbf{z}$, it then learns using the *restricted* hypothesis space $\mathcal{G} \circ f = \{g \circ f : g \in \mathcal{G}\}$. That is, the learner will be fed samples $\vec{z} \in Z^m$ drawn according to some distribution $P \in \mathcal{P}$, which in turn is drawn according to Q, and will search $\mathcal{G} \circ f$ for a hypothesis $g \circ f$ with small empirical loss on $\vec{z}$. Intuitively, if the capacity of $\mathcal{F}$ is much greater than the capacity of $\mathcal{G}$, then the number of samples required to learn using $\mathcal{G} \circ f$ will be much less than the number of samples required to learn using the full space $\mathcal{G} \circ \mathcal{F}$, a fact that is proved later in this chapter. Hence, if the learner can find a good representation f and the sample $\mathbf{z}$ is large enough, learning using $\mathcal{G} \circ f$ should be considerably quicker and more reliable than learning using $\mathcal{G} \circ \mathcal{F}$.

3.2. Motivation

Do the environments encountered in practice possess representations? In many important cases it is quite plausible that they do. Take for example the problem of character recognition. It seems a plausible hypothesis that the kinds of transformations one can apply to characters and still leave them essentially unchanged (i.e still recognisably the same character) are the same, regardless of the character concerned (a simple example is small rotations of the character). If this is true then there exists a representation f mapping the space of images of characters X into some other space V such that if a "character invariant transformation" is applied to character x, the value of $f(x)$ will remain "relatively unchanged"[2]. The same idea applies to speech recognition. It seems individual words are invariant under the same kinds of transformations and so a representation should exist that is insensisitive to such transformations. Recognizing faces is yet another example (see in this context the interesting paper by Weiss and Edelman (1993)). At any rate, the question of whether a useful representation exists in these cases is an empirical one—it can be verified either by examining the brains of biological learners or by attempting to learn a representation for the environments concerned. In the next section it is demonstrated how that task is greatly facilitated by a most excellent result concerning the sample complexity of representation learning.

If the above environments do indeed possess representations, then there is also strong evidence that the space $\mathcal{G}$ is quite small because humans are able to learn to recognise spoken words and written characters from a relatively small number of examples. The ability of humans to learn faces from just one example is evidence of an even smaller $\mathcal{G}$ for this task. Hence learning of an appropriate representation in

[2] This statement only makes sense if there is some kind of distance measure for comparing elements of V. It is shown in chapter 6 how one may be induced, by "pulling back" any distance measure on R, through the loss function l and then through $\mathcal{G}$.

all these environments is likely, by the argument at the end of the previous section, to greatly enhance learning of future tasks from the same environment.

3.3. The Advantage of Learning Many Tasks

In ordinary learning the learner would be learning one "task" P from the environment $\mathcal{P}$ and in the absence of any prior information would be using the full hypothesis space $\mathcal{G} \circ \mathcal{F}$. If $\mathcal{G} \circ \mathcal{F}$ is an appropriate hypothesis space for the environment then it will contain a reasonable, say $(\alpha/2, \nu)$-approximation to the best achievable on the task P, and thus as long as the learner can find such a hypothesis it will be $(\alpha/2, \nu)$-EC. By the argument following the definition of PAC-learning in the previous chapter, the learner will be (α, ν)-PAC if it also samples sufficiently many times to ensure that it is PC. By equation (9), page 19, this will be fulfilled if the learner samples

$$m = O\left(\log \mathcal{C}\left(\varepsilon, l_{\mathcal{G} \circ \mathcal{F}}\right)\right)$$

times from Z according to P, where $\varepsilon = \alpha\nu/8$ and only the capacity dependence of the bound has been written. In section C.3 it is proved that the capacity of a composition is proportional to the product of the capacities of the individual spaces, hence (loosely speaking—this will all be made rigorous shortly),

$$m = O\left(\log \mathcal{C}\left(\varepsilon, \mathcal{G}\right) + \log \mathcal{C}\left(\varepsilon, \mathcal{F}\right)\right). \tag{12}$$

In contrast, suppose instead the learner has received n tasks $P_1, \ldots, P_n$ drawn from $\mathcal{P}$ according to the environment Q, and is learning a representation f appropriate to the n tasks, then it turns out that the learner need only sample

$$m = O\left(\log \mathcal{C}\left(\varepsilon, \mathcal{G}\right) + \frac{\log \mathcal{C}\left(\varepsilon, \mathcal{F}\right)}{n}\right) \tag{13}$$

times from each task to be (α, ν)-PAC (take the empirical and true losses of the learner on the n tasks to be the the average of the ordinary empirical and true losses on each individual task and require for PC learning that these averages be close with high probability. Once again rigorous definitions are given shortly).

Thus, if the capacity of $\mathcal{F}$ is much greater than the capacity of $\mathcal{G}$, then by learning n tasks instead of one there will be an *n-fold reduction in the number of examples required per task for PAC learning.* This suggests a new approach to machine learning. Instead of only learning the task required, learn as many related tasks as possible. For instance, if a neural network is being trained to recognise handwritten characters then instead of only teaching it to recognise the roman alphabet, teach it kanji characters, and hiragana and arabic and the digits and so on. Many fewer examples of each character will be required to learn the lot, and if the number of characters is sufficiently great (greater than $O\left(\log \mathcal{C}\left(\varepsilon, \mathcal{F}\right)\right)$, as will be shown later) then the resulting representation will be a good one for learning new characters. In fact learning the kanji characters would be a very good test of this theory because there are enough of them (thousands) to ensure with high probability that the representation will be a good one, and to have plenty left over for testing the ease of learning using the representation on unseen characters. Note that this is a new form of "generalisation", one level of abstraction higher than the usual meaning of generalisation, for within this framework a learner generalises well if, after having learnt many different characters, it is able to learn new characters easily. Thus, not only is the learner required to generalise well in the ordinary sense by generalising well on the characters in the

training set, but also the learner is expected to have "learnt to learn" characters in general.

In a similar way, if a representation exists that is appropriate for spoken word recognition then the learner should be trained on many words, perhaps on many languages too, so that the required number of examples of each word is greatly reduced and so that the resulting representation may be useful in learning new words. Character recognition and speech recognition are but two of many problems that should lend themselves to representation learning.

In chapter 4 a gradient-based algorithm for training neural networks to learn representations is given and the results of some simple experiments presented that fully support the claims made here for the efficacy of representation learning. The remainder of this chapter is devoted to a theoretical justification for these claims, and to providing rigorous lower bounds on the number of examples required for PAC-like learning of representations.

3.4. Philosophical Aside

As with ordinary learning, the two main issues involved in representation learning are whether the learner can find a representation with small empirical loss (10) on a given (n, m) sample, and whether the empirical loss is in fact close to the true loss (11). This is the EC/PC question again and a similar argument to the one given at the end of chapter 2 can be used to show that EC and PC are also incompatible for representation learning, in the absence of any prior information about the kinds of environments Q that the representation learner is likely to be operating within. Using the same reasoning that led to the introduction of representation learning (or at least automated hypothesis space bias) in the first place, it would be tempting to suppose that Q is drawn from a collection $\mathcal{Q}$ of environments according to some *super-environmental measure* R, and attempt to generate prior information by sampling from $\mathcal{Q}$ according to R. Unfortunately, the same arguments would show PC and EC learning to still be incompatible at this new level of abstraction, and so one would be forced to go higher and higher in a never-ending process. The conclusion is that no finite sample, regardless of whether it is used for representation learning, ordinary learning, or super-environmental learning or whatever, will ever be big enough to solve the PC/EC dilemma. This conclusion should not really be surprising, it is equivalent to the fact that there is no *a priori* basis for induction, no matter what the circumstance. However this problem is basic to all of science and philosophy, and hence failing to solve it is no reason for discarding a particular line of enquiry.

In fact there are "superselection" rules, such as *Occam's Razor*, that, although they are *a priori* unjustified, do seem to work in practice as a means of breaking the induction dilemma. The modern version of Occam's Razor is *Kolmogorov Complexity*. The Kolmogorov Complexity of an object is essentially the length (in bits) of the shortest program that can be fed into a universal computer causing it to produce a description of the object as output[3]. It is trivial to show, up to an additive constant, that the Kolmogorov Complexity of an object is not affected by the choice of universal computer used to describe it. Thus Kolmogorov Complexity can be used to compare the complexity of two solutions to a particular problem. For example, it could be used to choose between two candidate representations, both with zero

[3]See Cover and Thomas (1991), chapter 12, for a highly readable discussion of Kolmogorov Complexity.

empirical loss on a training set $\mathbf{z}$. In general a representation that simply rote-learns $\mathbf{z}$ will have far higher complexity than one which takes advantage of regularities in the data, and hence will be rejected by this criterion (as desired). A general procedure for learning representations based on complexity is to always search among those representations with small complexity first. If no solution can be found, search amongst those representations with higher complexity and so on until a solution is found. This is equivalent to providing the learner with a sequence of representation spaces $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \subset \ldots$, which in the limit would include all possible representations[4], and hence in the limit the learner would be guaranteed to be EC[5,6]. Of course, if the sequence continues too far the capacity of the representation space $\mathcal{F}_n$ will become too big to ensure PC learning and at that stage the learner would have no choice but to collect further samples from the environment. However the learner is guaranteed to eventually find a solution by this procedure.

The obvious question now is why not do away with representation learning altogether and just apply the same complexity criterion to ordinary learning? There are two reasons why not. Firstly, ordinary learning is of no use if the goal is to find a representation that can be used to improve learning performance on further tasks drawn from the same environment. Secondly, by sampling from only one task in an environment containing many similar tasks, the ordinary learner ignores information that is freely available and although it is true that eventually the learner would find a solution to the single task, convergence would be greatly improved if the information from the other tasks was incorporated as well. More succinctly put, if the environment posesses a representation then the learner should not ignore that fact. Ordinary learning procedures are essentially indifferent to the existence of a representation in the environment.

Of course much more work needs to be done to turn these vague ideas into practical algorithms, but at least *in principle* they show that the PC/EC dilemma is not utterly intractable (provided that "simplicity" is truly a superselection criterion applicable in this universe). So for the remainder of this work it will be assumed that the hypothesis space $\mathcal{G} \circ \mathcal{F}$ has been chosen so that EC representation learning is possible, and attention focused instead on the problem of quantifying the conditions necessary for PC learning.

3.5. Bounds for PC Representation Learning

There are two ways that the generalisation performance of a representation learner can be measured. The first is how well the representation performs in practice, as measured by expression (11), page 25, and the second is how well the learner generalises only on the n tasks from its (n, m) sample. The analysis of both these problems is performed in this section, the latter leading to the relation (13) on page 26.

However, before proceeding, a slight complication with the form of the empirical loss (10) needs to be discussed, and some preliminary definitions must also be introduced.

[4] At least all computable ones, and its not worth bothering about the problem of environments that are generated by non-computable representations because no solution could ever be computed for them, no matter what learning procedure was used.

[5] Such a procedure is closely related to what Vapnik (1982) calls *structural risk minimization*.

[6] The fact that Kolmogorov complexity is not computable prevents this procedure from being carried to its ultimate limit. However the necessary finiteness of any real-world computational procedure means that the limit can never be reached anyway.

3.5.1. Computing the Empirical Loss. For the learner to compute (10) it must be able to compute $\inf_{g\in\mathcal{G}}\langle l_{g\circ f}\rangle_{\vec{z}}$. If the infimum is not attained in $\mathcal{G}$ for the sample $\vec{z}$ and representation f then it is unrealistic to assume that the learner can determine the infimum, and even if the infimum is attained it is unreasonable to force the learner to find it. In fact, for EC learning the learner is only ever required to get (α,ν)-close to the best possible. Thus, although for mathematical simplicity definitions (10) and (11) are used, the results presented here can easily be extended to cover the case in which the learner is only required to find on any sample $\vec{z}\in Z^m$ a function $g^*_{\vec{z}}$ satisfying

$$d_\nu\left(\langle l_{g^*_{\vec{z}}\circ f}\rangle_{\vec{z}}, \inf_{g\in\mathcal{G}}\langle l_{g\circ f}\rangle_{\vec{z}}\right) < \alpha,$$

for some $\nu > 0$, $0 < \alpha < 1$.

3.5.2. Preliminary Definitions. In searching for a representation minimising the empirical loss (10), it is assumed that the learner generates a sequence of n functions $(g_1\circ f,\ldots,g_n\circ f)$, where each $g_i\circ f$ is a map from X into A minimizing the empirical loss $\langle l_{g_i\circ f}\rangle_{\vec{z}_i}$. In what follows the loss of such sequences with respect to (n,m)-samples $\mathbf{z}\in Z^{(n,m)}$ needs to be defined, and also the loss with respect to sequences of distributions $\vec{P}=(P_1,\ldots,P_n)$. Toward this end the following general definitions are made. Given a sequence of functions $\vec{h}=(h_1,\ldots,h_n)$, with $h_i\colon X\to A$, and a sample $\mathbf{z}\in Z^{(n,m)}$, let $\vec{z}_i=(z_{i1},\ldots,z_{in})$ denote the ith row of $\mathbf{z}$ and define the *empirical loss of* $\vec{h}$ *with respect to* $\mathbf{z}$ by

$$E(\vec{h},\mathbf{z}) = \frac{1}{n}\sum_{i=1}^{n}\langle l_{h_i}\rangle_{\vec{z}_i} = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m} l_{h_i}(z_{ij}). \tag{14}$$

This is just the average over the n training sets $\vec{z}_1,\ldots,\vec{z}_n$ of the individual empirical losses of the hypotheses $h_1,\ldots,h_n$. Given a sequence of probability measures $\vec{P}=(P_1,\ldots,P_n)$, define the *true loss of* $\vec{h}$ *with respect to* $\vec{P}$ by

$$E(\vec{h},\vec{P}) = \frac{1}{n}\sum_{i=1}^{n}\langle l_{h_i}\rangle_{P_i}. \tag{15}$$

This is just the average of the n *true losses* of the hypotheses $h_1,\ldots,h_n$.

Given a family of functions $\mathcal{H}\colon X\to A$, define $\mathcal{H}^n\colon X^n\to A^n$ to be the n-fold Cartesian product of $\mathcal{H}$, $\mathcal{H}^n=\{(h_1,\ldots,h_n): h_i\in\mathcal{H}, 1\le i\le n\}$ where $(h_1,\ldots,h_n)$ $(x_1,\ldots,x_n)=(h_1(x_1),\ldots,h_n(x_n))$. Distinguish a special subset of $\mathcal{H}^n$, consisting of all functions from X^n into A^n of the form $(h,\ldots,h)$ and denote this subset by $\overline{\mathcal{H}}$. For all $h\in\mathcal{H}$ set $\bar{h}=(h,\ldots,h)$. Given $\vec{g}=(g_1,\ldots,g_n)\colon V^n\to A^n\in\mathcal{G}^n$ and $\vec{f}=(f_1,\ldots,f_n)\colon X^n\to V^n\in\mathcal{F}^n$, set $\vec{g}\circ\vec{f}=(g_1\circ f_1,\ldots,g_n\circ f_n)$.

With this notation the set of all sequences of the form $(g_1\circ f,\ldots,g_n\circ f)$ where $g_i\in\mathcal{G}$ and $f\in\mathcal{F}$ is denoted by $\mathcal{G}^n\circ\overline{\mathcal{F}}$. Such sequences will be written $\vec{g}\circ\bar{f}$.

In addition to the notion of the empirical loss of f with respect to sample $\mathbf{z}$ (10), and the true loss of f with respect to the environment Q (11), the notion of the loss of f with respect to a sequence of distributions $\vec{P}=(P_1,\ldots,P_n)$ needs to be defined:

$$E^*_{\mathcal{G}}(f,\vec{P}) = \frac{1}{n}\sum_{i=1}^{n}\inf_{g\in\mathcal{G}}\langle l_{g\circ f}\rangle_{P_i}. \tag{16}$$

The learner takes as input samples $\vec{z} \in Z^{(n,m)}$, for any values $m, n \geq 1$, and produces as output hypothesis representations $f \in \mathcal{F}$, so it is a map $\mathcal{A}$ from the space of all possible (n, m) samples into $\mathcal{F}$,

$$\mathcal{A}\colon \bigcup_{\substack{n\geq 1\\ m\geq 1}} Z^{(n,m)} \to \mathcal{F}.$$

As with ordinary learning, the results developed here can be easily generalised to cover stochastic learners.

3.5.3. Generalisation on n tasks. To learn a representation for the environment Q, the learner first samples n times from $\mathcal{P}$ according to Q to generate $\vec{P} = (P_1, \ldots, P_n)$, and then samples m times from Z according to each P_i, $1 \leq i \leq n$, to generate the (n, m) sample

$$\mathbf{z} = \begin{matrix} z_{11} & \cdots & z_{1m} \\ \vdots & \ddots & \vdots \\ z_{n1} & \cdots & z_{nm} \end{matrix}$$

The learner then searches for n hypotheses from $\mathcal{G} \circ \mathcal{F}$, $\vec{g} \circ \bar{f} = (g_1 \circ f, \ldots, g_n \circ f)$, all using the same representation f and all with empirical loss $\langle l_{g_i \circ f} \rangle_{\vec{z}_i}$ as small as possible, where $\vec{z}_i$ is the ith row of $\mathbf{z}$. The average empirical loss of the learner on the (n, m) sample $\mathbf{z}$ is then $E\left(\vec{g} \circ \bar{f}, \mathbf{z}\right)$ where this is defined as in equation (14), and the average true loss is $E(\vec{g} \circ \bar{f}, \vec{P})$ where this is defined as in equation (15). It may be that the n tasks learnt are all that is ever going to be required of the learner (for example the n tasks could include all the characters ever to be recognised, or all the words ever to be recognised, etc), i.e the n tasks entirely exhaust the environment of the learner, in which case it is not the performance of the representation f on future tasks that is of interest, but the performance of each $g_i \circ f$ on future examples of the same task. Good generalisation will be assured if the sample size is large enough to ensure with high probability that $E(\vec{g} \circ \bar{f}, \vec{P})$ and $E(\vec{g} \circ \bar{f}, \mathbf{z})$ are close. Following the treatment in chapter 2, given any $\nu > 0$ and $0 < \alpha < 1$, a bound on the probability that

$$d_\nu \left(E(\vec{g} \circ \bar{f}, \mathbf{z}), E(\vec{g} \circ \bar{f}, \vec{P}) \right) > \alpha$$

is required. Clearly the probability of this event is bounded by

$$\Pr \left\{ \mathbf{z} \in Z^{(n,m)} : \exists \vec{g} \circ \bar{f} \in \mathcal{G}^n \circ \overline{\mathcal{F}} : d_\nu \left(E(\vec{g} \circ \bar{f}, \mathbf{z}), E(\vec{g} \circ \bar{f}, \vec{P}) \right) > \alpha \right\}, \tag{17}$$

where the probability measure on $Z^{(n,m)}$ is the product measure $P_1^m \times \cdots \times P_n^m$ generated by $\vec{P} = (P_1, \ldots, P_n)$. With a little work and some more definitions, expression (17) may be converted into a form that can be bounded using essentially the same techniques used in the previous chapter.

If $Z^{(m,n)}$, the set of all $m \times n$ matrices over Z, is equipped with the measure $(P_1 \times \cdots \times P_n)^m$ (i.e. each *row* of a matrix is selected according to $P_1 \times \cdots \times P_n$), then the map $\psi : Z^{(n,m)} \to Z^{(m,n)}$, $\psi(\mathbf{z}) = \mathbf{z}^T$ where $\mathbf{z}^T$ is the transpose of $\mathbf{z}$ is clearly measure preserving if the measure on $Z^{(n,m)}$ is $P_1^m \times \cdots \times P_n^m$. Thus the sampling process on $Z^{(n,m)}$ according to $P_1^m \times \cdots \times P_n^m$ is equivalent to sampling from $Z^{(m,n)}$ according to $(P_1 \times \cdots \times P_n)^m$, which is the same as sampling m times from Z^n according to $P_1 \times \cdots \times P_n$. This observation, along with a suitable generalization of the loss function l to n dimensions, allows $E(\vec{g} \circ \bar{f}, \vec{P})$ and $E(\vec{g} \circ \bar{f}, \mathbf{z})$ to be interpreted

as expectations of loss functions, $l_{\vec{g}\circ\vec{f}}$, and hence turns the probability in equation (17) above into a statement about the probability of large deviation between true and empirical losses.

Given a loss function $l\colon Y \times A \to [0, M]$, define $l\colon Y^n \times A^n \to [0, M]$ by

$$l(\vec{y}, \vec{a}) = \frac{1}{n} \sum_{i=1}^{n} l(y_i, a_i).$$

By identifying $X^n \times Y^n$ with $(X \times Y)^n$ in the obvious way[7], and recalling that $Z = X \times Y$ so $Z^n = (X \times Y)^n \equiv X^n \times Y^n$ by this identification, for any function $h\colon X^n \to A^n$, a function $l_h : Z^n \to [0, M]$ may be defined as follows

$$l_h(\vec{z}) = l(\vec{y}, h(\vec{x})),$$

for all $\vec{z} \in Z^n, \vec{z} = (\vec{x}, \vec{y}), \vec{x} \in X^n, \vec{y} \in Y^n$. If $\mathcal{H}$ is a family of functions from X^n into A^n then denote by $l_{\mathcal{H}}$ the set $\{l_h : h \in \mathcal{H}\}$. Given a sequence of probability measures $\vec{P} = (P_1, \ldots, P_n)$, by abuse of notation denote the product measure $P_1 \times \cdots \times P_n$ on Z^n by $\vec{P}$ also, and define the *expectation with respect to* $\vec{P}$ of any function $f\colon Z^n \to [0, M]$ in the usual way,

$$\langle f \rangle_{\vec{P}} = \int_{Z^n} f(\vec{z})\, d\vec{P}(\vec{z}).$$

Define the expectation of $f\colon Z^n \to [0, M]$ with respect to the m-fold sample from Z^n, $\mathbf{z} = \{\vec{z}_1, \ldots, \vec{z}_m\} \equiv \begin{matrix} z_{11} & \cdots & z_{1n} \\ \vdots & \ddots & \vdots \\ z_{m1} & \cdots & z_{mn} \end{matrix} \in Z^{(m,n)}$ as

$$\langle f \rangle_{\mathbf{z}} = \frac{1}{m} \sum_{i=1}^{m} f\left(\vec{z}_i\right).$$

DEFINITION 3.1. For any probability measure P on Z^n, and any family of functions $\mathcal{H}\colon Z^n \to [0, M]$, a pseudo-metric d_P on $\mathcal{H}$ can be defined as in section 2.3,

$$d_P(h, h') = \int_{Z^n} |h(\vec{z}) - h'(\vec{z})|\, dP(\vec{z}),$$

for all $h, h' \in \mathcal{H}$. The ε-capacity of $\mathcal{H}$, $\mathcal{C}(\varepsilon, \mathcal{H})$, is then also defined as in §2.3, namely as the supremum of $\mathcal{N}(\varepsilon, \mathcal{H}, d_P)$ over all probability measures P such that all $h \in \mathcal{H}$ are P-measurable.

$$\mathcal{C}(\varepsilon, \mathcal{H}) = \sup_{P \in \mathcal{P}_{\mathcal{H}}} \mathcal{N}(\varepsilon, \mathcal{H}, d_P).$$

If $h\colon X^n \to A^n$ is expressible as a sequence $h = (h_1, \ldots, h_n)$, where $h_i\colon X \to A$ (that is, h does not mix the components of its argument), and $\hat{\mathbf{z}} \in Z^{(m,n)}$ then the expectation of l_h with respect to $\hat{\mathbf{z}}$ satisfies

$$\begin{aligned} \langle l_h \rangle_{\hat{\mathbf{z}}} &= \frac{1}{m} \sum_{i=1}^{m} l_h\left(\vec{\hat{z}}_i\right) \\ &= \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} l_{h_j}(\hat{z}_{ij}) \\ &= E(h, \hat{\mathbf{z}}^T) \end{aligned}$$

[7] $(x_1, \ldots, x_n, y_1, \ldots, y_n) \leftrightarrow (x_1, y_1, \ldots, x_n, y_n)$.

(recall equation (14), page 29). Also, for $\vec{P} = P_1 \times \cdots \times P_n$,

$$\begin{aligned}\langle l_h \rangle_{\vec{P}} &= \int_{Z^n} l_h(\vec{z})\, d\vec{P}(\vec{z}) \\ &= \int_{Z^n} \frac{1}{n} \sum_{i=1}^{n} l_{h_i}(z_i)\, dP_1(z_1) \ldots dP_n(z_n) \\ &= \frac{1}{n} \sum_{i=1}^{n} \int_Z l_{h_i}(z)\, dP_i(z) \\ &= E(h, \vec{P})\end{aligned}$$

(recall equation (15), page 29), where by abuse of notation $(P_1, \ldots, P_n)$ has been denoted by $\vec{P}$ also. Thus (17) can be rewritten as,

$$\begin{aligned}(18)\quad &\Pr\left\{\mathbf{z} \in Z^{(n,m)} \colon \exists \vec{g} \circ \bar{f} \in \mathcal{G}^n \circ \overline{\mathcal{F}} \colon d_\nu \left(E(\vec{g} \circ \bar{f}, \mathbf{z}), E(\vec{g} \circ \bar{f}, \vec{P}) \right) > \alpha \right\} \\ &\qquad = \Pr\left\{\mathbf{z} \in Z^{(m,n)} \colon \exists l_{\vec{g} \circ \bar{f}} \in l_{\mathcal{G}^n \circ \overline{\mathcal{F}}} \colon d_\nu \left(\langle l_{\vec{g} \circ \bar{f}} \rangle_{\mathbf{z}}, \langle l_{\vec{g} \circ \bar{f}} \rangle_{\vec{P}} \right) > \alpha \right\}.\end{aligned}$$

$l_{\mathcal{G}^n \circ \overline{\mathcal{F}}}$ is a subset of $l_{(\mathcal{G} \circ \mathcal{F})^n}$ which can in turn be written as $l_{\mathcal{G} \circ \mathcal{F}} \oplus \cdots \oplus l_{\mathcal{G} \circ \mathcal{F}}$ as defined in A.4. Therefore, assuming $l_{\mathcal{G}^n \circ \overline{\mathcal{F}}}$ is permissible[8] the conditions of theorem B.6 obtain for the probability in the right hand side of (18), hence,

$$\begin{aligned}&\Pr\left\{\mathbf{z} \in Z^{(m,n)} \colon \exists l_{\vec{g} \circ \bar{f}} \in l_{\mathcal{G}^n \circ \overline{\mathcal{F}}} \colon d_\nu \left(\langle l_{\vec{g} \circ \bar{f}} \rangle_{\mathbf{z}}, \langle l_{\vec{g} \circ \bar{f}} \rangle_{\vec{P}} \right) > \alpha \right\} \\ &\qquad\qquad \leq 4\mathcal{C}\left(\alpha\nu/8, l_{\mathcal{G}^n \circ \overline{\mathcal{F}}}\right) e^{-\frac{\alpha^2 \nu n m}{8M}}.\end{aligned}$$

By theorem C.10,

$$\mathcal{C}\left(\alpha\nu/8, l_{\mathcal{G}^n \circ \overline{\mathcal{F}}}\right) \leq \mathcal{C}\left(\varepsilon_1, l_{\mathcal{G}}\right)^n \mathcal{C}^*_{l_{\mathcal{G}}}\left(\varepsilon_2, \mathcal{F}\right),$$

where $\varepsilon_1 + \varepsilon_2 = \frac{\alpha\nu}{8}$ and the capacities in the right hand side are defined as in definitions B.4 and C.8. Thus, if the learner samples

$$(19)\qquad m > \frac{8M}{\alpha^2 \nu}\left[\ln \mathcal{C}\left(\varepsilon_1, l_{\mathcal{G}}\right) + \frac{1}{n} \ln \frac{4\mathcal{C}^*_{l_{\mathcal{G}}}\left(\varepsilon_2, \mathcal{F}\right)}{\delta}\right]$$

times from Z according to each distribution P_i, $1 \leq i \leq n$, then with probability at least $1 - \delta$ the d_ν distance between the true and empirical loss of the learner will be less than α. Ignoring the accuracy parameters α, ν and δ,

$$m = O\left(\log \mathcal{C}\left(\varepsilon_1, l_{\mathcal{G}}\right) + \frac{1}{n} \log \mathcal{C}^*_{l_{\mathcal{G}}}\left(\varepsilon_2, \mathcal{F}\right)\right),$$

which is the rigorous version of the order of magnitude estimate (13) on page 26.

Exactly the same argument with $n = 1$ yields the number of samples required for learning a single task P using the hypothesis space $\mathcal{G} \circ \mathcal{F}$,

$$m > \frac{8M}{\alpha^2 \nu}\left[\ln \mathcal{C}\left(\varepsilon_1, l_{\mathcal{G}}\right) + \ln \frac{4\mathcal{C}^*_{l_{\mathcal{G}}}\left(\varepsilon_2, \mathcal{F}\right)}{\delta}\right].$$

[8] The conditions ensuring permissibility of $l_{\mathcal{G}^n \circ \overline{\mathcal{F}}}$ are given in appendix A. However the motivation for the definitions given there and their relationship to the current framework is best understood after reading section 3.6 of the current chapter.

This gives an order of magnitude estimate,

$$m = O\left(\log \mathcal{C}\left(\varepsilon_1, l_{\mathcal{G}}\right) + \log \mathcal{C}^*_{l_{\mathcal{G}}}\left(\varepsilon_2, \mathcal{F}\right)\right),$$

which is the rigorous version of (12) on page 26.

Thus if $\mathcal{C}^*_{l_{\mathcal{G}}}\left(\varepsilon_2, \mathcal{F}\right) \gg \mathcal{C}\left(\varepsilon_1, l_{\mathcal{G}}\right)$ then there is a virtually linear (in n) reduction in the number of examples per task required for representation learning as compared with ordinary learning of a single task.

In appendix D, using the same methods as Haussler (1992), it is shown for neural network $\mathcal{F}$ and $\mathcal{G}$ that the logarithm of the capacities $\mathcal{C}^*_{l_{\mathcal{G}}}(\varepsilon_2, \mathcal{F})$ and $\mathcal{C}(\varepsilon_1, l_{\mathcal{G}})$ are proportional to the number of weights in each network. Hence the advantage gained by representation learning using neural networks is essentially proportional to the ratio of the size of the networks in $\mathcal{F}$ and the networks in $\mathcal{G}$.

In the next section, sufficient sampling conditions are derived such that a representation with small empirical loss (10) on an (n, m) sample also has a small true loss (11), i.e. it is a good representation for learning future tasks drawn from the environment.

3.5.4. Generalisation on all tasks. In representation learning, given a sample $\mathbf{z} \in Z^{(n,m)}$, the learner attempts to produce a representation $f \in \mathcal{F}$ with as small an empirical loss (10) on the sample $\mathbf{z}$ as possible. The empirical loss is an estimate of how well the learner can be expected to learn using f. To be confident that the empirical loss is a good estimate of the true loss (11), the probability

$$\Pr\left\{\mathbf{z} \in Z^{(n,m)} \colon d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}), \mathbf{z}), E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}), Q)\right) > \alpha\right\} \tag{20}$$

must be bounded, where the samples $\mathbf{z}$ are generated according to the (n, m) sampling process[9]. Note that with the (n, m) sampling process, in addition to the sample $\mathbf{z}$ there is also generated implicitly a sequence of probability measures, $\vec{P} = (P_1, \ldots, P_n)$ although these are not supplied to the learner. Hence the (n, m) sampling process can be thought of as a sampling process on $Z^{(n,m)} \times \mathcal{P}^n$. This notion is used in the following lemma, the proof of which follows directly from the triangle inequality for d_ν.

LEMMA 3.1. *If*

$$\Pr\left\{(\mathbf{z}, \vec{P}) \in Z^{(n,m)} \times \mathcal{P}^n \colon d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}), \mathbf{z}), E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}), \vec{P})\right) > \frac{\alpha}{2}\right\} \leq \frac{\delta}{2}, \tag{21}$$

and

$$\Pr\left\{(\mathbf{z}, \vec{P}) \in Z^{(n,m)} \times \mathcal{P}^n \colon d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}), \vec{P}), E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}), Q)\right) > \frac{\alpha}{2}\right\} \leq \frac{\delta}{2}, \tag{22}$$

then

$$\Pr\left\{\mathbf{z} \in Z^{(n,m)} \colon d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}), \mathbf{z}), E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}), Q)\right) > \alpha\right\} \leq \delta.$$

Thus in order to determine when (20) holds, the conditions under which (21) and (22) hold must be found. The two inequalities are treated separately.

[9] The full expression for this probability is:

$$\int_{\mathcal{P}^n} \int_{Z^{(n,m)}} \theta\left(d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}), \mathbf{z}), E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}), Q)\right) - \alpha\right) dP_1^m \times \cdots \times dP_n^m(\mathbf{z})\, dQ^n\left(P_1, \ldots, P_n\right),$$

where θ is the Heaviside step function.

The First Inequality.

LEMMA 3.2.

$$\begin{aligned}(23)\quad &\Pr\left\{(\mathbf{z},\vec{P})\in Z^{(n,m)}\times\mathcal{P}^n\colon d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\mathbf{z}),E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\vec{P})\right)>\alpha\right\}\\ &\le \Pr\left\{(\mathbf{z},\vec{P})\in Z^{(n,m)}\times\mathcal{P}^n\colon \exists\vec{g}\circ\bar{f}\in\mathcal{G}^n\circ\overline{\mathcal{F}}\colon d_\nu\left(E(\vec{g}\circ\bar{f},\mathbf{z}),E(\vec{g}\circ\bar{f},\vec{P})\right)>\alpha\right\}\end{aligned}$$

PROOF. Suppose that $(\mathbf{z},\vec{P})\in Z^{(n,m)}\times\mathcal{P}^n$ are such that $d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\mathbf{z}),E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\vec{P})\right)>\alpha$. To simplify things, denote $\mathcal{A}(\mathbf{z})$ by f. Suppose first that $E^*_{\mathcal{G}}(f,\mathbf{z})\le E^*_{\mathcal{G}}(f,\vec{P})$. By definition of the infimum, for all $\varepsilon>0$, $\exists\vec{g}\in\mathcal{G}^n$ such that

$$E(\vec{g}\circ\bar{f},\mathbf{z})<E^*_{\mathcal{G}}(f,\mathbf{z})+\varepsilon.$$

Hence by property (3), lemma B.1 of the d_ν metric, for all $\varepsilon>0$, $\exists\vec{g}\in\mathcal{G}^n$ such that

$$d_\nu\left(E(\vec{g}\circ\bar{f},\mathbf{z}),E^*_{\mathcal{G}}(f,\mathbf{z})\right)<\varepsilon.$$

Let $\vec{g}\in\mathcal{G}^n$ satisfy this inequality. By definition,

$$E^*_{\mathcal{G}}(f,\vec{P})\le E(\vec{g}\circ\bar{f},\vec{P}).$$

Hence

$$E^*_{\mathcal{G}}(f,\mathbf{z})\le E^*_{\mathcal{G}}(f,\vec{P})\le E(\vec{g}\circ\bar{f},\vec{P}),$$

and as $d_\nu\left(E^*_{\mathcal{G}}(f,\mathbf{z}),E^*_{\mathcal{G}}(f,\vec{P})\right)>\alpha$, by the compatibility of d_ν with the ordering on the reals (see lemma B.1),

$$d_\nu\left(E^*_{\mathcal{G}}(f,\mathbf{z}),E(\vec{g}\circ\bar{f},\vec{P})\right)>\alpha=\alpha+\delta,\ \text{say.}$$

By the triangle inequality for d_ν,

$$\begin{aligned}d_\nu\left(E(\vec{g}\circ\bar{f},\mathbf{z}),E(\vec{g}\circ\bar{f},\vec{P})\right)+d_\nu\left(E(\vec{g}\circ\bar{f},\mathbf{z}),E^*_{\mathcal{G}}(f,\mathbf{z})\right)&\\ \ge d_\nu\left(E^*_{\mathcal{G}}(f,\mathbf{z}),E(\vec{g}\circ\bar{f},\vec{P})\right)=\alpha+\delta.&\end{aligned}$$

Thus,

$$d_\nu\left(E(\vec{g}\circ\bar{f},\mathbf{z}),E(\vec{g}\circ\bar{f},\vec{P})\right)>\alpha+\delta-\varepsilon,$$

and for any $\varepsilon>0$ a $\vec{g}$ satisfying this inequality can be found. Choosing $\varepsilon=\delta$ shows that there exists $\vec{g}\in\mathcal{G}^n$ such that

$$d_\nu\left(E(\vec{g}\circ\bar{f},\mathbf{z}),E(\vec{g}\circ\bar{f},\vec{P})\right)>\alpha.$$

If, instead, $E^*_{\mathcal{G}}(f,\vec{P})<E^*_{\mathcal{G}}(f,\mathbf{z})$, then an identical argument can be run with the role of $\mathbf{z}$ and $\vec{P}$ interchanged to show that there exists $\vec{g}\in\mathcal{G}^n$ such that

$$d_\nu\left(E(\vec{g}\circ\bar{f},\vec{P}),E(\vec{g}\circ\bar{f},\mathbf{z})\right)>\alpha.$$

Thus in both cases,

$$\begin{aligned}d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\mathbf{z}),E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\vec{P})\right)>\alpha&\\ \Rightarrow\exists\vec{g}\circ\bar{f}\in\mathcal{G}^n\circ\overline{\mathcal{F}}\colon d_\nu\left(E(\vec{g}\circ\bar{f},\mathbf{z}),E(\vec{g}\circ\bar{f},\vec{P})\right)>\alpha,&\end{aligned}$$

and so

$$\begin{aligned}&\Pr\left\{(\mathbf{z},\vec{P})\in Z^{(n,m)}\times\mathcal{P}^n : d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\mathbf{z}),E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\vec{P})\right)>\alpha\right\}\\ &\le \Pr\left\{(\mathbf{z},\vec{P})\in Z^{(n,m)}\times\mathcal{P}^n : \exists \vec{g}\circ\bar{f}\in\mathcal{G}^n\circ\overline{\mathcal{F}}\colon d_\nu\left(E(\vec{g}\circ\bar{f},\mathbf{z}),E(\vec{g}\circ\bar{f},\vec{P})\right)>\alpha\right\},\end{aligned}$$

as required. □

By the nature of the (n,m) sampling process,

(24)

$$\begin{aligned}&\Pr\left\{(\mathbf{z},\vec{P})\in Z^{(n,m)}\times\mathcal{P}^n : \exists \vec{g}\circ\bar{f}\in\mathcal{G}^n\circ\overline{\mathcal{F}}\colon d_\nu\left(E(\vec{g}\circ\bar{f},\mathbf{z}),E(\vec{g}\circ\bar{f},\vec{P})\right)>\alpha\right\}\\ &=\int\limits_{\vec{P}\in\mathcal{P}^n}\Pr\left\{\mathbf{z}\in Z^{(n,m)}\colon \exists \vec{g}\circ\bar{f}\in\mathcal{G}^n\circ\overline{\mathcal{F}}\colon d_\nu\left(E(\vec{g}\circ\bar{f},\mathbf{z}),E(\vec{g}\circ\bar{f},\vec{P})\right)>\alpha\right\}dQ^n(\vec{P}),\end{aligned}$$

where the probability measure on $Z^{(n,m)}$ used in the integrand on the right hand side is the product measure $P_1^m\times\cdots\times P_n^m$ generated by $\vec{P}=(P_1,\ldots,P_n)\in\mathcal{P}^n$. The probability in the integrand has already been bounded in §3.5.3, thus replacing α and δ by $\alpha/2$ and $\delta/2$ in formula (19) from that section gives the following lemma ensuring the first inequality (21) is satisfied.

LEMMA 3.3. *If*

$$m\ge\frac{32M}{\alpha^2\nu}\left[\ln\mathcal{C}\left(\varepsilon_1,l_{\mathcal{G}}\right)+\frac{1}{n}\ln\frac{8\mathcal{C}^*_{l_{\mathcal{G}}}\left(\varepsilon_2,\mathcal{F}\right)}{\delta}\right],$$

where $\varepsilon_1+\varepsilon_2=\frac{\alpha\nu}{16}$, *then for any representation learner*

$$\mathcal{A}\colon\bigcup_{\substack{n\ge1\\m\ge1}}Z^{(n,m)}\to\mathcal{F},$$

$$\Pr\left\{(\mathbf{z},\vec{P})\in Z^{(n,m)}\times\mathcal{P}^n : d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\mathbf{z}),E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\vec{P})\right)>\frac{\alpha}{2}\right\}\le\frac{\delta}{2}.$$

The Second Inequality. To establish a bound for the probability in the second inequality,

$$\Pr\left\{(\mathbf{z},\vec{P})\in Z^{(n,m)}\times\mathcal{P}^n : d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\vec{P}),E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),Q)\right)>\varepsilon\right\},\tag{25}$$

first note that

$$\begin{aligned}&\Pr\left\{(\mathbf{z},\vec{P})\in Z^{(n,m)}\times\mathcal{P}^n : d_\nu\left(E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),\vec{P}),E^*_{\mathcal{G}}(\mathcal{A}(\mathbf{z}),Q)\right)>\varepsilon\right\}\\ &\qquad\le\Pr\left\{\vec{P}\in\mathcal{P}^n : \exists f\in\mathcal{F}\colon d_\nu\left(E^*_{\mathcal{G}}(f,\vec{P}),E^*_{\mathcal{G}}(f,Q)\right)>\varepsilon\right\}.\end{aligned}\tag{26}$$

Recall definitions (16) and (11) for $E^*_{\mathcal{G}}(f,\vec{P})$ and $E^*_{\mathcal{G}}(f,Q)$, where $\vec{P}=(P_1,\ldots,P_n)$:

$$E^*_{\mathcal{G}}(f,\vec{P})=\frac{1}{n}\sum_{i=1}^n\inf_{g\in\mathcal{G}}\langle l_{g\circ f}\rangle_{P_i},$$

$$E^*_{\mathcal{G}}(f,Q)=\int_{\mathcal{P}}\inf_{g\in\mathcal{G}}\langle l_{g\circ f}\rangle_P\,dQ(P).$$

For each $g \circ f \in \mathcal{G} \circ \mathcal{F}$, define a function $\bar{l}_{g \circ f} : \mathcal{P} \to [0, M]$ by

$$\bar{l}_{g \circ f}(P) = \langle l_{g \circ f} \rangle_P$$

and let $\bar{l}_{\mathcal{G} \circ \mathcal{F}}$ denote the set of all such functions. For each $f \in \mathcal{F}$, define a function $l^*_f : \mathcal{P} \to [0, M]$ by

$$l^*_f(P) = \inf_{g \in \mathcal{G}} \bar{l}_{g \circ f}(P)$$

and let $l^*_{\mathcal{F}}$ denote the set of all such functions.

It is now possible to introduce the minimal σ-algebra on $\mathcal{P}$ ensuring all things are measurable. Recalling definition A.1 from appendix A, $\sigma_{\bar{l}_{\mathcal{G} \circ \mathcal{F}}}$ is the correct σ-algebra to use on $\mathcal{P}$, and in fact $\mathcal{P}$ should be the set of probability measures $\mathcal{P}_{l_{\mathcal{G} \circ \mathcal{F}}}$. Furthermore, the environmental measure Q should be a member of $\mathcal{P}_{\bar{l}_{\mathcal{G} \circ \mathcal{F}}}$ and the σ-algebra on Z should be $\sigma_{l_{\mathcal{G} \circ \mathcal{F}}}$. This makes everything measurable that needs to be, including all $l^*_f \in l^*_{\mathcal{F}}$ (see lemma A.2 in appendix A, although the notation there will be more easily understood after reading section 3.6 of the present chapter). Unless otherwise stated it will be assumed from now on that Q, $\mathcal{P}$, etc. all belong to the appropriate spaces.

The expectation of l^*_f with respect to $\vec{P} = (P_1, \ldots, P_n)$ is

$$\begin{aligned} \langle l^*_f \rangle_{\vec{P}} &= \frac{1}{n} \sum_{i=1}^{n} l^*_f(P_i) \\ &= \frac{1}{n} \sum_{i=1}^{n} \inf_{g \in \mathcal{G}} \langle l_{g \circ f} \rangle_{P_i} \\ &= E^*_{\mathcal{G}}(f, \vec{P}). \end{aligned}$$

Similarly,

$$\begin{aligned} \langle l^*_f \rangle_Q &= \int_{\mathcal{P}} l^*_f(P)\, dQ(P) \\ &= \int_{\mathcal{P}} \inf_{g \in \mathcal{G}} \langle l_{g \circ f} \rangle_P dQ(P) \\ &= E^*_{\mathcal{G}}(f, Q). \end{aligned}$$

Thus,

$$\begin{aligned} (27) \quad & \Pr\left\{ \vec{P} \in \mathcal{P}^n : \exists f \in \mathcal{F} : d_\nu \left(E^*_{\mathcal{G}}(f, \vec{P}), E^*_{\mathcal{G}}(f, Q) \right) > \varepsilon \right\} \\ & \qquad = \Pr\left\{ \vec{P} \in \mathcal{P}^n : \exists l^*_f \in l^*_{\mathcal{F}} : d_\nu \left(\langle l^*_f \rangle_{\vec{P}}, \langle l^*_f \rangle_Q \right) > \varepsilon \right\}. \end{aligned}$$

Definition 3.2. For any probabiltiy measure $Q \in \mathcal{P}_{\bar{l}_{\mathcal{G} \circ \mathcal{F}}}$ on $\mathcal{P}_{l_{\mathcal{G} \circ \mathcal{F}}}$, define the pseudo-metric d_Q on $l^*_{\mathcal{F}}$ by

$$d_Q \left(l^*_f, l^*_{f'} \right) = \int_{\mathcal{P}} |l^*_f(P) - l^*_{f'}(P)|\, dQ(P),$$

and define the ε-capacity of $l^*_{\mathcal{F}}$ by

$$\mathcal{C}(\varepsilon, l^*_{\mathcal{F}}) = \sup_{Q \in \mathcal{P}_{\bar{l}_{\mathcal{G} \circ \mathcal{F}}}} \mathcal{N}(\varepsilon, l^*_{\mathcal{F}}, d_Q),$$

with $\mathcal{C}(\varepsilon, l^*_{\mathcal{F}}) = \infty$ if the supremum is undefined.

With these definitions the following theorem is obtained immediately from corollary B.7.

THEOREM 3.4. *Let $l^*_{\mathcal{F}}$ be a permissible*[10] *family of maps from $\mathcal{P}$ into $[0, M]$ and let Q be a probability measure on $\mathcal{P}$. Let $\vec{P}$ be generated by n independent trials from $\mathcal{P}$ according to Q. Then for all $\nu > 0$, $0 < \alpha < 1$,*

$$\Pr\left\{\vec{P} \in \mathcal{P}^n : \exists l^*_f \in l^*_{\mathcal{F}} : d_\nu\left(\langle l^*_f\rangle_{\vec{P}}, \langle l^*_f\rangle_Q\right) > \alpha\right\} \leq 4\mathcal{C}(\alpha\nu/8, l^*_{\mathcal{F}})e^{-\frac{\alpha^2\nu n}{8M}}.$$

With one more definition, the capacity of $l^*_{\mathcal{F}}$ can be bounded in terms of the capacity of $\mathcal{F}$.

DEFINITION 3.3. For any $Q \in \mathcal{P}^{-}_{l_{\mathcal{G}}\circ\mathcal{F}}$, define the probability measure Q_Z on Z by

$$Q_Z(S) = \int_{\mathcal{P}} P(S)\, dQ(P),$$

for all S in the σ-algebra $\sigma_{l_{\mathcal{G}}\circ\mathcal{F}}$ on Z.

Recalling definition C.8 for the metric $d^*_{[P,l_{\mathcal{G}}]}$ on $\mathcal{F}$ gives the following lemma:

LEMMA 3.5. *For all $l^*_f, l^*_{f'} \in l^*_{\mathcal{F}}$,*

$$d_Q\left(l^*_f, l^*_{f'}\right) \leq d^*_{[Q_Z, l_{\mathcal{G}}]}(f, f').$$

PROOF. Note that if h, h' are bounded, positive functions on an arbitrary set X, then

$$\left|\inf_{x\in X} h(x) - \inf_{x\in X} h'(x)\right| \leq \sup_{x\in X} |h(x) - h'(x)|. \tag{28}$$

Now, for all $l^*_f, l^*_{f'} \in l^*_{\mathcal{F}}$,

$$\begin{aligned}
d_Q\left(l^*_f, l^*_{f'}\right) &= \int_{\mathcal{P}} \left|l^*_f(P) - l^*_{f'}(P)\right| dQ(P) \\
&= \int_{\mathcal{P}} \left|\inf_{g\in\mathcal{G}} \int_Z l_{g\circ f}(z)\, dP(z) - \inf_{g\in\mathcal{G}} \int_Z l_{g\circ f'}(z)\, dP(z)\right| dQ(P) \\
&\leq \int_{\mathcal{P}} \sup_{g\in\mathcal{G}} \left|\int_Z l_{g\circ f}(z) - l_{g\circ f'}(z)\, dP(z)\right| dQ(P) \qquad \text{(by (28) above)} \\
&\leq \int_{\mathcal{P}} \sup_{g\in\mathcal{G}} \int_Z |l_{g\circ f}(z) - l_{g\circ f'}(z)|\, dP(z)\, dQ(P) \\
&\leq \int_{\mathcal{P}} \int_Z \sup_{g\in\mathcal{G}} |l_{g\circ f}(z) - l_{g\circ f'}(z)|\, dP(z)\, dQ(P) \\
&= \int_Z \sup_{g\in\mathcal{G}} |l_{g\circ f}(z) - l_{g\circ f'}(z)|\, dQ_Z(z) \\
&= d^*_{[Q_Z, l_{\mathcal{G}}]}(f, f').
\end{aligned}$$

□

[10] See section 3.6 of the present chapter for the conditions ensuring permissibility of $l^*_{\mathcal{F}}$.

Thus the surjective map $\psi\colon \left(\mathcal{F}, d^*_{[Q_Z, l_\mathcal{G}]}\right) \to (l^*_\mathcal{F}, d_Q)$ where $\psi(f) = l^*_f$ is a contraction and so by lemma B.4,

$$\mathcal{N}\left(\varepsilon, l^*_\mathcal{F}, d_Q\right) \leq \mathcal{N}\left(\varepsilon, \mathcal{F}, d^*_{[Q_Z, l_\mathcal{G}]}\right).$$

By lemma A.2, $\mathcal{P}_{l_{\mathcal{G}\circ\mathcal{F}}}$ is guaranteed to contain the measure Q_Z for every $Q \in \mathcal{P}_{\bar{l}_{\mathcal{G}\circ\mathcal{F}}}$, and so

LEMMA 3.6.

$$\mathcal{C}\left(\varepsilon, l^*_\mathcal{F}\right) \leq \mathcal{C}^*_{l_\mathcal{G}}\left(\varepsilon, \mathcal{F}\right).$$

The probability in the second inequality can now be bounded.

LEMMA 3.7. *For all $\nu > 0$, $0 < \alpha < 1$ and $0 < \delta < 1$, assuming $l^*_\mathcal{F}$ is permissible and*

$$n \geq \frac{32M}{\alpha^2\nu} \ln\left(\frac{8\mathcal{C}^*_{l_\mathcal{G}}\left(\frac{\alpha\nu}{16}, \mathcal{F}\right)}{\delta}\right),$$

then for any representation learner,

$$\mathcal{A}\colon \bigcup_{\substack{n\geq 1\\ m\geq 1}} Z^{(n,m)} \to \mathcal{F},$$

$$\Pr\left\{(\mathbf{z}, \vec{P}) \in Z^{(n,m)} \times \mathcal{P}^n : d_\nu\left(E^*_\mathcal{G}(\mathcal{A}(\mathbf{z}), \vec{P}), E^*_\mathcal{G}(\mathcal{A}(\mathbf{z}), Q)\right) > \frac{\alpha}{2}\right\} \leq \frac{\delta}{2}.$$

Main Bound. Putting lemmas 3.7, 3.3 and 3.1 together gives the following theorem.

THEOREM 3.8. *Let $\mathcal{F}, \mathcal{G}$ be families of functions with the structure $X \xrightarrow{\mathcal{F}} V \xrightarrow{\mathcal{G}} A$, and let l be a loss function $l\colon Y \times A \to [0, M]$. Assume $\mathcal{G}\circ\mathcal{F}$ is such that $l_{\mathcal{G}^n\circ\overline{\mathcal{F}}}$ and $l^*_\mathcal{F}$ are permissible*[11]*. Let $\mathbf{z} \in Z^{(n,m)}$ be an (n, m) sample from Z according to an environmental measure Q. For all $0 < \alpha, \delta, \varepsilon_1, \varepsilon_2 < 1$, $\nu > 0$, $\varepsilon_1 + \varepsilon_2 = \frac{\alpha\nu}{16}$, if*

$$n \geq \frac{32M}{\alpha^2\nu} \ln\frac{8\mathcal{C}^*_{l_\mathcal{G}}\left(\frac{\alpha\nu}{16}, \mathcal{F}\right)}{\delta},$$

$$\text{and} \quad m \geq \frac{32M}{\alpha^2\nu}\left[\ln \mathcal{C}\left(\varepsilon_1, \mathcal{G}\right) + \frac{1}{n}\ln\frac{8\mathcal{C}^*_{l_\mathcal{G}}\left(\varepsilon_2, \mathcal{F}\right)}{\delta}\right],$$

and $\mathcal{A}$ is any representation learner,

$$\mathcal{A}\colon \bigcup_{\substack{n\geq 1\\ m\geq 1}} Z^{(n,m)} \to \mathcal{F},$$

then

$$\Pr\left\{\mathbf{z} \in Z^{(n,m)} : d_\nu\left(E^*_\mathcal{G}(\mathcal{A}(\mathbf{z}), \mathbf{z}), E^*_\mathcal{G}(A(\mathbf{z}), Q)\right) > \alpha\right\} \leq \delta.$$

[11] Both these conditions can be subsumed within one criterion known as *f-permissibility*, see section 3.6 and appendix A.

3.5.5. Discussion. Theorem 3.8 shows, unsurprisingly, that both n and m need to be sufficiently large to ensure with high probability good generalisation from a representation. The interaction between the two bounds is best understood by assuming that $\mathcal{F}$ and $\mathcal{G}$ are neural networks and applying the bounds for the capacities derived in appendix D. Setting the total number of weights in $\mathcal{F}$ to be $W_{\mathcal{F}}$, and similarly $W_{\mathcal{G}}$ for $\mathcal{G}$, and looking only at the dependence on $\alpha, \nu, \delta, W_{\mathcal{F}}$ and $W_{\mathcal{G}}$ of the above bounds, gives[12]

$$n = \tilde{O}\left(\frac{W_{\mathcal{F}}}{\alpha^2 \nu}\right) \tag{29}$$

$$m = \tilde{O}\left(\frac{1}{\alpha^2 \nu}\left[W_{\mathcal{G}} + \frac{W_{\mathcal{F}}}{n}\right]\right). \tag{30}$$

The bound on m for ordinary learning would read in this case,

$$m = \tilde{O}\left(\frac{1}{\alpha^2 \nu}\left[W_{\mathcal{G}} + W_{\mathcal{F}}\right]\right). \tag{31}$$

If the representation is not to be used for learning future tasks from the same environment then the bound (29) on n can be ignored, and then a comparison between (30) and (31) clearly demonstrates the superiority of representation learning over ordinary learning, at least on a number of examples required per task basis and in the case that $W_{\mathcal{F}} \gg W_{\mathcal{G}}$. However, if the representation is to be used for learning future tasks from the same environment, then the bound on n must be taken seriously, leading to the following expression for the *total* number of examples mn required for good generalisation,

$$mn = \tilde{O}\left(\frac{W_{\mathcal{F}} W_{\mathcal{G}}}{\alpha^4 \nu^2}\right).$$

On face value, the fact that this quantity is far greater than that required for ordinary learning might prompt the conclusion that representation learning is infeasible. This is not the case, as (1): representation learning is not intended to be performed *on-line* as is often true of ordinary learning and (2): for representation learning it is *assumed* that there are many tasks from which to collect examples so the increased burden may be spread out. Furthermore, once a successful representation f has been learnt, the hypothesis space used by the learner will be $\mathcal{G} \circ f$, not $\mathcal{G} \circ \mathcal{F}$. Applying lemma C.4 and using the fact that the capacity of a space consisting of a single function is one, gives $\mathcal{C}(\varepsilon, l_{\mathcal{G} \circ f}) = \mathcal{C}(\varepsilon, l_{\mathcal{G}})$. This gives an order of magnitude estimate for the number of examples required for learning using a representation of

$$m = \tilde{O}\left(\frac{W_{\mathcal{G}}}{\alpha^2 \nu}\right). \tag{32}$$

Comparing (32) with (31) shows that if $W_{\mathcal{F}} \gg W_{\mathcal{G}}$, all the extra work involved in learning the representation is rewarded when it comes time to learn new tasks using the representation.

[12] The $\tilde{O}$ notation indicates that logarithmic factors have been suppressed in the same way that constant factors are suppressed with the O notation. It is borrowed from Haussler et. al. (1994).

3.6. General Framework

Learning a representation is just one example of hypothesis space bias. Other examples from artificial neural network research include biasing by limiting the number of hidden nodes and layers posessed by the networks and limiting the number of non-zero weights or the size of the weights. In this section, the model of hypothesis space bias used in the previous section is generalised to cover such learning scenarios, and in fact any learning scenario in which the learner receives an (n, m) sample from the environment and uses it to bias the hypothesis space. Bounds on the (n, m)-sample size required for PC learning will be derived and expressed, as before, in terms of the capacity of certain classes of functions.

3.6.1. Preliminaries. In general, a learner that receives (n, m) samples $\mathbf{z} \in Z^{(n,m)}$ and biases its hypothesis space in some way, must effectively have a *family* of hypothesis spaces from which to choose. Let $H = \{\mathcal{H}\}$ denote this family. Given an (n, m)-sample $\mathbf{z} = (\vec{z}_1, \ldots, \vec{z}_n)$, drawn according to some probability measure Q on the space of probability measures $\mathcal{P}$ on Z, and a loss function $l\colon Y \times A \to [0, M]$, the learner searches for a hypothesis space $\mathcal{H} \in H$ with minimal *empirical loss* on $\mathbf{z}$, where this is denoted by $E^*(\mathcal{H}, \mathbf{z})$ and defined by

$$E^*(\mathcal{H}, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} \inf_{h \in \mathcal{H}} \langle l_h \rangle_{\vec{z}_i}. \tag{33}$$

$E^*(\mathcal{H}, \mathbf{z})$ is an estimate of the *true loss* of $\mathcal{H}$,

$$E^*(\mathcal{H}, Q) = \int_{\mathcal{P}} \inf_{h \in \mathcal{H}} \langle l_h \rangle_P \, dQ(P). \tag{34}$$

The true loss of a hypothesis space $\mathcal{H}$ is a measure of how effective $\mathcal{H}$ is for learning within the environment Q. To see that representation learning is a special case of this framework, let $\mathcal{F}$ be the representation space and $\mathcal{G}$ the output function space of the learner, and let $H = \{\mathcal{G} \circ f : f \in \mathcal{F}\}$. The empirical and true loss of a hypothesis space $\mathcal{G} \circ f \in H$ is the same as the empirical and true loss of the corresponding representation f, as defined in equations (10) and (11) in section 3.1.

In determining the empirical loss of a hypothesis space $\mathcal{H}$, the learner searches for a sequence of n functions $\vec{h} = (h_1, \ldots, h_n), h_i \in \mathcal{H}$ with minimal empirical loss on the sample $\mathbf{z}$, where this is defined as in (14) and reproduced below,

$$E(\vec{h}, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} \langle l_{h_i} \rangle_{\vec{z}_i}.$$

If the learner is biasing the hypothesis space with the intention of using it to learn future tasks from the same environment, then, as with representation learning, it needs to know how large to make n and m to ensure with high probability that (33) and (34) are close. However, as discussed in section 3.5.3, the (n, m) sample $\mathbf{z}$ may contain examples of all the tasks the learner is ever going to be asked to learn, in which case the learner will only ever be using the sequence $\vec{h} = (h_1, \ldots, h_n)$ and will primarily be interested in bounding the deviation between (14) and the true loss of $\vec{h}$, where this is defined as in (15) and reproduced below,

$$E(\vec{h}, \vec{P}) = \frac{1}{n} \sum_{i=1}^{n} \langle l_{h_i} \rangle_{P_i},$$

where $\vec{P} = (P_1, \ldots, P_n)$ are the probability measures used to generate $\mathbf{z}$. Both kinds of deviation can be bounded with simple extensions to the theory developed in previous sections.

3.6.2. Generalisation on n tasks. To begin with, for any family of hypothesis spaces H, define

$$H_\sigma = \{h \in \mathcal{H} : \mathcal{H} \in H\},$$

and define

$$H^n = \{\mathcal{H}^n : \mathcal{H} \in H\},$$

where $\mathcal{H}^n = \{(h_1, \ldots, h_n) : h_i \in \mathcal{H}, 1 \leq i \leq n \text{ and } (h_1, \ldots, h_n)(x_1, \ldots, x_n) = (h_1(x_1), \ldots, h_n(x_n)) \text{ for all } (x_1, \ldots, x_n) \in X^n\}$. It will be assumed that H has been constructed so that H_σ represents the completely unbiased hypothesis space the learner would have to use if it had no extra information about the kinds of learning problems to be encountered.

The operations σ and raising to the power of n clearly do not commute. In fact,

$$[H^n]_\sigma \subseteq [H_\sigma]^n \tag{35}$$

always. For example, in representation learning, with $H = \{\mathcal{G} \circ f : f \in \mathcal{F}\}$ as above, $[H^n]_\sigma = \mathcal{G}^n \circ \overline{\mathcal{F}}$ while $[H_\sigma]^n = (\mathcal{G} \circ \mathcal{F})^n = \mathcal{G}^n \circ \mathcal{F}^n$, and the two are equal only in the trivial case that $\mathcal{F}$ contains a single function. The extent of this noncommutativity controls the advantage that can be achieved through learning (n, m) samples as opposed to ordinary (m) samples. Recalling equations (12) and (13) in section 3.3, it can be seen that, depending on the relative capacities of $\mathcal{G}$ and $\mathcal{F}$, anything from no reduction at all to an n-fold reduction in the number of examples per function required for PC learning can be achieved through learning an (n, m) sample as opposed to an ordinary sample. No reduction essentially corresponds to equality in expression (35) wherea maximum reduction occurs when the two sets are maximally different. In fact the range exhibited by representation learning is prototypical of all (n, m)-sample learning, in that an n-fold reduction is the maximum possible for *any* learner, as will be demonstrated shortly.

A learner that receives an (n, m) sample $\mathbf{z}$ and produces n hypotheses $(h_1, \ldots, h_n)$ in response, where all the h_i belong to the same hypothesis space $\mathcal{H} \in H$, is effectively a map $\mathcal{A}$ from the space of all (n, m) samples into $[H^n]_\sigma$,

$$\mathcal{A} : \bigcup_{\substack{n \geq 1 \\ m \geq 1}} Z^{(n,m)} \to [H^n]_\sigma ,$$

although, as usual the results derived here apply equally well to a stochastic learner.

Given a loss function $l : Y \times A \to [0, M]$ and a hypothesis space family H consisting of hypothesis spaces whose elements are all functions mapping $X \to A$, define $l_H = \{l_\mathcal{H} : \mathcal{H} \in H\}$ (recall that $l_\mathcal{H} = \{l_h : h \in \mathcal{H}\}$). The definitions and arguments of section 3.5.3 can be adpated to this more general learning scenario by simply replacing all occurences of $\mathcal{G}^n \circ \overline{\mathcal{F}}$ in that section with $[H^n]_\sigma$. This yields the following theorem on the sample size m required for PC learning using the hypothesis space family H[13].

[13]Note that $l_{[H^n]_\sigma} = \bigcup_{\mathcal{H} \in H} \{l_\mathcal{H} \oplus \cdots \oplus l_\mathcal{H}\} \subseteq l_{H_\sigma} \oplus \cdots \oplus l_{H_\sigma}$ and so theorem B.6 applies to $l_{[H^n]_\sigma}$.

THEOREM 3.9. *Let H be a family of hypothesis spaces such that each hypothesis space $\mathcal{H} \in H$ is a set of functions mapping X into A. Let l be a loss function $l\colon Y \times A \to [0, M]$ and suppose that l_H is f-permissible*[14]*. Let $\vec{P} = (P_1, \ldots, P_n) \in \mathcal{P}_{l_{H_\sigma}}$ be n probability measures on Z. Let $\mathbf{z} \in Z^{(n,m)}$ be an (n, m) sample generated by sampling m times from Z according to each P_i. For all $0 < \alpha < 1$, $0 < \delta < 1$, $\nu > 0$ and any (n, m) sample learner*

$$\mathcal{A}\colon \bigcup_{\substack{n\geq 1\\ m\geq 1}} Z^{(n,m)} \to [H^n]_\sigma\,,$$

if

$$m > \frac{8M}{\alpha^2 \nu n} \ln \frac{4\mathcal{C}\left(\varepsilon, l_{[H^n]_\sigma}\right)}{\delta},$$

then

$$\Pr\{\mathbf{z} \in Z^{(n,m)}\colon d_\nu\left(E(l_{\mathcal{A}(\mathbf{z})}, \mathbf{z}), E(l_{\mathcal{A}(\mathbf{z})}, \vec{P})\right) > \alpha\} < \delta.$$

The $n = 1$ case of this theorem gives the sample size required for ordinary learning using the unbiased space H_σ,

$$m > \frac{8M}{\alpha^2 \nu} \ln \frac{4\mathcal{C}\left(\varepsilon, l_{H_\sigma}\right)}{\delta}.$$

Using the results on capacity from appendix C it is a fairly trivial exercise to show

$$\mathcal{C}\left(\varepsilon, l_{H_\sigma}\right) \leq \mathcal{C}\left(\varepsilon, l_{[H^n]_\sigma}\right) \leq \mathcal{C}\left(\varepsilon, l_{H_\sigma}\right)^n,$$

which, in conjuction with theorem 3.9, shows that at worst an (n, m) sample learner will require as many examples per function for PC learning as an ordinary learner, and at best a factor of n less than the ordinary learner. The ratio

$$I\left(H, n, \varepsilon\right) = \frac{1}{n} \frac{\log \mathcal{C}\left(\varepsilon, l_{[H^n]_\sigma}\right)}{\log \mathcal{C}\left(\varepsilon, l_{H_\sigma}\right)}$$

measures the usefulness of H for learning (n, m) samples, and will be called the *learning impedance* of H. A learning impedance of 1 indicates that the learner can do no better than ordinary learning with H, i.e. the learner would be just as well off learning a single task at a time with H_σ, while an impedance of $\frac{1}{n}$ (the best possible) indicates that PC learning of an (n, m) sample using H can be achieved with m a factor of n smaller than that required for ordinary learning.

For representation learning, $H = \{\mathcal{G} \circ f : f \in \mathcal{F}\}$, and the impedance of H obeys (roughly speaking)

$$I(H, n, \varepsilon) = \frac{\frac{1}{n} + \frac{\log \mathcal{C}(\varepsilon_1, l_\mathcal{G})}{\log \mathcal{C}^*_{l_\mathcal{G}}(\varepsilon_2, \mathcal{F})}}{1 + \frac{\log \mathcal{C}(\varepsilon_1, l_\mathcal{G})}{\log \mathcal{C}^*_{l_\mathcal{G}}(\varepsilon_2, \mathcal{F})}}$$

where $\varepsilon_1 + \varepsilon_2 = \varepsilon$. Thus the smaller the ratio of the logarithm of the capacities of $\mathcal{G}$ and $\mathcal{F}$, the smaller the learning impedance of H. The limiting cases of $I(H, n, \varepsilon) = 1$ and $I(H, n, \varepsilon) = \frac{1}{n}$ correspond to total domination of H by $\mathcal{G}$ and $\mathcal{F}$ respectively.

[14] Among other things this ensures $l_{[H^n]_\sigma}$ is permissible—see appendix A.

3.6.3. Learning the hypothesis space. In this section bounds on n and m ensuring small deviation between (33) and (34) are given, similar to the bounds in theorem 3.8. The learner is now taking an (n, m) sample $\mathbf{z}$ as input and producing a hypothesis space $\mathcal{H} \in H$ as output, so it is a map,

$$\mathcal{A}\colon \bigcup_{\substack{n\geq 1\\ m\geq 1}} Z^{(n,m)} \to H.$$

DEFINITION 3.4. For all $h \in H_\sigma$, define $\bar{l}_h : \mathcal{P}_{l_{H_\sigma}} \to [0, M]$ by

$$\bar{l}_h(P) = \langle \bar{l}_h \rangle_P.$$

For all $\mathcal{H} \in H$, define $l^*_{\mathcal{H}} \colon \mathcal{P}_{l_{H_\sigma}} \to [0, M]$ by

$$l^*_{\mathcal{H}}(P) = \inf_{h\in\mathcal{H}} \bar{l}_h(P).$$

Let l^*_H denote the set of all such functions.

Note that for representation learning, $l^*_H = l^*_{\mathcal{F}}$.

The same arguments leading to theorem 3.8 can be used in this more general context to yield the following theorem.

THEOREM 3.10. *Let H be a hypothesis space family such that each $\mathcal{H} \in H$ is a set of functions mapping X into A. Let l be a loss function $l\colon Y \times A \to [0, M]$, and assume that l is such that l_H is f-permissible*[15] *Let $\mathbf{z} \in Z^{(n,m)}$ be an (n, m) sample from Z generated according to some $Q \in \mathcal{P}_{\bar{l}_{H_\sigma}}$. For all $0 < \alpha, \delta < 1$, $\nu > 0$, if*

$$n \geq \frac{32M}{\alpha^2\nu} \ln \frac{8\mathcal{C}\left(\frac{\alpha\nu}{16}, l^*_H\right)}{\delta},$$

$$\text{and} \quad m \geq \frac{32M}{\alpha^2\nu n} \ln \frac{8\mathcal{C}\left(\frac{\alpha\nu}{16}, l_{[H^n]_\sigma}\right)}{\delta},$$

and $\mathcal{A}$ is any hypothesis space learner,

$$\mathcal{A}\colon \bigcup_{\substack{n\geq 1\\ m\geq 1}} Z^{(n,m)} \to H,$$

then

$$\Pr\left\{\mathbf{z} \in Z^{(n,m)} \colon d_\nu\left(E^*(\mathcal{A}(\mathbf{z}), \mathbf{z}), E^*(A(\mathbf{z}), Q)\right) > \alpha\right\} \leq \delta.$$

[15] This guarantees permissibility of both $l_{[H^n]_\sigma}$ and l^*_H. See lemma A.2.

CHAPTER 4

Algorithms and Experiments

In this chapter algorithms are given for learning representations with Artificial Neural Networks, and the results of several experiments are presented which verify the theoretical conclusions of the previous chapter.

All the examples considered here concern perhaps the most familiar of learning scenarios—that of learning Boolean functions over some input domain X. Thus the outcome space Y is the set $\{0,1\}$, or equivalently $\{\pm 1\}$, the probability measures $P \in \mathcal{P}$ consist of a measure P on X and a function $f : X \to Y$, so that samples $z = (x, y) \in Z = X \times Y$ are always of the form $(x, f(x))$. For the particular learning scenarios treated in this section P is the same regardless of the function f. Thus, denoting the space of all Boolean functions on the domain X by $\mathcal{F}_2(X)$, the environmental measure Q is effectively a measure on $\mathcal{F}_2(X)$. An (n, m) sample $\mathbf{z} \in Z^{(n,m)}$ is produced by first sampling n times from $\mathcal{F}_2(X)$ according to Q to generate a list of functions $\vec{f} = (f_1, \ldots, f_n)$, and then sampling m times from X, n times over according to P to generate $x_{11}, \ldots, x_{1m}, \ldots, x_{n1}, \ldots, x_{nm}$. The final (n, m) sample is

$$\mathbf{z} = \begin{array}{cccc} (x_{11}, f_1(x_{11})), & (x_{12}, f_1(x_{12})), & \ldots & (x_{1m}, f_1(x_{1m})) \\ (x_{21}, f_2(x_{21})), & (x_{22}, f_2(x_{22})), & \ldots & (x_{2m}, f_2(x_{2m})) \\ \vdots & \vdots & \ddots & \vdots \\ (x_{n1}, f_n(x_{n1})), & (x_{n2}, f_n(x_{n2})), & \ldots & (x_{nm}, f_n(x_{nm})) \end{array}$$

The learner's hypotheses are functions $f : X \to [0,1]$ so $\mathbf{z}$ is intended to give the learner some idea of the kinds of functions it will be asked to learn.

The three main advantages of representation learning over ordinary machine learning—as established in chapter 3—are:

- The number of examples m required per function for PC learning decreases as the number of functions n being learnt increases.
- If n is sufficiently large—determined principally by the capacity of the representation space $\mathcal{F}$—then the representation learnt will, with high probability, be a good one for learning future functions drawn from the same environment Q.
- Learning with a representation will in general be far more efficient than learning without.

In order to take advantage of these theoretical results in practical learning problems an algorithm must be found for learning representations from (n, m) samples.

If the representation space $\mathcal{F}$ and the space $\mathcal{G}$ consist of feedforward networks (as defined in chapter D), then it is a simple matter to extend existing gradient-descent "backpropagation" type algorithms for training those networks to training algorithms for representation learning. Such an algorithm is used to learn representations in two experimental situations. In the first the environment Q is restricted to have support only on a kind of "translation invariant" Boolean function, and in the second Q's support is restricted to a subset of the symmetric Boolean functions. In both cases the three theoretical results summarised above are well supported. These experiments are discussed in detail in the last two sections of this chapter. However the first section gives the details of a somewhat simpler experiment using binary neural networks and exhaustive search rather than gradient descent to learn representations.

All experiments were performed on the CM5 machine at the South Australian Centre for Parallel Supercomputing, and all programs were written in C^*, Thinking Machine's parallel version of C.

4.1. Learning Binary Neural Network Representations

In this first experiment the second central tenet of representation learning theory is verified, namely if n and m are sufficiently large, a representation f with small empirical loss on $\mathbf{z} \in Z^{(n,m)}$ will be "useful" for learning future functions drawn from the same environment. The general idea of the experiment is to first choose an arbitrary representation space $\mathcal{F}$ and output function space $\mathcal{G}$ for the representation learner to use, and then the support of the environmental measure Q is chosen to be $\mathcal{G} \circ f^*$ where f^* is some representation in $\mathcal{F}$. By doing this it is guaranteed that for any (n, m) sample chosen according to Q, there will exist at least one representation (namely f^*) with zero empirical loss on the sample, and also if $\mathcal{F}$ and $\mathcal{G}$ are sufficiently simple it will be computationally feasible to find a representation with zero-empirical loss on any (n, m) sample simply by exhaustively searching through all representations $f \in \mathcal{F}$. The main experiment consisted of generating (n, m) samples for many different values of n and m, finding all representations with zero empirical loss on the sample and then measuring the performance of a learner that used the resulting representation to learn. The details of the experiment follow.

The input space X of the learner was chosen to be $X = \{\pm 1\}^5$ and all examples $x \in X$ were generated according to the uniform distribution on X, which will be denoted by P. With this restriction on the distribution on X, the only thing that the environmental measure Q has control over are the functions seen by the learner. The support of Q, a subset F of $\mathcal{F}_2(X)$, was constructed by firstly selecting at random a *binary neural network* f^* with five input nodes, no hidden layers and three output nodes, and then composing any Boolean function with the output of f^*. A binary neural network is one with $\{\pm 1\}$ valued weights, no thresholds and the hard-limiting sign function

$$\theta(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \leq 0 \end{cases}$$

for a squashing function. Thus f^* is a function mapping[1] $X = \{\pm 1\}^5$ into $\{\pm 1\}^3$. For later reference, denote the space of all such binary neural networks by $N_{5,3}$. Q's support F thus consisted of all Boolean functions from X into $\{\pm 1\}$ that can be

[1] There is nothing special about the numbers being used in this example, they were chosen merely to ensure the tractability of the experiment.

formed by the composition of any function from $\mathcal{F}_2(\{\pm 1\}^3)$ with f^*. F is a highly restricted subset of all possible Boolean functions on X because F contains at most 256 functions whereas $|\mathcal{F}_2(X)| = 2^{32}$. Functions were selected from F by choosing a function from $\mathcal{F}_2(\{\pm 1\}^3)$ uniformly at random and then composing it with f^*.

To ensure that the learner could achieve zero empirical loss on any (n, m) sample the choices $\mathcal{G} = \mathcal{F}_2(\{\pm 1\}^3)$ for the learner's output function space and $\mathcal{F} = N_{5,3}$ for its representation space were made. Hence the output space Y and the action space A were the same $\{\pm 1\}$. The loss function l was chosen to be simply $l(a, y) = 0$ if $a = y$ and $l(a, y) = 1$ otherwise, i.e

$$l(a, y) = 1 - \delta(a, y)$$

where δ is the Kronecker delta function. With this choice of l the empirical loss of any function $g \circ f \in \mathcal{G} \circ \mathcal{F}$ on a sample $\vec{z} = ((x_1, y_1), \ldots, (x_m, y_m))$ is given by

$$E(g \circ f, \vec{z}) = 1 - \frac{1}{m} \sum_{i=1}^{m} \delta(g \circ f(x_i), y_i).$$

The empirical loss of a representation $f \in \mathcal{F}$ on an (n, m) sample $\mathbf{z} = (\vec{z}_1, \ldots, \vec{z}_n)$ is given by (recall equation (10) in chapter 3),

$$E^*_{\mathcal{G}}(f, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} \inf_{g \in \mathcal{G}} E(g \circ f, \vec{z}_i).$$

The fact that $\mathcal{G}$ consisted of every Boolean function on the range of $\mathcal{F}$ simplified the task of computing the empirical loss of a representation f. To see this, note that given any sample $\vec{z}_i = ((x_{i1}, y_{i1}), \ldots, (x_{im}, y_{im}))$, each input vector x_{ij} will be mapped by f to one of the eight elements of $\{\pm 1\}^3$. Denoting the elements of $\{\pm 1\}^3$ by $s_1, \ldots, s_8$, $\vec{z}_i$ can be partitioned by f into a disjoint union of sets $S^i_1, \ldots, S^i_8$ where $S^i_a = \{(x_j, y_j) \in \vec{z}_i : f(x_j) = s_a\}$, $1 \leq a \leq 8$. Note that some of the S^i_a may be empty. For each set S^i_a let $\text{plus}(S^i_a)$ be the number of points (x_j, y_j) in S^i_a such that $y_j = 1$ and similarly let $\text{minus}(S^i_a)$ be the number of points for which $y_j = -1$. Clearly the function g that minimizes the empirical loss of $g \circ f$ on $\vec{z}_i$ is the one $g(s_i) = 1$ if $\text{plus}(S^i_a) > \text{minus}(S^i_a)$ and $g(s_i) = -1$ if $\text{minus}(S^i_a) > \text{plus}(S^i_a)$ (the choice of $g(s_i)$ does not affect the value of the empirical loss if $\text{plus}(S^i_a) = \text{minus}(S^i_a)$). With this choice of g the empirical loss of $g \circ f$ on $\vec{z}_i$ is $\frac{1}{m} \sum_{a=1}^{8} \min(\text{plus}(S^i_a), \text{minus}(S^i_a))$ and hence the empirical loss of the representation f on the sample $\mathbf{z} = (\vec{z}_1, \ldots, \vec{z}_n)$ is given by

$$E^*_{\mathcal{G}}(f, \mathbf{z}) = \frac{1}{mn} \sum_{i=1}^{n} \sum_{a=1}^{8} \min\left(\text{plus}(S^i_a), \text{minus}(S^i_a)\right). \tag{36}$$

The important thing about this formula is that it can be computed directly from the behaviour of f on the sample $\mathbf{z}$, so that it is not necessary to explicitly search the space $\mathcal{G}$ to determine f's empirical loss.

4.1.1. The Experiment. (n, m) samples $\mathbf{z}$ were generated according to the environment Q described above for values of m in the set $\{2, 6, 10, 14, 18, 22\}$ and values of n in the set $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Ten independent (n, m) samples were generated for each (n, m) pair, and the results were averaged across all ten samples. For each (n, m) sample $\mathbf{z}$, $\mathcal{F}$ was exhaustively searched to find *all* representations with *zero* empirical

loss on $\mathbf{z}$, utilising formula (36). The environment Q was then sampled ten more times to generate ten new Boolean functions $g_1 \circ f^*, \ldots, g_{10} \circ f^*$, and training sets with sizes of 2 up to 22 again were generated for each of the ten functions. Thus there were 60 new training sets $\vec{z}_2^1, \vec{z}_6^1, \vec{z}_{10}^1, \ldots, \vec{z}_{14}^{10}, \vec{z}_{18}^{10}, \vec{z}_{22}^{10}$. Each zero-loss representation f was then tested for its learning ability against each of the ten new test functions as follows. For each training set $\vec{z}_m^i$, $1 \leq i \leq 10, m \in \{2, 6, 10, 14, 18, 22\}$, the function $g \in \mathcal{G}$ giving the smallest value of $E(g \circ f, \vec{z}_m^i)$ was found and then the total error between $g \circ f$ and the original function $g_i \circ f^*$ was determined (simply by going through all 32 (2^5) vectors in the input space X and counting up the number of times $g \circ f$ and $g_i \circ f^*$ disagreed). For each training set size and each zero-loss representation the total error was averaged across all ten functions, and then further averaged across all of the zero-loss representations. This rather cryptic process yields a number x for each triple n, m and m_1 which is an estimate of the expected generalisation error that a learner would see if it received an (n, m) sample $\mathbf{z}$ drawn according to Q, selected a representation $\hat{f}$ uniformly at random with zero loss on $\mathbf{z}$, then received a further m_1 examples of a new function drawn according to Q and found a function $g \circ \hat{f}$ with smallest empirical loss on the new sample. x is an estimate of the expected true error (i.e generalisation error) of $g \circ \hat{f}$. Plotting x as a function of m_1 for each value of m and n in the range specified above yields a set of expected *generalisation curves*. For comparison purposes two more generalisation curves were also generated. The first is the curve that would be produced in an ordinary learning scenario, i.e using the same training sets $z_2^1, z_6^1, \ldots$ above, the full space $\mathcal{G} \circ \mathcal{F}$ was searched to find all networks with zero loss on each training set and then the total error for each network was computed and averaged across all ten functions and across all networks with zero loss. The second curve was the one that would be generated by a representation learner that had first learnt the exact representation f^*. Note that the exact representation curve is the best curve (on average) that a representation learner could achieve.

The generalisation curves for representative values of m and n are plotted in figures 4.1.1 and 4.1.2, along with a plot of the ordinary learning curve and the exact representation curve. Notice that the generalisation behaviour of the exact representation curve is superior to the ordinary learning curve—the exact representation learner has been given a big head start by having its input layer weights already set to their correct values whereas the ordinary learner has to determine both the input layer weights and the output function g. For small values of n and m the generalisation curves for the representation learner are considerably worse than the ordinary machine learning curve (the cause of the dip in the early part of the ordinary learning curve is a mystery, but may simply be an artefact of insufficient sampling). This is to be expected because there is insufficient information in a small (n, m) sample to determine a good representation and so, after the representation learning phase, the representation learner in most cases will be locked into using a bad representation when learning new functions. However, as n and m increase, the generalisation curves of the representation learner improve, and eventually the representation learner begins to perform better than the ordinary learner. As n increases even further, for large enough values of m, the learner's performance approaches that of the best possible (the exact) representation curve. Thus, for large enough values of n and m, a representation f with zero empirical loss on the data is pretty close to the true representation f^*, and hence exhibits generalisation behaviour close to that achieved by f^*.

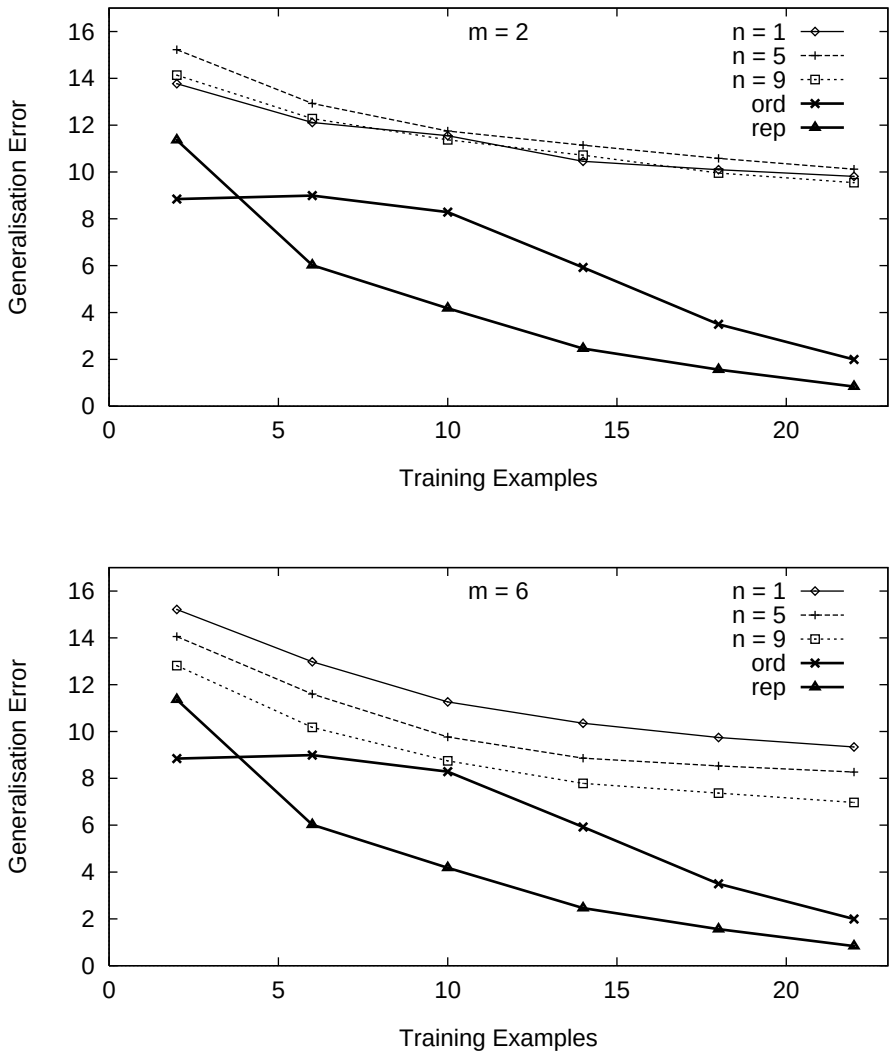

FIGURE 4.1.1. Binary representation generalisation curves ($m = 2$ and $m = 6$). n is the number of functions sampled from the environment and m is the number of times each function was sampled. The curves labelled "rep" and "ord" are the best representation learning curve and ordinary learning curve respectively.

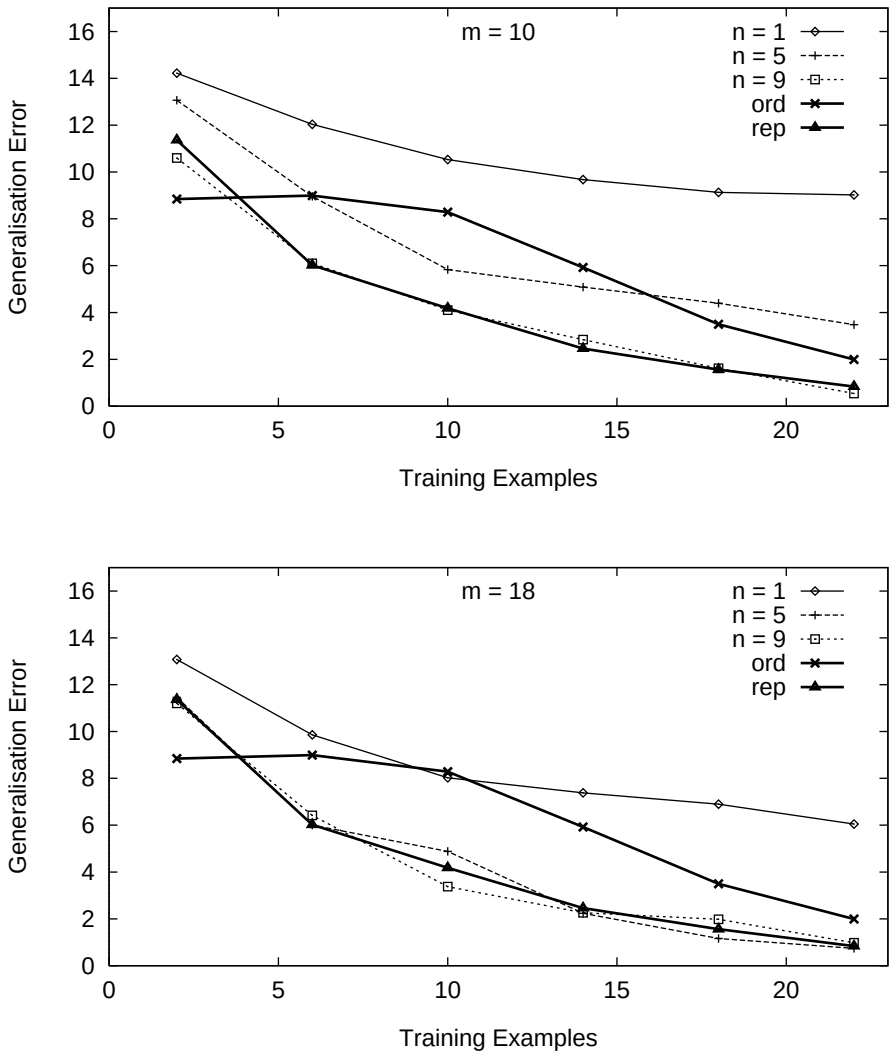

FIGURE 4.1.2. Binary representation generalisation curves ($m = 10$ and $m = 18$). n is the number of functions sampled from the environment and m is the number of times each function was sampled. The curves labelled "rep" and "ord" are the best representation learning curve and ordinary learning curve respectively.

The conclusion from this experiment is clear: for sufficiently large n and m the generalisation performance of a learner is significantly improved if it first learns an internal representation.

To some extent this experiment is quite artificial; in particular, the environment of the learner was hardly a very realistic one: Boolean functions formed by composition of an arbitrary function with a binary neural network. Also the learning algorithm employed—exhaustive search—is clearly not a feasible one in practice. However as an in principle demonstration of the effectiveness of representation learning it is fine. The next section is devoted to a discussion of two more experiments that are closer to the kind of problems one sees in real-life machine learning situations (although they are still toy problems).

4.2. Learning Continuous Neural Network Representations

In this section it is shown how gradient-descent may be used to train neural networks to learn representations from (n, m) samples[2]. Two sets of experiments are performed. The first is a kind of very basic machine vision problem where the neural network is attached to a one-dimensional "retina" and has to learn an appropriate representation for an environment in which all the functions are invariant under *translations* in their inputs. In the second the functions in the environment are all invariant under permutations of the inputs, i.e they are all *symmetric* functions. These experiments were chosen because they are simple enough to be performed within a reasonable amount of computer time, but complex enough to exhibit the phenomena desired, and because it is easily verified that a neural network representation exists for both experiments.

Both experiments support the three main theoretical results of chapter 3. Firstly, in both experiments as the number of functions being learnt increases there is a corresponding decrease in the number of examples required per function for good generalisation. Secondly, if the number of functions learnt is sufficiently large then the resulting representation is a good one to use for learning further functions drawn from the same environment, and thirdly, the number of examples required to ensure good generalisation is far less if one initially learns a good representation.

4.2.1. Gradient Descent for Representation Learning. Given an (n, m) sample

$$\mathbf{z} = \begin{matrix} z_{11} & \dots & z_{1m} \\ \vdots & \ddots & \vdots \\ z_{n1} & \dots & z_{nm} \end{matrix}$$

where $z_{ij} = (x_{ij}, y_{ij})$, a representation space $\mathcal{F}$ and an output function space $\mathcal{G}$, it is required that the learner find a representation $f \in \mathcal{F}$ minimizing the *mean-squared* representation error, that is

$$E^*_{\mathcal{G}}(f, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} \inf_{g \in \mathcal{G}} \frac{1}{m} \sum_{j=1}^{m} \left(g \circ f(x_{ij}) - y_{ij} \right)^2$$

must be minimal.

[2] The methods presented here can be generalised quite easily to cover learning with any kind of continuous function network, not just neural networks.

$\mathcal{F}$ and $\mathcal{G}$ are assumed to consist of *differentiable families of neural networks*[3]. The most common procedure for training differentiable neural networks is to use some form of gradient descent algorithm (vanilla backprop, conjugate gradient, etc) to minimize the error of the network on the sample being learnt. For example, in ordinary learning the learner would receive a single sample $\vec{z} = ((x_1, y_1), \ldots, (x_m, y_m))$ and would perform some form of gradient descent to find a function $g \circ f \in \mathcal{G} \circ \mathcal{F}$ such that

$$E(g \circ f, \vec{z}) = \frac{1}{m} \sum_{i=1}^{m} (g \circ f(x_i) - y_i)^2 \tag{37}$$

is minimal. This procedure works because it is a relatively simple matter to compute the gradient, $\nabla_{\mathbf{w}} E(g \circ f, \mathbf{z})$, where $\mathbf{w}$ are the parameters (weights) of the networks in $\mathcal{F}$ and $\mathcal{G}$.

Applying this method directly to the problem of minimising $E^*_{\mathcal{G}}(f, \mathbf{z})$ above would mean calculating the gradient $\nabla_{\mathbf{w}} E^*_{\mathcal{G}}(f, \mathbf{z})$ where now $\mathbf{w}$ refers only to the parameters of $\mathcal{F}$. However, due to the presence of the infimum over $\mathcal{G}$ in the formula for $E^*_{\mathcal{G}}(f, \mathbf{z})$, calculating the gradient in this case is much more difficult than in the ordinary learning scenario. Fortunately this problem has a simple solution. Recalling the definition of the function space $\mathcal{G}^n \circ \overline{\mathcal{F}}$ from section 3.5.2 (it is just the set of all sequences of functions $\vec{g} \circ \bar{f} = (g_1 \circ f, \ldots, g_n \circ f)$ where $g_1, \ldots, g_n \in \mathcal{G}$ and $f \in \mathcal{F}$) , define the *mean-squared error* of $\vec{g} \circ \bar{f} \in \mathcal{G}^n \circ \overline{\mathcal{F}}$ with respect to an (n, m) sample $\mathbf{z}$ as

$$E(\vec{g} \circ \bar{f}, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{m} \sum_{j=1}^{m} (g_i \circ f(x_{ij}) - y_{ij})^2 \tag{38}$$

where $\vec{g} = (g_1, \ldots, g_n)$. Note that $\frac{1}{m} \sum_{j=1}^{m} (g_i \circ f(x_{ij}) - y_{ij})^2$ is just the mean-squared error of the network $g_i \circ f$ on the sample $\vec{z}_i = ((x_{i1}, y_{i1}), \ldots, (x_{im}, y_{im}))$, so that the mean-squared error of $\vec{g} \circ \bar{f}$ on $\mathbf{z} = (\vec{z}_1, \ldots, \vec{z}_n)$ is nothing more than the average of the mean-squared error of each $g_i \circ f$ on $\vec{z}_i$. Clearly,

$$\inf_{\vec{g} \circ \bar{f} \in \mathcal{G}^n \circ \overline{\mathcal{F}}} E(\vec{g} \circ \bar{f}, \mathbf{z}) = \inf_{f \in \mathcal{F}} E^*_{\mathcal{G}}(f, \mathbf{z}). \tag{39}$$

If $\vec{g} \circ \bar{f}$ is such that

$$\left| E(\vec{g} \circ \bar{f}, \mathbf{z}) - \inf_{\vec{g} \circ \bar{f} \in \mathcal{G}^n \circ \overline{\mathcal{F}}} E(\vec{g} \circ \bar{f}, \mathbf{z}) \right| \leq \varepsilon,$$

equation (39) implies, along with the fact that, by definition, $E(\vec{g} \circ \bar{f}, \mathbf{z}) > E^*_{\mathcal{G}}(f, \mathbf{z})$,

$$\left| E^*_{\mathcal{G}}(f, \mathbf{z}) - \inf_{f \in \mathcal{F}} E^*_{\mathcal{G}}(f, \mathbf{z}) \right| \leq \varepsilon.$$

Thus if the learner finds a function $\vec{g} \circ \bar{f}$ such that $E(\vec{g} \circ \bar{f}, \mathbf{z})$ is close to minimal then so too will $E^*_{\mathcal{G}}(f, \mathbf{z})$ be close to minimal. So instead of minimising $E^*_{\mathcal{G}}(f, \mathbf{z})$ the learner can equivalently minimise $E(\vec{g} \circ \bar{f}, \mathbf{z})$. The advantage of this approach is that essentially the same techniques used for computing the gradient in ordinary learning can be used to compute the gradient of $E(\vec{g} \circ \bar{f}, \mathbf{z})$.

[3] The algorithm presented here does not actually depend in any way on $\mathcal{F}$ and $\mathcal{G}$ being neural networks—it applies equally well if $\mathcal{F}$ and $\mathcal{G}$ are any parameterised families of differentiable functions.

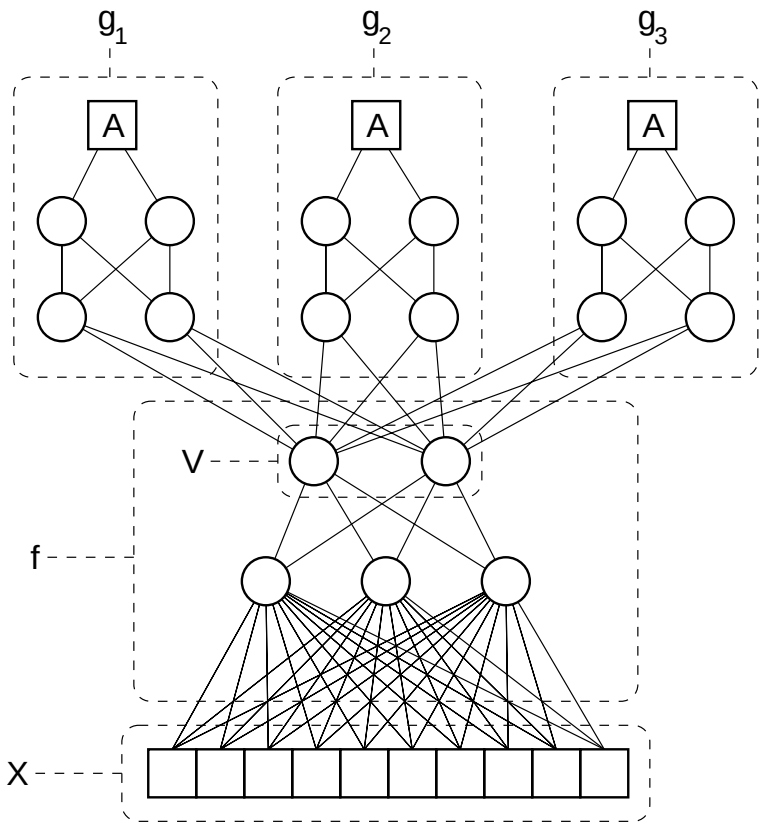

FIGURE 4.2.3. A neural network for representation learning.

An example of a neural network of the form $\vec{g} \circ \bar{f}$ is illustrated in figure 4.2.3, where $\vec{g} = (g_1, g_2, g_3)$. In the figure, f belongs to a family $\mathcal{F}$ consisting of neural networks with ten inputs, one hidden layer with three nodes and one output layer with two nodes. g_1, g_2 and g_3 belong to a family $\mathcal{G}$ consisting of networks with two input nodes, two hidden layers each with two hidden nodes and one output node. Note that as the networks in $\mathcal{G}$ are composed with networks from $\mathcal{F}$, the dimensionality of the input of $\mathcal{G}$ (i.e the number of input nodes for the networks in $\mathcal{G}$) must equal the dimensionality of the output of $\mathcal{F}$. The language introduced in the theoretical chapters is also illustrated in the diagram, with the input space denoted by X, the internal space by V and the output space of the network denoted by A.

Consider now the problem of computing the derivative of $E(\vec{g} \circ \bar{f}, \mathbf{z})$ with respect to a weight in the i^{th} output network g_i. Denoting the weight by w_i and recalling equation (38),

$$\frac{\partial}{\partial w_i} E(\vec{g} \circ \bar{f}, \mathbf{z}) = \frac{1}{n} \frac{\partial}{\partial w_i} \frac{1}{m} \sum_{j=1}^{m} (g_i \circ f(x_{ij}) - y_{ij})^2$$

which is just $\frac{1}{n}$ times the derivative of the ordinary learning error (37) of $g_i \circ f$ on sample $\vec{z}_i = ((x_{i1}, y_{i1}), \ldots, (x_{im}, y_{im}))$ with respect to the weight w_i. This can be

computed using any standard formula for the derivative (e.g backpropagation).

Alternatively, if w is a weight in the representation network f then

$$\frac{\partial}{\partial w}E(\vec{g}\circ\bar{f},\mathbf{z}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial w}\frac{1}{m}\sum_{j=1}^{m}\left(g_i\circ f(x_{ij})-y_{ij}\right)^2$$

which is simply the *average* of the derivatives of the ordinary learning errors over all the samples $(\vec{z}_1,\ldots,\vec{z}_n)$ and hence can also be computed using standard formulae.

4.2.2. Representation Learning Using Backpropagation. In this section it is outlined for those readers familiar with the backpropagation learning algorithm (Rumelhart et. al. (1986)) how it can be used to learn representations. It is assumed that the architectures of the networks in $\mathcal{F}$ and $\mathcal{G}$ are known[4] and that the learner has been supplied with an (n,m) sample

$$\mathbf{z} = \begin{array}{cccc} (x_{11},y_{11}) & (x_{12},y_{12}) & \ldots & (x_{1m},y_{1m}) \\ (x_{21},y_{21}) & (x_{22},y_{22}) & \ldots & (x_{2m},y_{2m}) \\ \vdots & \vdots & \ddots & \vdots \\ (x_{n1},y_{n1}) & (x_{n2},y_{n2}) & \ldots & (x_{nm},y_{nm}) \end{array}$$

Note that it is not necessary for the architectures of all the output networks to be the same.

(1) Construct the network $\mathcal{G}^n\circ\overline{\mathcal{F}}$ as shown for example in figure 4.2.3 and randomly initialise all the weights in the representation and output networks.

(2) For each subsample (x_{ij},y_{ij}):
- Initialise the input nodes of f to the components of x_{ij} and perform a *forward pass* through f to yield $f(x_{ij})$ at f's output nodes.
- Set the input nodes of the i^{th} output network, g_i to the value of f's output nodes (note that the number of output nodes from f must equal the number of input nodes to each of the $g's$).
- Perform a forward pass through the i^{th} output network g_i to yield the network's output $o_{ij}=g_i(f(x_{ij}))$ on input x_{ij}.
- Compute the error $(o_{ij}-y_{ij})^2$.
- Backpropagate this error information in the usual way through the output network g_i *and* the representation network f.
- Update the gradient for each weight based on the backpropagated errors.

(3) The total error is the sum of all the individual $(o_{ij}-y_{ij})^2$ errors. This should be divided by $m\times n$ to yield the average error.

(4) Update the weights using your favourite gradient procedure (vanilla gradient descent, conjugate gradient descent, etc) and start all over again at (2).

Note that for each input x_{ij}, error information is only backpropagated through the output network g_i. It may well be the case that the desired output can easily be

[4] Architecture selection is still a major unsolved problem in Neural Network research and will not be addressed here. However it is argued in chapter 6 that the information provided by (n,m) samples is much more appropriate for hierarchical construction of neural networks than is the information provided by a single sample. Some ideas are also presented for how this information may be used to "grow" a representation of an appropriate size.

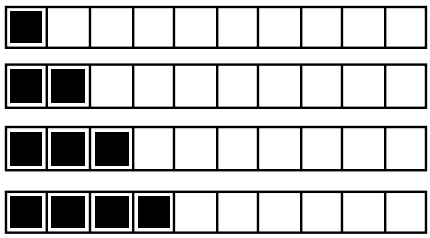

FIGURE 4.2.4. The inputs seen by the learner in experiment one consisted of these four and any of their translates (including wrapping around at the edge).

computed for the other output networks too, in which case ordinary backpropagation can be used to train the entire network.

4.2.3. Experiment One: Learning Translation Invariance. In this first experiment a neural network was trained to perform a very simple "machine vision" task in which it had to learn to recognise different objects drawn on its one dimensional retina, no matter where they were drawn. Thus all the tasks in the environment of the neural network were *translationally invariant*.

The input space X consisted of a ten-pixel, one-dimensional "retina" in which all the pixels could be either on (1) or off (0) (so in fact $X = \{0,1\}^{10}$). However the network did not see all possible input vectors during the course of its training, the only vectors with a non-zero probability of appearing in the training set were those consisting of from one to four active adjacent pixels placed somewhere in the retina (see figure 4.2.4). Thus there were 40 input vectors in all: 4 objects times 10 possible positions (for asthetic reasons the objects were allowed to "wrap around" at the edge so that the retina should more appropriately be viewed as a circle rather than a line.)

The functions in the environment of the network consisted of all possible *translationally invariant* Boolean functions over the input space (except the trivial "constant 0" and "constant 1" functions). The requirement of translation invariance means that the environment consisted of just 14 different functions—all the Boolean functions on four objects (of which there are $2^4 = 16$) less the two trivial ones (the functions are shown in figure 4.2.5). Thus the environment was highly restricted, both in the number of different input vectors seen—40 out of a possible 1024—and in the number of different functions to be learnt—14 out of a possible 2^{1024}. It is this kind of principled restriction of the environment of a learner that makes generalisation possible (and of course without it generalisation is impossible). (n, m) samples were generated from this environment by firstly choosing n functions (with replacement, so a function could be chosen more than once) uniformly from the fourteen possible, and then choosing m input vectors (with replacement again), for each function, uniformly from the 40 possible input vectors.

The architecture of the network was similar to the one shown in figure 4.2.3, the only difference being that the output networks $g \in \mathcal{G}$ for this experiment had only one hidden layer, not two. The network in the figure is for learning $(3, m)$ samples (it has 3 output networks), in general for learning (n, m) samples the network will have n output networks. The network architecture is more complex than is needed to solve the learning problem, however it was deliberately made that way to demonstrate that the learning algorithm works on complex networks.

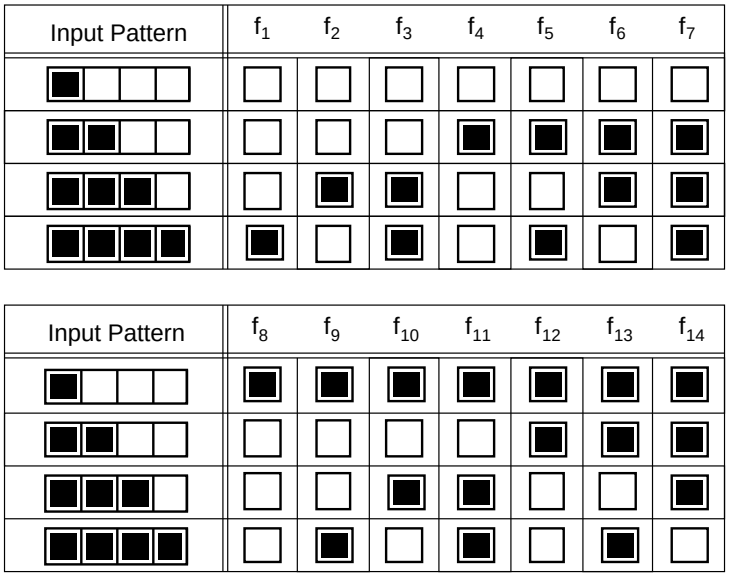

FIGURE 4.2.5. The functions in the environment of the learner: all *translationally invariant* Boolean functions on the restricted set of inputs shown in figure 4.2.4, excluding the two constant functions.

Each node in the network computed its activation by applying the standard sigmoid squashing function to a weighted sum of its inputs. Thus if a is the activation of the node, $\vec{w} = (w_1, \ldots, w_n)$ its weight vector, w_T its threshold, and $\vec{x} = (x_1, \ldots, x_n)$ is the vector of inputs to the node, then

$$a = \frac{1}{1 + e^{-(\sum_{i=1}^{n} w_i x_i + w_T)}}$$

The network was trained on (n, m) samples with n ranging from 1 to 21 in steps of four and m ranging from 1 to 151 in steps of 10. Conjugate-gradient descent was used with exact line search with the gradients for each weight computed according to the algorithm outlined in the previous section. Training was halted if the mean-squared error (equation (38)) fell below 10^{-6} *or* the L^{∞} error (i.e the maximum absolute difference between the desired output and the actual output for any of the training examples and any of the output networks) fell below 0.01. These are fairly stringent halting requirements but as a solution is known to exist they are not unreasonable. If the relative reduction in error over 5 training iterations was less than 0.01%, the network was deemed to have struck a local minimum and training was restarted with a new randomly generated set of weights. Weights were chosen uniformly in the interval $[-1, 1]$. All weights were clipped when their absolute values exceeded 20, except for the thresholds which were permitted to grow to 80, although they never got that far in any of the simulations. At any iteration of the algorithm, conjugate gradient descent was only performed on the subspace of weights excluding the clipped weights. However, if the gradient vector at any stage indicated that a reduction in absolute value of a clipped weight would occur if a step was taken in the downhill direction, then the clipped weight was reintroduced into the optimization process.

This method prevented the network heading blindly for the local minimum that often exists "at infinity" due to the asymptotic behaviour of the sigmoid squashing function. The clipping of a weight for several iterations and then its subsequent reintroduction into the optimisation process was not an uncommon event, and occasionally led to the discovery of a global minimum that would not otherwise have been discovered. However the main utility of this technique was its early termination of unfruitful directions of search.

Once the network had sucessfully learnt the (n, m) sample its generalization ability was tested on all n functions in the training set. In this case the generalisation error (i.e *true* error) could be computed exactly by calculating the network's output (for all n functions) for each of the 40 input vectors, and comparing the result with the desired output. Both the mean-squared and L^∞ generalisation errors were computed.

In an ordinary learning situtation the generalisation error of a network would be plotted as a function of m, the number of examples in the training set, resulting in a *generalisation curve*. For representation learning there are two parameters m and n so the curve becomes a *generalisation surface*. Plots of the generalisation surface are shown in figure 4.2.6 for three independent simulations. All three cases support the theoretical result that the number of examples m required for good generalisation decreases with increasing n. This is most easily seen by observing the behaviour of the surface for fixed values of m. Although it is out of range and not shown in the plots, the number of examples required for good generalisation was around 100–150 when only one function from the environment was learnt ($n = 1$, the ordinary learning scenario), while the plots show that less than 21 examples per function are required if $n \geq 17$.

For (n, m) samples that led to a generalisation error of less than 0.01, the representation network f was extracted and tested for its *true error*, where this is defined as in equation (11) and in the current framework translates to

$$E^*_{\mathcal{G}}(f, Q) = \frac{1}{560} \sum_{i=1}^{14} \inf_{g \in \mathcal{G}} \sum_{j=1}^{40} (g \circ f(x_j) - f_i(x_j))^2$$

where $x_1, \ldots, x_{40}$ are the 40 input vectors seen by the learner and $f_1, \ldots, f_{14}$ are all the functions in the environment. $E^*_{\mathcal{G}}(f, Q)$ measures how useful the representation f is for learning all functions in the environment, not just the ones used in generating the (n, m) sample f was trained on. To measure $E^*_{\mathcal{G}}(f, Q)$, entire training sets consisting of 40 input-output pairs were generated for each of the 14 functions in the environment. The architecture used for learning (n, m) samples was reused, only this time with just one output network g and f as the representation network. The network was taught each of the fourteen functions in turn by keeping the weights of f fixed and using conjugate-gradient descent to set the weights of g so that the combination $g \circ f$ had minimal error on the training set. To be (nearly) certain that a minimal solution had been found for each of the functions, learning was started from 32 different random weight initialisations for g (this number was chosen so that the CM5 could perform all the restarts in parallel) and the best result from all 32 recorded. For each (n, m) sample giving perfect generalisation, $E^*_{\mathcal{G}}(f, Q)$ was calculated and then averaged over all (n, m) samples with the same value of n, and finally averaged over all three simulations, to give an indication of the behaviour of $E^*_{\mathcal{G}}(f, Q)$ as a function of n. This is plotted in figure 4.2.9, along with the L^∞ representation error for the three simulations (i.e, the maximum error over all 14 functions and all 40 examples and over

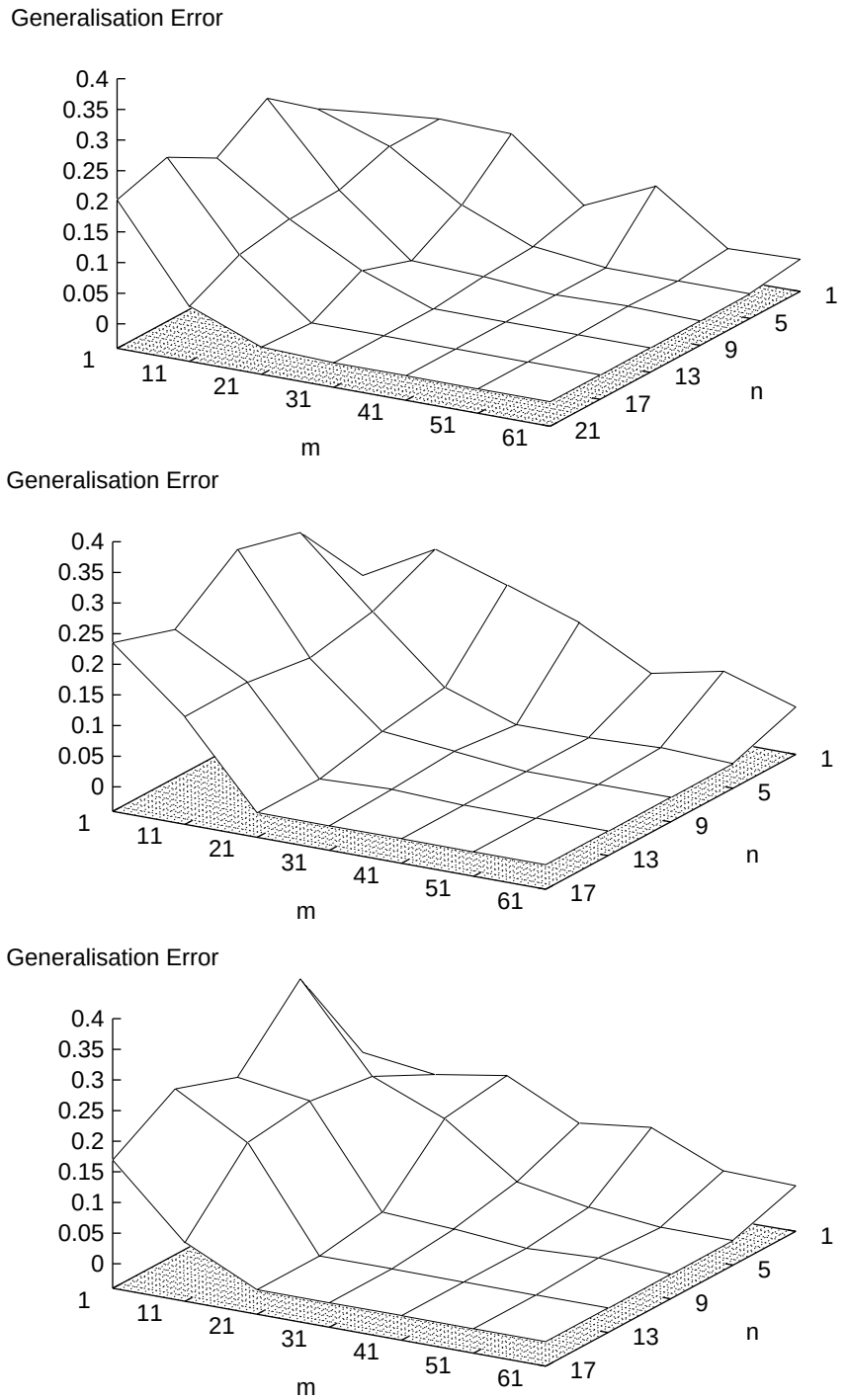

FIGURE 4.2.6. Mean-squared generalisation error as a function of n—the number of functions learnt from the environment—and m—the number of examples per function, for three independent simulations from the first experiment.

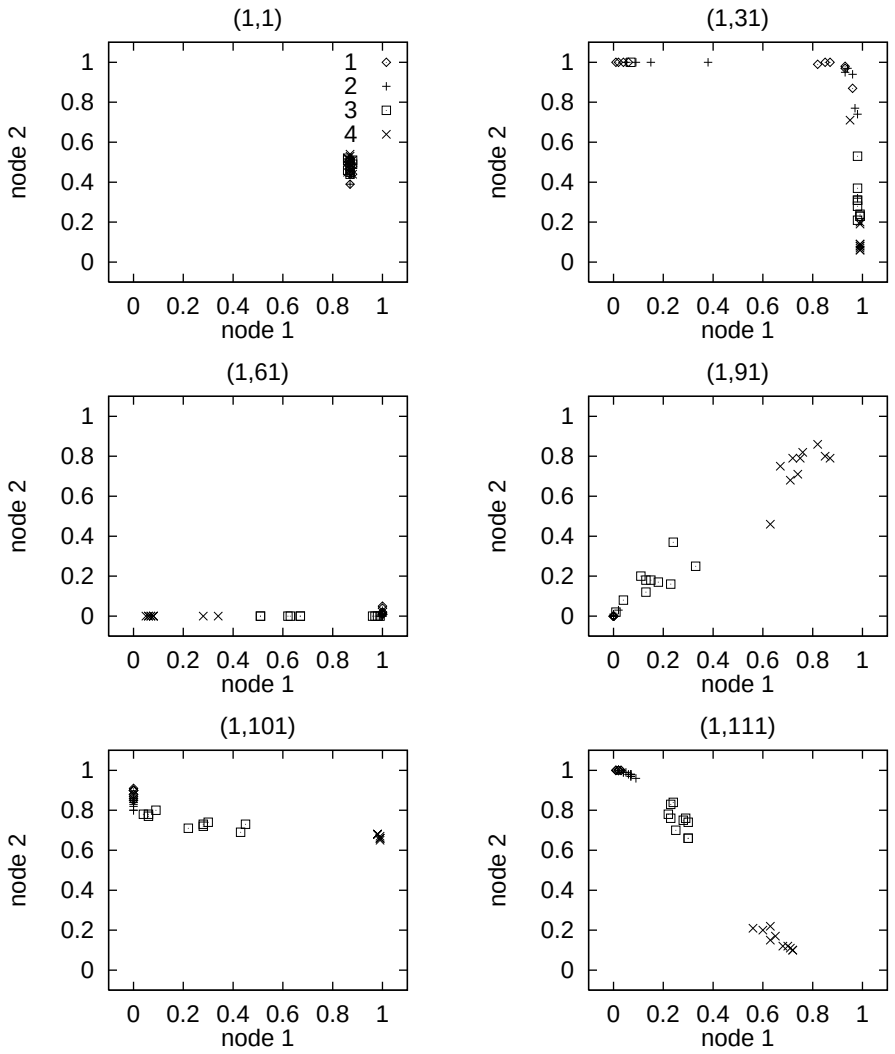

FIGURE 4.2.7. Plots of the representation's output for $(1, m)$ samples from the first experiment.

all three simulations). Qualitatively the curves support the theoretical conclusion that the representation error should decrease with an increasing number of tasks n in the (n, m) sample. However, note that the representation error is very small, even when the representation is derived from learning only one function from the environment. This can be explained as follows. For a representation to be a good one for learning in this environment it must be translationally invariant and distinguish all the four objects it sees (i.e. have different values on all four objects). For small values of n, to achieve perfect generalisation the representation is forced to be translationally invariant and so half of the problem is already solved. However, depending upon the particular functions in the (n, m) sample, the representation may not have to distinguish all four objects, for example it may map two objects to the same element of V if

Layer 1

Node	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_T
0	-7.7	0.8	-7.0	-5.4	-3.2	-4.3	-1.8	-4.5	-4.8	-3.1	0.5
1	5.4	5.6	5.8	5.3	5.7	6.0	5.6	5.5	5.5	5.5	-16.8
2	-9.2	-13.1	-8.4	-10.8	-8.8	-8.0	-9.6	-8.8	-9.3	-9.2	10.5

Layer 2

Node	w_0	w_1	w_2	w_T
0	-2.3	20.0	-20.0	-16.2
1	-5.9	15.4	-17.6	-0.3

TABLE 4.2.1. Weight values for the representation in figure 4.2.10.

none of the functions in the sample distinguish those objects (a function distinguishes two objects if it has a different value on those two objects). However, because the representation network is continuous it is very unlikely that it will map different objects to *exactly* the same element of V—there will always be slight differences. When the representation is used to learn a function that does distinguish the objects mapped to nearly the same element of V, often an output network g with sufficiently large weights can be found to amplify this difference and produce a function with small error. This is why a representation that is simply translationally invariant does quite well in general. This argument is supported by a plot in figure 4.2.7 of the representation's behaviour for $(n, m) = \{(1, 1), (1, 31), (1, 61), (1, 91), (1, 101), (1, 111)\}$ for the first simulation. The four different symbols marking the points in the plots correspond to the four different input objects. The ordinary generalisation error of the network for $m = 91, 101$ and 111 was zero. For the $(1, 91)$ plot the three and four pixel objects are well separated by the representation while the one and two pixel objects are not separated at all (they are both crammed into the bottom left-hand corner of the plot). This representation will not be a good one for learning any function that distinguishes the one and two pixel objects. A similar conclusion holds for the $(1, 101)$ and $(1, 111)$ samples, except that a closer look reveals that there is a slight separation between the representation's output for the one and two pixel objects. This separation can be exploited to learn a function that distinguishes the two objects.

For comparison, the behaviour of representations learnt on $(5, m)$ and $(21, m)$ samples is shown in figure 4.2.8. For large enough m these representations do distinguish well all four objects.

The behaviour of a typical perfect representation is shown pictorially in figure 4.2.10, and its weights in table 4.2.1. Notice that the representation is effectively operating as an "object detector" for the four objects in the environment.

Representation vs. No Representation. The other principal purpose of learning a representation is that it should greatly reduce the number of examples required to learn future tasks from the same environment. This was experimentally verified by taking a representation f known to be perfect for the environment in experiment one, and using it to learn all the functions in the environment in turn. Hence the

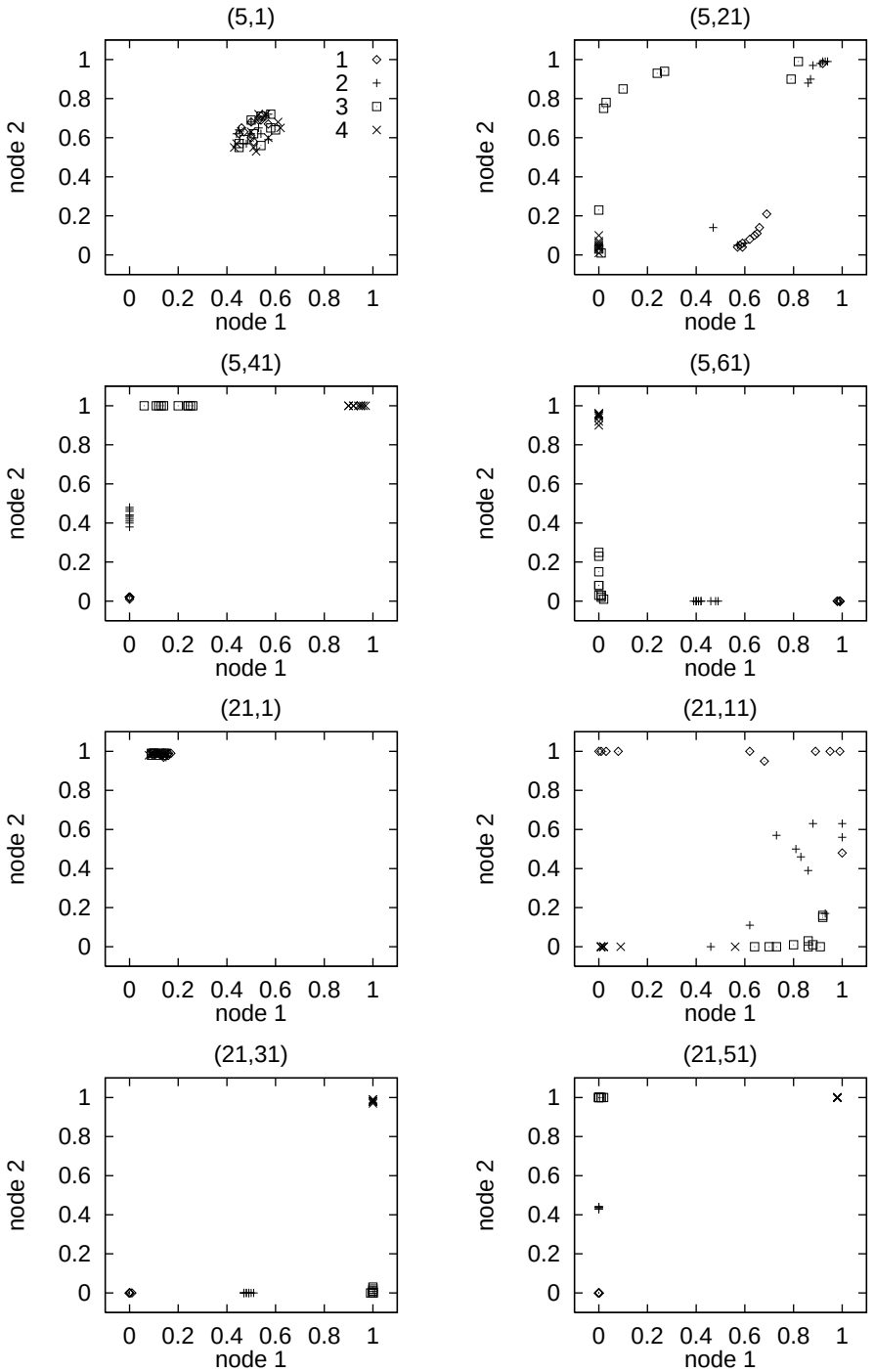

FIGURE 4.2.8. Plots of the representation's output for $(5, m)$ and $(21, m)$ samples from the first experiment.

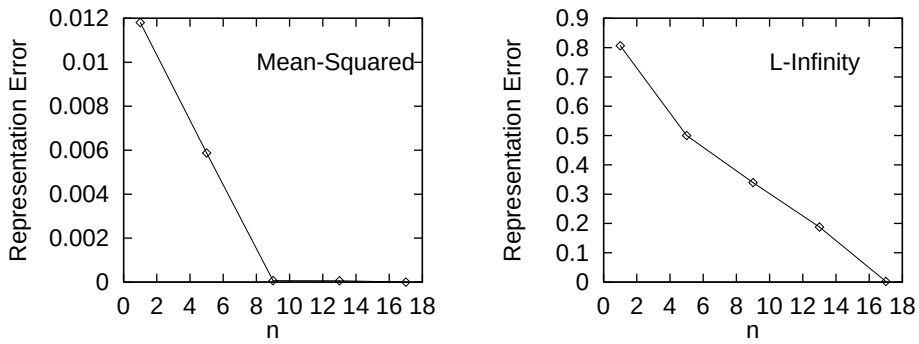

FIGURE 4.2.9. Mean-Squared and L^∞ representation error curves for the first experiment.

hypothesis space of the learner was $\mathcal{G} \circ f$, rather than the full space $\mathcal{G} \circ \mathcal{F}$. By way of comparison all the functions in the environment were also learnt using the full space. The generalisation curves (i.e. the generalisation error as a function of the number of examples in the training set) were calculated for all 14 functions in each case. The generalisation curves for all the functions were very similar; the first four are plotted in figure 4.2.11, for learning with a representation (Gof in the graphs) and without (GoF). The full curves for learning without a representation are also plotted in figure 4.2.12. These curves are the average of 32 different simulations obtained by using different random starting points for the weights in g (and f when using the full space to learn). Learning with a good representation is clearly far superior to learning without.

4.2.4. Experiment Two: Learning Symmetric Functions. The second neural network representation learning experiment was very similar to the first, the only difference being in the architectures of the networks and in the environment. This time the environment was chosen to consist of a subset of all the *symmetric* Boolean functions on 10 variables. The symmetric Boolean functions are those invariant under changes in the order of their inputs, or equivalently, the functions that depend only upon the number of "1's" in the input, not where the "1's" are located. The number of different input vectors seen by the learner was also restricted to consist only of those containing between one and four "1's". The probability distribution P on the input space was the same for all functions in the environment: input vectors were selected by firstly choosing uniformly a number from 1 to 4, and then placing that many active pixels randomly within the input vector. Once again the two trivial Boolean functions were excluded so that the environment consisted of 14 functions in all. Clearly, a representation exists for this environment—one that simply counts the number of active pixels in its input will do.

The neural networks in the representation space $\mathcal{F}$ had no hidden layers, ten input nodes and three output nodes. The output networks $\mathcal{G}$ also had no hidden layers, three input nodes (matching the number of the output nodes for networks in $\mathcal{F}$) and one output node. The networks were trained on (n, m) samples with n ranging from 1 to 21 in steps of four and m ranging from 1 to 171 in steps of 10. The same training and testing regime was followed as in the previous experiment. The (mean-squared) generalisation error as a function of m and n is plotted in figure 4.2.13 for three

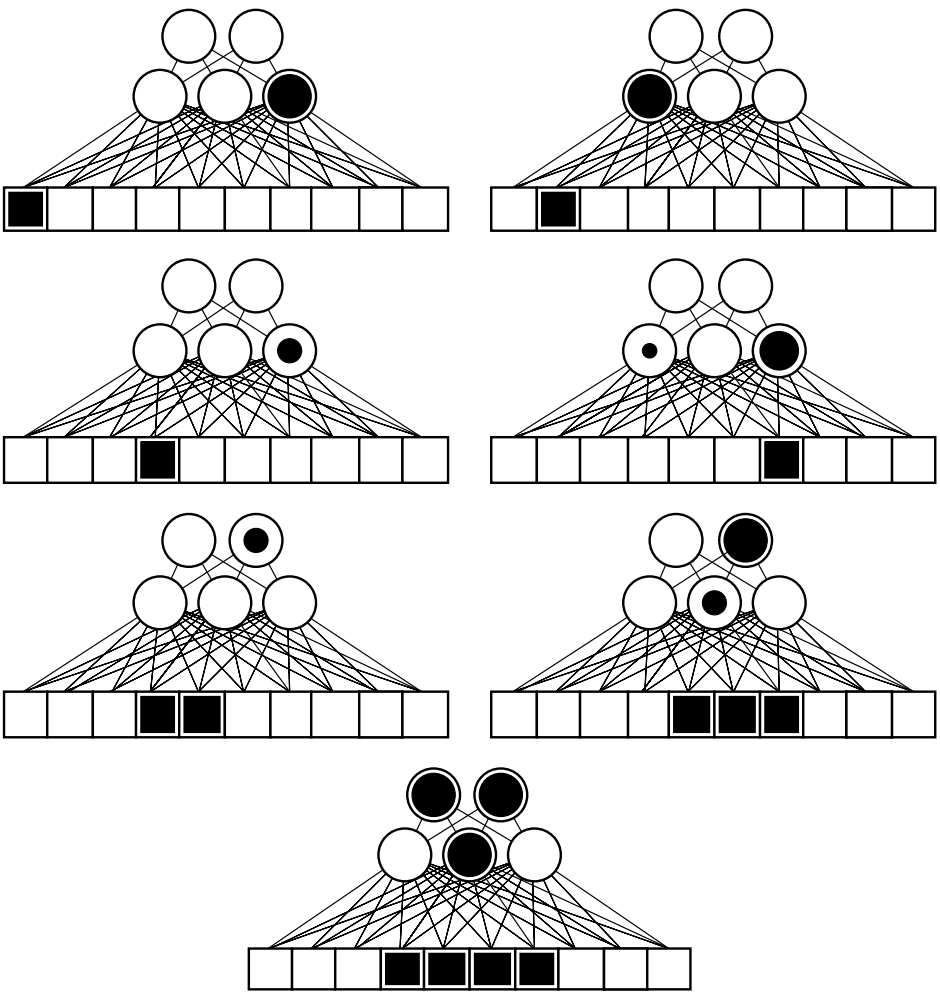

FIGURE 4.2.10. Behaviour of a perfect representation for the first experiment. The representation's output node activations were invariant under translations in the input object (to within about 1%). The hidden layer activations were also invariant for the two, three and four pixel objects, and there were four different hidden layer activation vectors for the one pixel object, depending on the position of the pixel in the input. Representative input vectors and the corresponding activations are shown here. This representation was the result of learning a $(21, 21)$ sample.

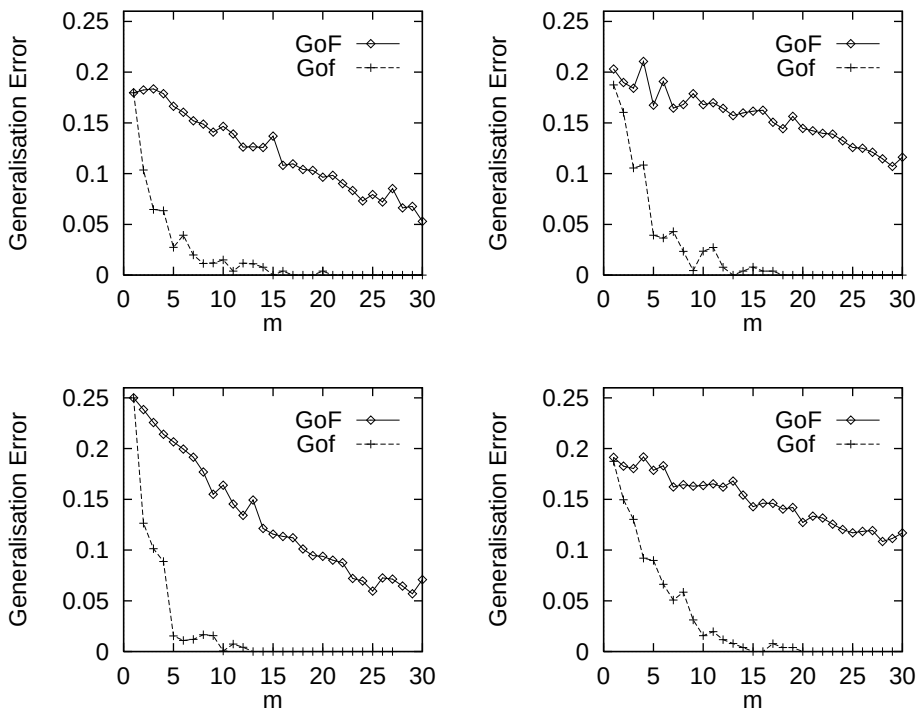

FIGURE 4.2.11. Generalisation curves for learning with a representation (Gof) vs. learning without (GoF) for the first four functions from the environment in the first experiment.

independent simulations. Once again these graphs clearly show that fewer examples are required for good generalisation if many functions are learnt simultaneously from the same environment.

As in the first experiment, (n, m) samples resulting in networks with perfect generalisation had their representations tested, and it was found that no more than five functions needed to be sampled from the environment before a perfect representation was learnt. Plots of the mean-squared and L^∞ representation error are given in figure 4.2.14. The remarkably low representation error has essentially the same explanation as in the previous experiment. The behaviour of a typical perfect representation is shown in figure 4.2.15. Note that the representation maps the input vectors into points in R^3 in such a way that all the functions in the environment can be implemented as linearly separable maps from R^3 into $\{0, 1\}$ (look at the cube in figure 4.2.15—any subset of the four labelled vertices can be sliced off with a plane). Given that the networks in $\mathcal{G}$ are only zero-hidden-layer "perceptrons", the representation is forced to behave in this way if it is to be a useful one for learning all functions from the environment.

Again, the learning ability of a network using a perfect representation was compared with a network that had to learn with the full space, and the generalisation curves of both for four different functions from the environment are plotted in figure 4.2.16. The full generalisation curves for learning with the full space $\mathcal{G} \circ \mathcal{F}$ are also shown in figure 4.2.17. These curves were generated in the same manner as the cor-

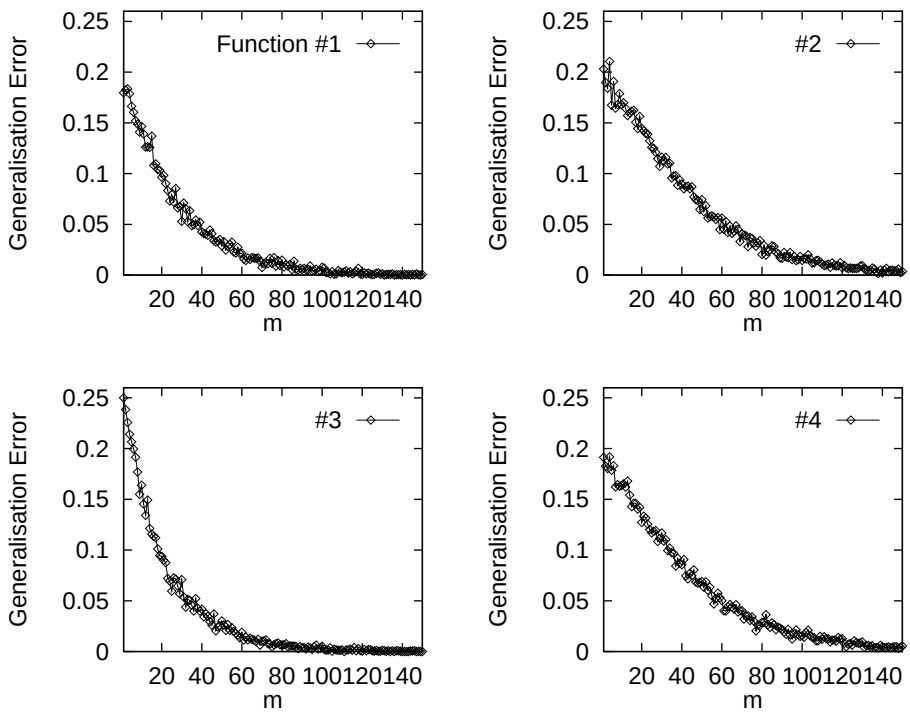

FIGURE 4.2.12. Full generalisation curves for learning without a representation for the first four functions from the environment of the first experiment.

responding curves in experiment one. Learning with a representation is clearly far better than learning without.

4.2.5. Local Minima. The number of restarts needed (or equivalently the number of local minima encountered) in typical simulations from both experiments are plotted as a function of n and m in figures 4.2.18 and 4.2.19. Because the CM5 used for these experiments had 32 nodes, all simulations were automatically started with 32 different random initial weight vectors. The number of restarts shown in the graphs do not include the initial 32.

The behaviour of the first experiment is not particularly illuminating. Essentially all that can be concluded from the graphs in figure 4.2.18 is that the number of local minima tended to increase monotonically with the number of examples. However the behaviour of the second experiment has an interesting interpretation. I have no explanation for why the two experiments behaved so differently.

Observe for the second experiment that the number of restarts required increased fairly monotonically with m until the value of m corresponding to the transition to perfect generalisation was reached (compare figure 4.2.13), at which point the graph drops rapidly back to zero. This behaviour can be understood as follows: firstly, the ease of training for values of m far below and far above the transition to perfect generalisation indicates that for these values of m there are mostly global, not local, minima on the error surface. For values of m on the upside of the transition from

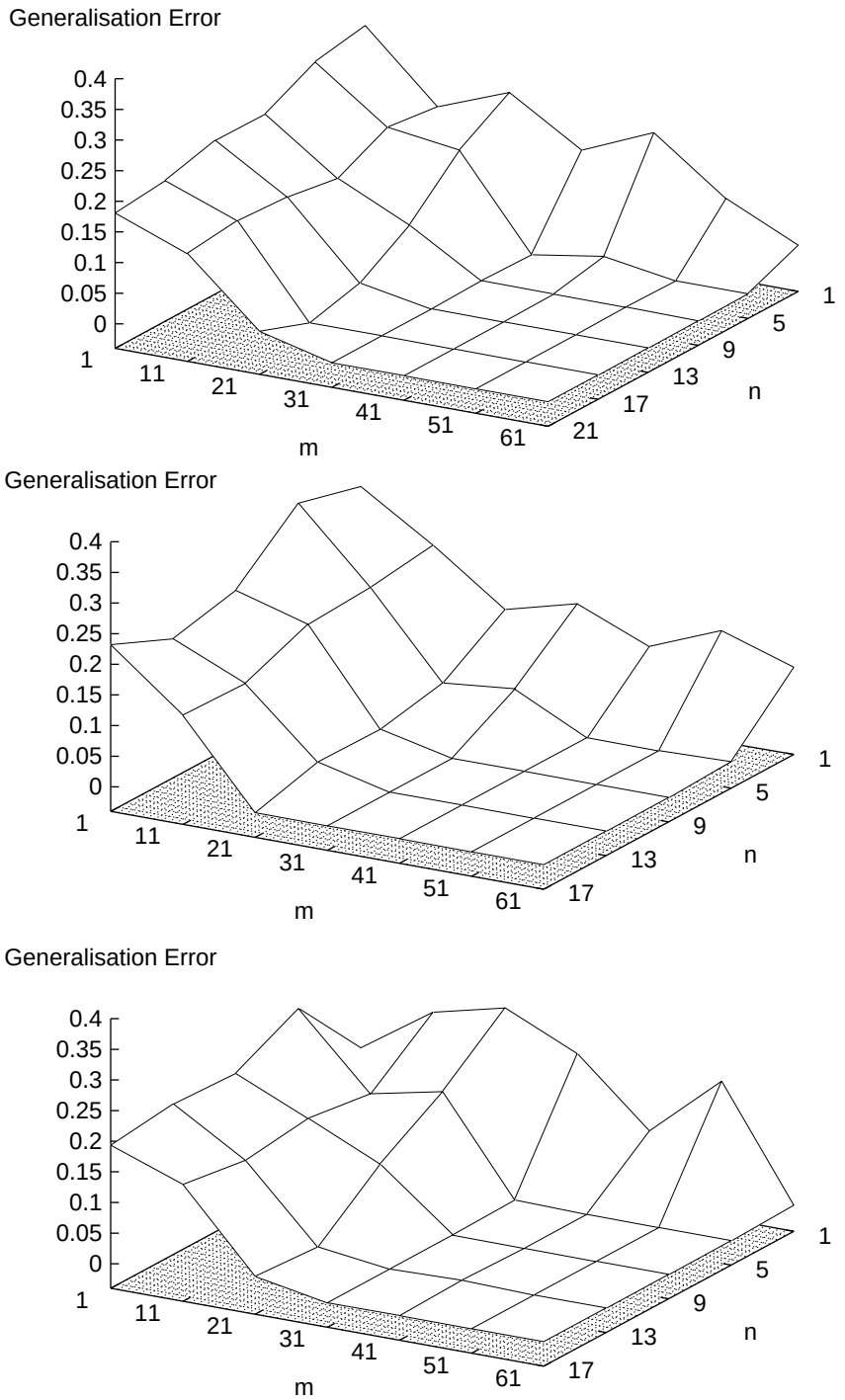

FIGURE 4.2.13. Generalisation error as a function of n—the number of functions learnt from the environment—and m—the number of examples per function, for three independent simulations from the second experiment.

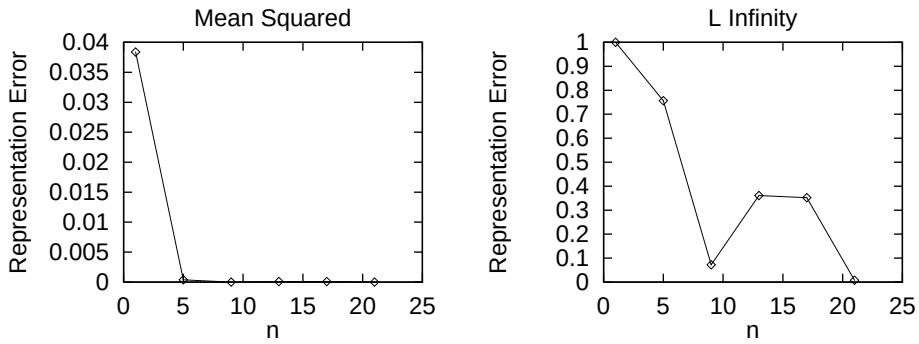

FIGURE 4.2.14. Mean-squared and L^∞ representation error curves for the second experiment.

poor to good generalisation, the fact that all solutions turn out to be true solutions to the problem indicates that most of the global minima correspond to true solutions. These points are clearly also global minima for values of m on the lower side of the transition point. However as most solutions for small values of m turn out to have poor generalisation, these global minima must constitute only a small fraction of all the global minima on the error surface—the other ones being points with low error on the training set but poor generalisation ability. As m increases it is these points of poor generalisation that must discontinue being global minima of the error surface, i.e. their error values must "lift". Thus they become local minima for m around the transition point and eventually get ironed out of the error surface altogether at high enough values of m. This explains the preponderance of local minima around the transition point. With such a large number of deep local minima it takes many restarts before the initial weight values (by chance) land in the attracting basin of a global minima.

As a final point, part of the cause of the local minima in these representation learning simulations is no doubt the fact that error information for f is filtered first by the output functions g. It will be shown in chapter 6 how this can be avoided in the case that the environment of learner consists of classification problems.

4.3. Zip-Code Network

One of the more famous examples of a successful application of neural networks is the network developed by Le Cun et. al. (1989) for classification of handwritten digits in zip-codes. The final hidden layer of their network consisted of 30 nodes and these were fully connected to 10 output nodes, one for recognising each of the digits. Within the framework of representation learning discused here, the final hidden layer of their network can be viewed as the output of a representation, while the 10 output nodes (and their respective incoming weights) can be viewed as 10 output functions $g_1, \ldots, g_{10}$. It would be interesting to extract the representation part of the network and use it to learn some new characters, such as 'A', 'B', etc. My guess is that the network's ability to learn new characters will be quite limited. However if it *can* successfully learn many new characters using the existing representation then that would be evidence *against* the thesis being advocated here that the information necessary to learn a good representation is spread across many examples of similar

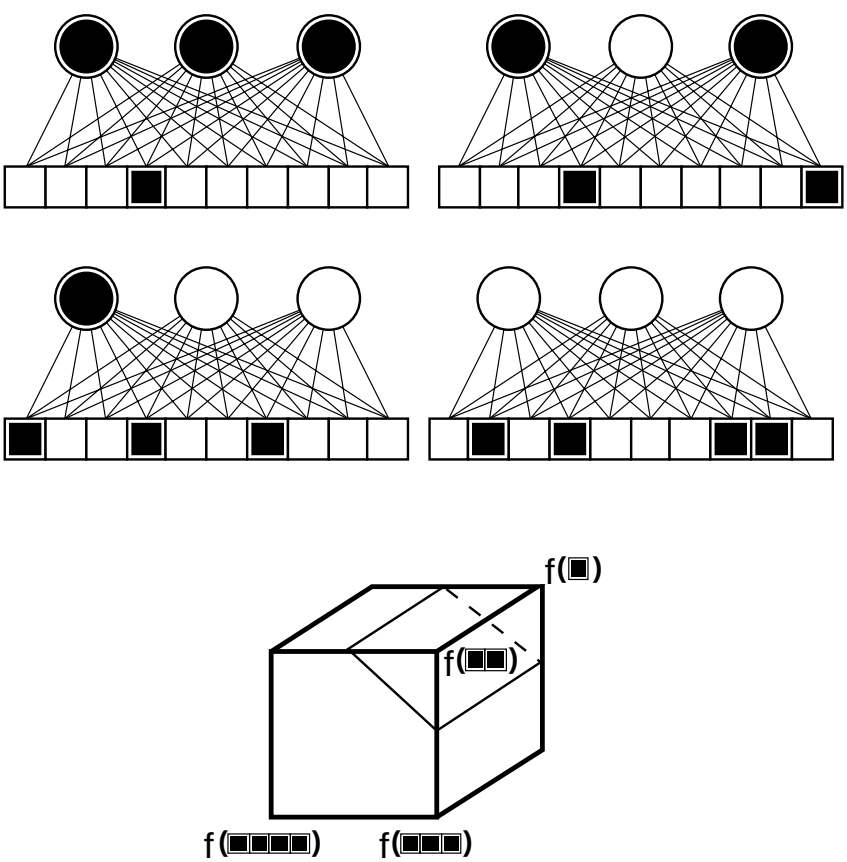

FIGURE 4.2.15. Behaviour of a perfect representation f for the second experiment. The output layer of f is invariant (within about 1%) with respect to changes in the order of its inputs. Four example input vectors and the corresponding output vectors are shown here. The cube illustrates how any function from the environment can be implemented as a linearly separable map applied to the output of f (one example is shown).

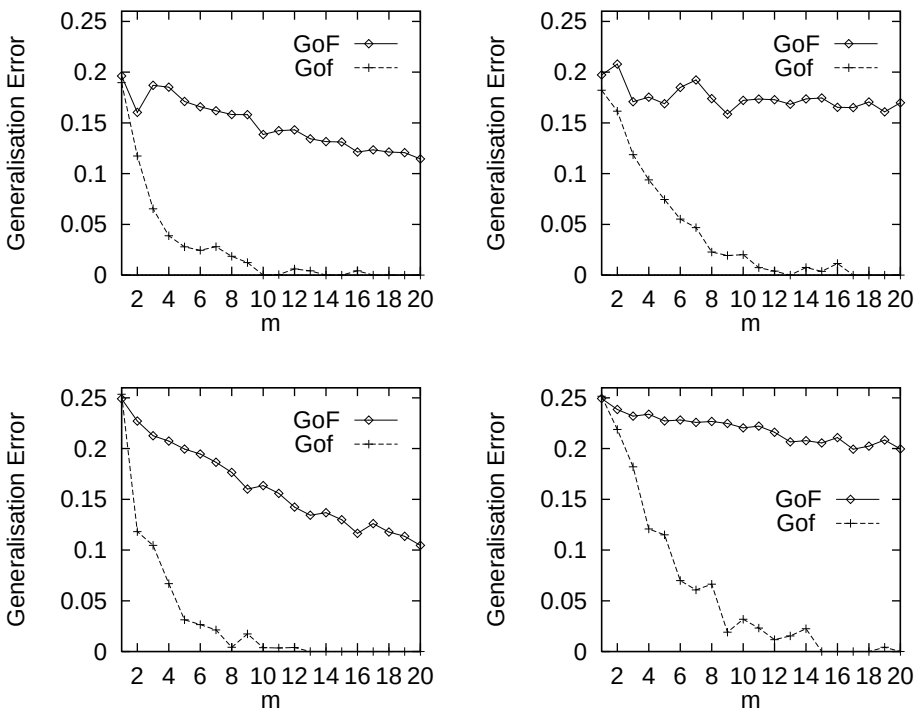

FIGURE 4.2.16. Generalisation curves for learning with a representation (Gof) vs. learning without (GoF) for four different functions from the environment in the second experiment.

learning tasks. On the other hand, if the network is incapable of learning a significant number of new characters with the existing representation then it would be interesting to see if the network's *architecture* is sufficient to learn a good representation by training it on many more character classification problems than just the digits (this can be achieved by simply adding an extra output node to the network for each new character and training with normal backpropagatation). If the theory here is correct, the number of examples of each character required should decrease as the number of characters increases. The other two principles of representation learning (speed and effectiveness of learning new characters using a good representation) could also be tested.

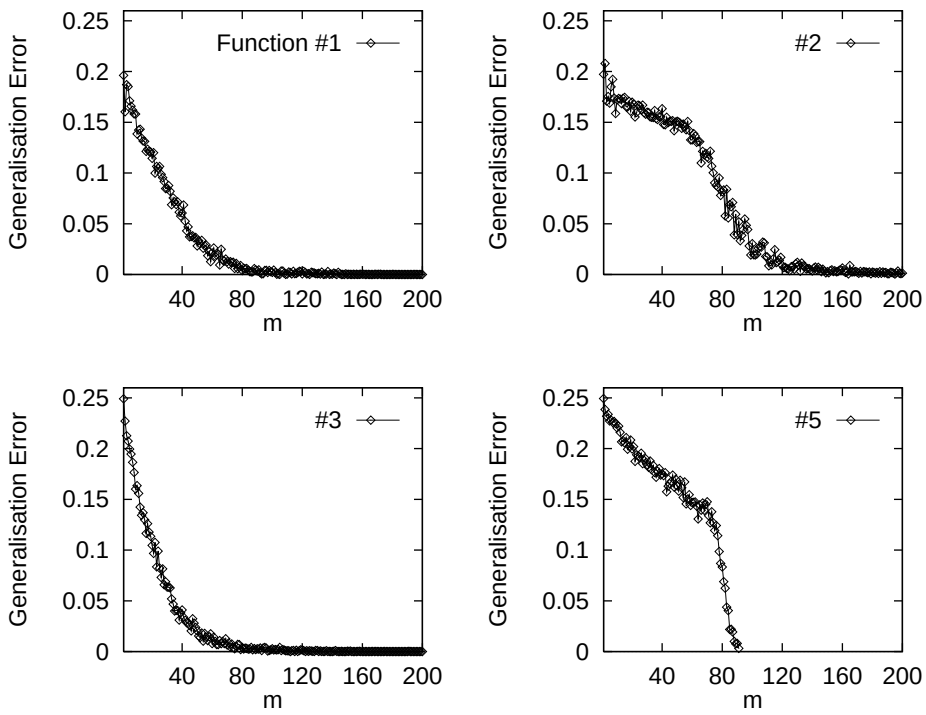

FIGURE 4.2.17. Full generalisation curves for learning without a representation for four different functions from the environment in the second experiment.

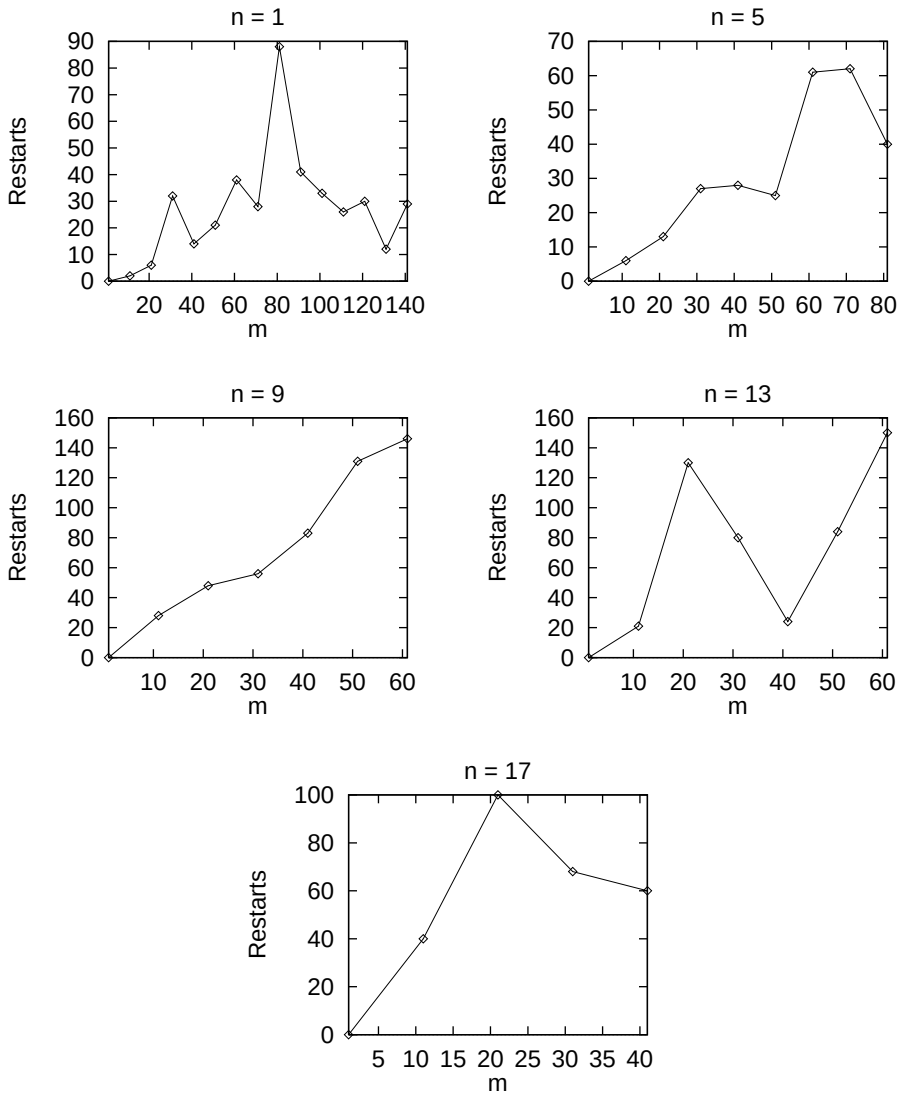

FIGURE 4.2.18. Restarts needed (or local minima encountered) before finding a solution for a typical simulation from the first experiment.

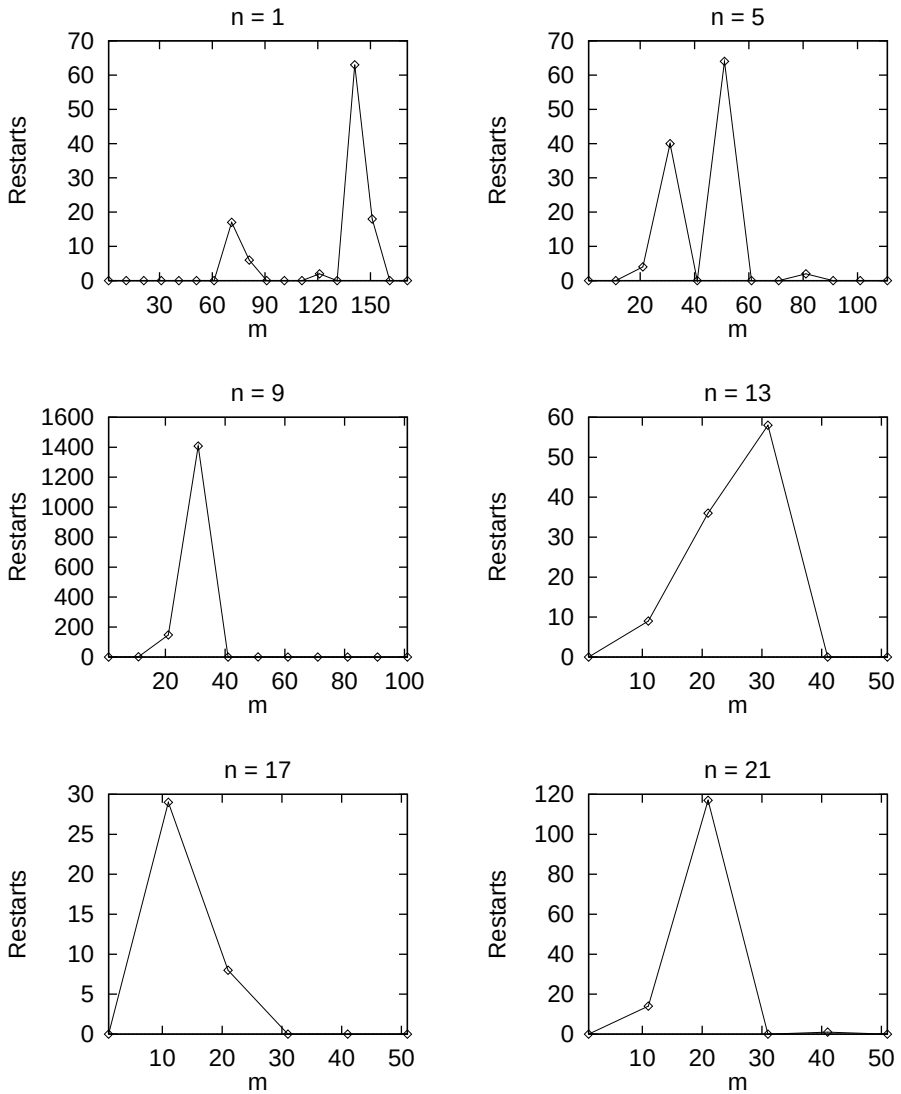

FIGURE 4.2.19. Restarts needed (or local minima encountered) before finding a solution for a typical simulation from the second experiment.

CHAPTER 5

Vector Quantization

In this chapter it is shown how the idea of the *environment* of a learning process can be applied to the problem of vector quantization and how this leads to a natural choice for the distortion measure in the quantization process. This distortion measure turns out to be optimal in a very general sense.

5.1. The Problem

As any real communication channel has only finite capacity, transmitting continuous data (e.g speech signals or images) requires firstly that it be transformed into a discrete representation. Typically, given a probability space (X, P, σ_X), one chooses a *quantization* of X, $\{x_1, \ldots, x_k\} \subset X$ and instead of transmitting $x \in X$, the index of the "nearest" quantization point $q_d(x) = \min_{x_i} d(x_i, x)$ is transmitted, where d is some function (not necessarily a metric) measuring the distance between points in X. d is called a *distortion measure*. The quantization points are chosen so that the expected distance between x and its quantization is minimal, i.e $\vec{x} = \{x_1, \ldots, x_k\}$ are chosen to minimize the *reconstruction error*,

$$E_d(\vec{x}) = \int_X d(x, q_d(x))\, dP(x). \tag{40}$$

There are simple methods, for iteratively generating good quantization sets $\{x_1, \ldots, x_k\}$ (See Cover and Thomas (1991) chapter 13 and references therein).

Some examples of distortion measures are the *Hamming metric*,

$$d(x, x') = 1 - \delta(x, x'),$$

where δ is the Kronecker delta function, and the *squared Euclidean distortion* for vector-valued X,

$$d(x, x') = \|x - x'\|^2.$$

The use of these distortion measures has more to do with their convenient mathematical properties than their applicability to a particular problem domain. For example, suppose X is a space of images and it is images of characters that are being transmitted over the channel. An image of the character "A" and another translated image of the same character would be considered "close" in this context, although the squared Euclidean distance between the two images would be large, quite likely larger than the distance between an image of "A" and an image of "B" in the same location.

Thus the Euclidean distortion measure does not at all capture the idea of two images being "close" in this context. Another example is that of speech coding—there is a large squared Euclidean distance between a speech signal and a small translation of it in time, although both sound very similar to a human observer. Although improved distortion measures based on entropy considerations have been found, the situation is still quite unsatisfactory. To quote from Cover and Thomas (1991), page 340,

> In image coding, however, there is at present no real alternative to using the mean squared error as the distortion measure.

The next section shows how this problem may be solved, at least *in principle*, by using the idea of the *environment* of a learning process.

5.2. The Solution

What makes the translated image of an "A" close to the original, while an untranslated image of a "B" quite distant? And what makes two speech signals nearly identical even though they are miles apart from a Euclidean perspective? In both cases it is because there is an *environment* of classifiers that behave as if the distinct images of "A" are close, and as if the speech signals are close. For example, although there is nothing *intrinsic* about "A" and its translate that makes them close, there is something about the set of all character classifiers that makes them close. A classifier for the character "A" is just a function $f_A\colon X \to \{0,1\}$ such that if x is an image of an "A" then $f_A(x) = 1$ and if not, then $f_A(x) = 0$. Classifiers for "B", "C", "#" and any other character-like objects similarly exist in the environment of X. All these classifiers have the property that they are *constant* across images of the same character, either "0" if the character is not theirs, or "1" if it is. Thus it is this space of classifiers that determines what makes images of characters close together. Similarly for speech signals. There exist word classifiers in the environment of the space of all speech signals that are constant across those sets of signals corresponding to a particular word. The word classifiers induce the distance measure on the speech signals.

In order to formalize this idea, suppose that the *environment* of a probability space (X, P, σ_X) consists of a set of functions $\mathcal{F}$ mapping X into a space (Y, σ), where $\sigma\colon Y \times Y \to \mathrm{R}$ (σ may be a metric), and an *environmental probability measure* Q on $\mathcal{F}$. An environment so defined induces the following natural distortion measure on X,

$$\rho(x,y) = \int_{\mathcal{F}} \sigma(f(x), f(y))\, dQ(f), \tag{41}$$

for all $x, y \in X$. Note that if σ is a metric on Y then ρ is a pseudo-metric on X. In relation to the character transmission problem, $\mathcal{F}$ would consist of all character-like classifiers, Y would be the set $\{0,1\}$ and σ would be the Hamming metric. Then if x and y are two images of the same character, $f(x) = f(y)$ for all $f \in \mathcal{F}$ and so $\rho(x,y) = 0$, as required. If x and y are images of different characters, two of the functions in $\mathcal{F}$ would have $f(x) \neq f(y)$ which would give $\rho(x,y) > 0$, assuming the functions have positive probability under Q. Thus ρ behaves exactly as desired for a distortion measure in this context. Note that ρ depends only upon the environment $(\mathcal{F}, Q)$ and not upon X or its probability measure P. Thus problems with the same X but different environments (for example character classification and face recognition—different environments for the space of images) will generate different

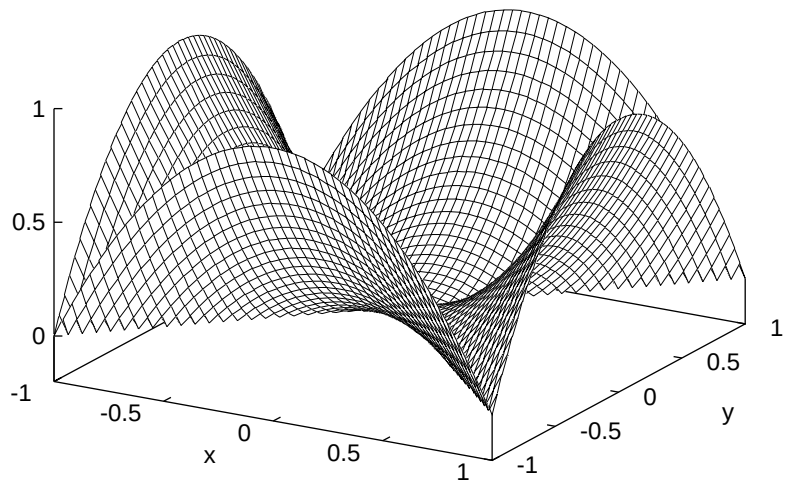

FIGURE 5.2.20. $\rho(x, y)$ for a quadratic environment.

distortion measures. The following pair of simple examples further illustrate the behaviour of ρ as a distortion measure.

EXAMPLE 5.1. Let $X = Y = [0, 1]$, $\sigma(y, y') = |y - y'|$ for all $y, y' \in Y$ and $\mathcal{F}$ be the set of all linear functions mapping $[0, 1]$ into $[0, 1]$ with slope a uniformly distributed also in the range $[0, 1]$. Thus $\mathcal{F} = \{f : x \mapsto ax : a \in [0, 1]\}$. For any $x, y \in X$,

$$\begin{aligned}\rho(x, y) &= \int_{\mathcal{F}} \sigma(f(x), f(y))\, dQ(f) \\ &= \int_0^1 |ax - ay|\, da \\ &= \frac{1}{2}|x - y|.\end{aligned}$$

Thus $\mathcal{F}$ induces the Euclidean metric on $[0, 1]$ in this case.

EXAMPLE 5.2. Let $X = Y = [-1, 1]$, $\sigma(y, y') = |y - y'|$ for all $y, y' \in Y$ and $\mathcal{F} = \{f : x \mapsto ax^2\}$ with a uniformly distributed in the range $[0, 1]$ again. This time,

$$\begin{aligned}\rho(x, y) &= \int_{-1}^1 |ax^2 - ay^2|\, da \\ &= |x - y||x + y|.\end{aligned}$$

ρ is plotted in figure 5.2.20. Note that $\rho(x, y) = 0$ if $x = y$ and if $x = -y$, so that x and $-x$ are zero distance apart under ρ. This reflects the fact that $f(x) = f(-x)$ for all $f \in \mathcal{F}$. Notice also that $\rho(x, y)$ is the ordinary Euclidean distance between x and

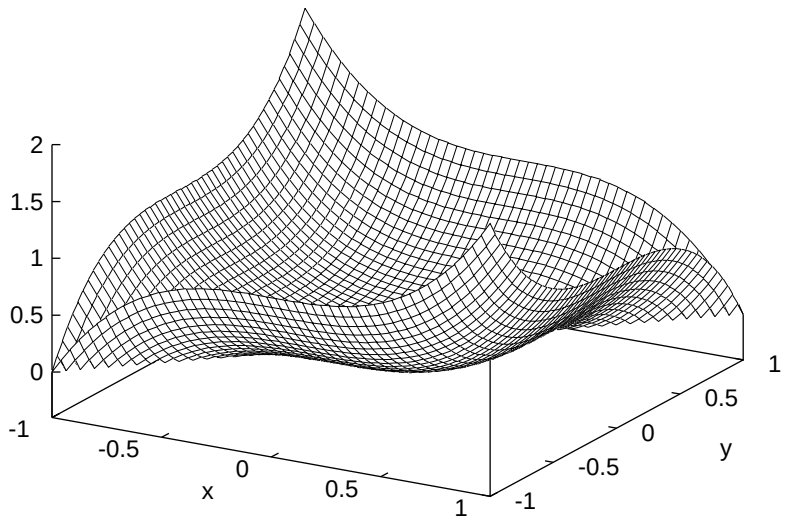

FIGURE 5.2.21. $\rho(x, y)$ for a cubic environment.

y, *scaled* by $|x + y|$. Thus two points with fixed Euclidean distance become further and further apart under ρ as they are moved away from the origin. This reflects the fact that the quadratic functions in $\mathcal{F}$ have larger variation in their range around large values of x than they do around small values of x. This can also be seen by calculating the ε-ball around a point x under the ρ metric (i.e the set of points $x' \in X$ such that $\rho(x, x') \leq \varepsilon$). To first order in $\frac{\varepsilon}{x}$ this is

$$\left[-x - \frac{\varepsilon}{2x}, -x + \frac{\varepsilon}{2x}\right] \bigcup \left[x - \frac{\varepsilon}{2x}, x + \frac{\varepsilon}{2x}\right].$$

Note that the Euclidean diameter of the ε-ball around x decreases linearly with x's—Euclidean again—distance from the origin. Although not as illuminating as the metric for a quadratic environment, the metric for a cubic environment is particularly attractive and so is plotted in figure 5.2.21.

5.3. Optimal Function Approximation

In this section it is shown that ρ is the optimal distortion measure to use in vector quantization if the goal is to find piecewise constant approximations to the functions in the environment.

Piecewise constant approximations to $f \in \mathcal{F}$ are generated by specifying a quantization $\vec{x} = \{x_1, \ldots, x_k\}$ of X and a partition $\vec{X} = \{X_1, \ldots, X_k\}$ of X that is *faithful* to $\{x_1, \ldots, x_k\}$ in the sense that $x_i \in X_i$ for $1 \leq i \leq k$. The piecewise constant approximation $\hat{f}$ to any function f is then defined by $\hat{f}(x) = f(x_i)$ for all $x \in X_i$, $1 \leq i \leq k$. The most natural way to measure the deviation between f and $\hat{f}$ in this

context is with the pseudo-metric d_P,

$$d_P(f, \hat{f}) = \int_X \sigma(f(x), \hat{f}(x))\, dP(x).$$

$d_P(f, \hat{f})$ is the expected difference between $f(x)$ and $\hat{f}(x)$ on a sample x drawn at random from X according to P. The *reconstruction error* of $\mathcal{F}$ with respect to the pair $\vec{x} = \{x_1, \ldots, x_k\}$ and $\vec{X} = \{X_1, \ldots, X_k\}$ is defined to be the expected deviation between f and its approximation $\hat{f}$, measured according to the distribution Q on $\mathcal{F}$:

$$E_{\mathcal{F}}(\vec{x}, \vec{X}) = \int_{\mathcal{F}} d_P(f, \hat{f})\, dQ(f).$$

The quantization and partition should be chosen so as to minimize $E_{\mathcal{F}}$.

Given any quantization $\vec{x} = \{x_1, \ldots, x_k\}$ and distortion measure ρ as in equation (41), define the partition $\vec{X}^\rho = \{X_1^\rho, \ldots, X_k^\rho\}$ by $X_i^\rho = \{x \in X : \rho(x, x_i) \leq \rho(x, x_j), \text{for all} j \neq i\}$ (break any ties by rolling an appropriately sided dice). Call this the partition of X *induced* by ρ and $\vec{x}$.

THEOREM 5.1. *The reconstruction error $E_{\mathcal{F}}$ of $\mathcal{F}$ is minimal with respect to a quantization $\vec{x} = \{x_1, \ldots, x_k\}$ minimizing the reconstruction error $E_\rho(\vec{x})$ of X, and the partition $\vec{X}^\rho$ induced by ρ and $\{x_1, \ldots, x_k\}$.*

PROOF. This is proved in two steps. First it is shown that if $\vec{x} = \{x_1, \ldots, x_k\}$ is any quantization of X, then the partition $\vec{X}$ minimizing $E_{\mathcal{F}}(\vec{x}, \vec{X})$ is the one induced by ρ. The proof is completed by showing that $E_{\mathcal{F}}(\vec{x}, \vec{X}^\rho) = E_\rho(\vec{x})$.

Thus, let $\vec{x} = \{x_1, \ldots, x_k\}$ be any quantization of X and let $\vec{X}^\rho = \{X_1^\rho, \ldots, X_k^\rho\}$ be the corresponding partition induced by ρ. Denote the approximation of $f \in \mathcal{F}$ with respect to this partition by $\hat{f}_\rho$. Let $\vec{X} = \{X_1, \ldots, X_k\}$ be any other partition of X that is faithful to $\vec{x}$ and let $\hat{f}$ denote the approximation of f with respect to this second partition. Define $X_{ij} = X_i \cap X_j^\rho, 1 \leq i \leq k, 1 \leq j \leq k$. Note that the X_{ij}'s are also a partition of X. The reconstruction error of $\mathcal{F}$ with respect to the partition $\vec{X}$ satisfies:

$$\begin{aligned}
E_{\mathcal{F}}(\vec{x}, \vec{X}) &= \int_{\mathcal{F}} d_P(f, \hat{f})\, dQ(f) \\
&= \int_{\mathcal{F}} \int_X \sigma(f(x), \hat{f}(x))\, dP(x)\, dQ(f) \\
&= \sum_{i,j=1}^{k} \int_{X_{ij}} \int_{\mathcal{F}} \sigma(f(x), f(x_i))\, dQ(f)\, dP(x) \\
&= \sum_{i,j=1}^{k} \int_{X_{ij}} \rho(x, x_i)\, dP(x) \\
&\geq \sum_{i,j=1}^{k} \int_{X_{ij}} \rho(x, x_j)\, dP(x) \\
&= \int_{\mathcal{F}} d_P(f, \hat{f}_\rho)\, dQ(f) \\
&= E_{\mathcal{F}}(\vec{x}, \vec{X}^\rho),
\end{aligned}$$

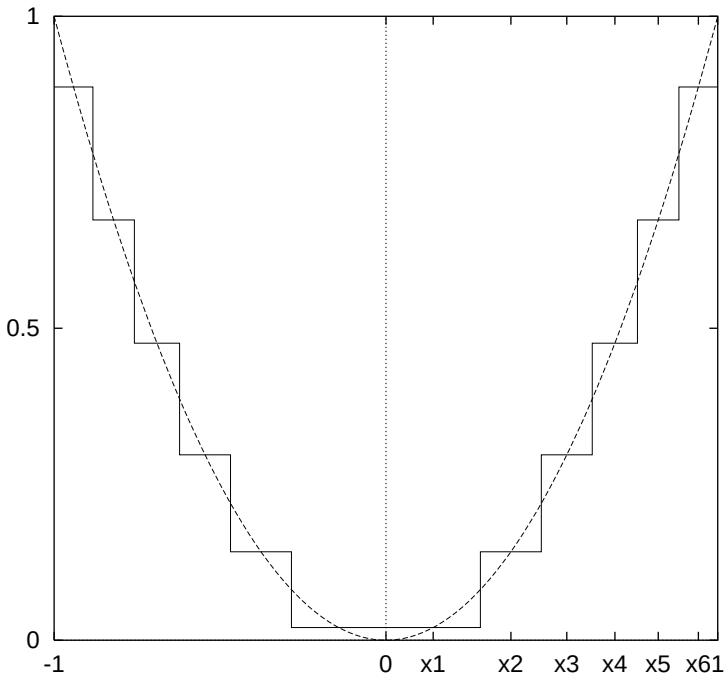

FIGURE 5.3.22. Optimal quantization points and the corresponding approximation to $f(x) = x^2$ for the quadratic environment of example 5.2. The six quantization points are (to three significant figures) $x_1 = 0.142$, $x_2 = 0.377$, $x_3 = 0.545$, $x_4 = 0.690$, $x_5 = 0.821$, $x_6 = 0.942$.

and so the first part of the proof is established. Now observe that

$$\begin{aligned} E_{\mathcal{F}}(\vec{x}, \vec{X}^{\rho}) &= \int_{\mathcal{F}} dP(f, \hat{f}_{\rho})\, dQ(f) \\ &= \int_{\mathcal{F}} \int_{X} \sigma(f(x), \hat{f}_{\rho}(x))\, dP(x)\, dQ(f) \\ &= \int_{X} \int_{\mathcal{F}} \sigma(f(x), f(q_{\rho}(x)))\, dQ(f)\, dP(x) \\ &= \int_{X} \rho(x, q_{\rho}(x))\, dP(x) \\ &= E_{\rho}(\vec{x}), \end{aligned}$$

which completes the proof. □

EXAMPLE 5.3. For the quadratic environment of example 5.2, the optimal quantization for $k = 6$ is shown in figure 5.3.22 along with f and $\hat{f}_{\rho}$ for $f(x) = x^2$. Note how the optimal quantization reduces the deviation between f and its approximation by spacing its points closer together for larger values of x.

Even in such a simple environment as this one the optimal quantization is not so easy to calculate. First note by the symmetry of the environment that the quantization points $\{x_1, \ldots, x_k\}$ can all be assumed to be positive, and without loss of gener-

ality suppose that $x_1 \leq x_2 \leq \cdots \leq x_k$. The contribution of any triplet x_{i-1}, x_i, x_{i+1} to the quantization error $E_\rho(\vec{x})$ is

$$\int_{x_{i-1}}^{\sqrt{\frac{1}{2}\left(x_{i-1}^2+x_i^2\right)}} x^2 - x_{i-1}^2\, dx \;+\; \int_{\sqrt{\frac{1}{2}\left(x_{i-1}^2+x_i^2\right)}}^{x_i} x_i^2 - x^2\, dx \;+\; \int_{x_i}^{\sqrt{\frac{1}{2}\left(x_i^2+x_{i+1}^2\right)}} x^2 - x_i^2\, dx$$
$$+ \int_{\sqrt{\frac{1}{2}\left(x_i^2+x_{i+1}^2\right)}}^{x_{i+1}} x_{i+1}^2 - x^2\, dx.$$

Minimizing this expression with respect to x_i gives,

$$x_i^2 = \frac{1}{4}\left(x_{i-1}^2 + x_{i+1}^2\right) + \frac{1}{4\sqrt{2}}\sqrt{x_{i-1}^4 + 6x_{i-1}^2 x_{i+1}^2 + x_{i+1}^4}.$$

A similar procedure can be used to show that x_1 and x_k must satisfy,

$$x_1 = \frac{x_2}{\sqrt{7}},$$
$$x_k = \frac{4 + \sqrt{2 + 7x_{k-1}^2}}{7}.$$

Optimal quantization points can be found numerically by initialising $x_i = i/k$ for $1 \leq i \leq k$ and then iteratively updating the points in order according to these relations. This procedure is guaranteed to converge to the optimal solution or a limit cycle, although none of the latter were observed in any of the tests, probably because they do not exist in this case or have starting conditions of measure zero. An error of 1 part in a million was achieved in 36 iterations for the six points in figure 5.3.22.

5.4. Learning ρ

For most environments encountered in practise (e.g speech recognition or image recognition), ρ will be unknown. However it may be estimated or *learnt* in the same manner that a representation is learnt—by sampling from the environment. One method is to sample M times from $\mathcal{F}$ according to Q to generate $\{f_1, \ldots, f_M\}$ and then N times from X according to P to generate $\{x_1, \ldots, x_N\}$. For any pair x_a, x_b an estimate of $\rho(x_a, x_b)$ is given by

$$\hat{\rho}(x_a, x_b) = \frac{1}{M}\sum_{i=1}^{M} \sigma(f_i(x_a), f_i(x_b)).$$

This generates $\frac{N\times(N-1)}{2}$ training *triples*,

$$\{(x_1, x_2, \hat{\rho}(x_1, x_2)), \ldots, (x_{N-1}, x_N, \hat{\rho}(x_{N-1}, x_N))\},$$

which can be used as data to train, for example, a neural network. That is, the neural network would have two sets of inputs, one set for x_a and one set for x_b, and a real-valued output representing the network's estimate of $\hat{\rho}(x_a, x_b)$. The network would be trained to approximate $\hat{\rho}$. Although the theory is not developed here, a similar analysis to that given for representation learning could be applied in this situation

to generate estimates for the size of M required to ensure that $\hat{\rho}$ is close to ρ on the training set, and also bounds on the size of N ensuring good generalisation from the resulting neural network.

Another method for learning ρ is to use the result of a previous (n, m) learning process. That is, a hypothesis $\vec{g} \circ \vec{f}$ estimates ρ by

$$\hat{\rho}(x, y) = \frac{1}{n} \sum_{i=1}^{n} \sigma \left(g_i \circ f(x), g_i \circ f(y) \right) .$$

Note that this method would work using the output of any (n, m) learning process, not just a representation learner.

5.5. Discussion

It has been shown that the existence of an *environment* for a quantization process leads to a *canonical* distortion measure which turns out to be optimal if approximation of the functions in the environment is the aim. A couple of brief ideas were mentioned for learning the distortion measure by sampling from the environment.

If the environment consists of classification problems then it is fairly easy to see (and is demonstrated in the next chapter) that learning the distortion measure is equivalent to learning a representation for the environment that effectively solves all the classification problems. Thus, not surprisingly, finding a good distortion measure for environments such as speech recognition and image recognition is formally equivalent to solving the learning problem for those environments. This may not seem like much of a step forwards, as one unsolved problem has simply been converted into another unsolved problem (although the claim in this thesis is that representation learning should go part way towards solving the latter problem), however from a purely theoretical perspective it has its advantages because it offers a broader framework in which to tackle both problems, and also allows the derivation of the optimality property. In the next chapter it is shown how a marriage of the ideas of the current chapter with the representation learning ideas of earlier chapters leads to a much improved method for representation learning, and offers some hope for solving the problem of architecture selection in neural networks.

CHAPTER 6

Learning a Representation Directly

If the environment of the learner is of the form $X \xrightarrow{f} V \xrightarrow{\mathcal{G}} Y$ and the learner knows the form of V and $\mathcal{G}$, it is possible to directly learn a representation for the environment without learning the functions $g \in \mathcal{G}$ as well. This is to be contrasted with the procedure used in Chapter 4 in which the output functions g had to be learnt along with the representation. In this chapter it is shown how to do this for several different function spaces $\mathcal{G}$, and an experiment is presented demonstrating the effectiveness of the technique for a simple environment.

The work in this chapter is very much "in progress"—in fact the ideas only occurred to me whilst writing the previous chapter. However I believe it represents a promising direction for further investigation and so have included it despite its unfinished nature.

6.1. An error measure for the representation.

Let the environment of the learner be of the form $X \xrightarrow{f} V \xrightarrow{\mathcal{G}} Y$, let $\sigma : Y \times Y \to [0, M]$ be a loss function on $Y \times Y$ and P be a probability measure on X. Let Q be the environmental probability measure on $\mathcal{G}$. As the representation f is fixed, the distortion measure ρ (41) from the previous chapter can be written

$$\rho(x, y) = \int_{\mathcal{G}} \sigma\big(g(f(x)), g(f(y))\big)\, dQ(g),$$

for all $x, y \in X$. Define the distortion measure $\rho_{\mathcal{G}}$ on V by

$$\rho_{\mathcal{G}}(v, w) = \int_{\mathcal{G}} \sigma(g(v), g(w))\, dQ(g),$$

for all $v, w \in V$. Note that $\rho_{\mathcal{G}}(f(x), f(y)) = \rho(x, y)$ for all $x, y \in X$.

Suppose the learner knows what the spaces $\mathcal{G}$ and V are, and the measure Q, and is itself trying to learn a representation mapping X into V that is appropriate for the environment. Denote the learner's estimate for the representation by $\hat{f}$. To generate data for the learner, suppose X is sampled N times according to P to generate $(x_1, \ldots, x_N)$ and $\mathcal{G}$ is sampled M times according to Q to generate $(g_1 \circ f, \ldots, g_M \circ f)$. The learner is given the $M \times N$ pairs $(x_i, g_j \circ f(x_i))$ as a training set. Note that this is not an (M, N) sample. From this training set the learner

can estimate the distance between any pair x_i and x_j under ρ by

$$\hat{\rho}(x_i, x_j) = \frac{1}{M} \sum_{k=1}^{M} \sigma(g_k \circ f(x_i), g_k \circ f(x_j)).$$

For sufficiently large M, $\hat{\rho}(x_i, x_j)$ will with high probability be a good estimate of $\rho(x_i, x_j)$[1]. Given that the learner knows the function space $\mathcal{G}$ and the environmental measure Q, it can, *in principle*, calculate $\rho_{\mathcal{G}}(v, w)$ for any pair $v, w \in V$. Thus it can calculate $\rho_{\mathcal{G}}(\hat{f}(x_i), \hat{f}(x_j))$ for any pair x_i and x_j in its training set and compare this quantity with $\hat{\rho}(x_i, x_j)$. By the relation, $\rho(x, y) = \rho_{\mathcal{G}}(f(x), f(y))$, if $\hat{f}$ is close to the true representation f, and if $\hat{\rho}$ is a good estimate of ρ, then $\hat{\rho}(x_i, x_j)$ should be close to $\rho_{\mathcal{G}}(\hat{f}(x_i), \hat{f}(x_j))$. This suggests that the following error measure is appropriate for the learner's representation $\hat{f}$:

$$E(\hat{f}) = \sum_{i,j=1}^{N} \left[\rho_{\mathcal{G}}(\hat{f}(x_i), \hat{f}(x_j)) - \hat{\rho}(x_i, x_j) \right]^2. \tag{42}$$

An error of close to zero indicates that $\hat{f}$ maps all the training examples x_i into V in such a way that their mutual distances are the same as the mutual distances between the $f(x_i)$, where f is the true representation. For suitably smooth $\rho_{\mathcal{G}}$, $E(\hat{f})$ can be minimized using gradient descent.

6.2. Two Example Calculations of $\rho_{\mathcal{G}}$.

EXAMPLE 6.1. Suppose that $V = \mathrm{R}^n$ and $\mathcal{G}$ consists of all linear maps from V into R. $\mathcal{G}$ is the vector space dual of V and so is itself isomorphic to R^n. With this in mind, take the measure Q on $\mathcal{G}$ to be Lebesgue measure on R^n, but restrict Q's support to the cube $[-\alpha, \alpha]^n$. Let $\sigma(x, y) = (x - y)^2$ for all $x, y \in \mathrm{R}$. $\rho_{\mathcal{G}}$ can then be reduced as follows:

$$\begin{aligned}
\rho_{\mathcal{G}}(\vec{v}, \vec{w}) &= \int_{\mathcal{G}} \sigma(g(\vec{v}), g(\vec{w}))\, dQ(g) \\
&= \int\limits_{\vec{a} \in [-\alpha, \alpha]^n} (\vec{a} \cdot \vec{v} - \vec{a} \cdot \vec{w})^2 \, d\vec{a} \\
&= \int_{-\alpha}^{\alpha} \cdots \int_{-\alpha}^{\alpha} \left(\sum_{i=1}^{n} a_i v_i - \sum_{i=1}^{n} a_i w_i \right)^2 da_1 \ldots da_n \\
&= \frac{2\alpha^3}{3} \|\vec{v} - \vec{w}\|^2.
\end{aligned}$$

Thus $\mathcal{G}$ and Q induce the squared Euclidean distance on V in this case.

EXAMPLE 6.2. Take the same example as above but this time threshold the output of each $g \in \mathcal{G}$ with the Heaviside step function, and take Q to have support only on the unit ball in R^n (rather than the cube, $[-\alpha, \alpha]^n$). This gives,

$$\rho_{\mathcal{G}}(\vec{v}, \vec{w}) = \frac{\theta}{\pi},$$

[1] Although I have not done this, the techniques of chapter 3 and appendix B could no doubt be extended to provide bounds on M ensuring good agreement bewteen $\hat{\rho}$ and ρ.

where θ is the angle between $\vec{v}$ and $\vec{w}$.

6.3. Representation Learning for Classification Problems

An environment consisting only of classification problems generates a very simple metric on the input space X. Any classifier environment is equivalent to a set of Boolean functions $\mathcal{H}: X \to \{0,1\}$ with the property that if $h(x) = 1$ for some $x \in X$ and $h \in \mathcal{H}$, then $h'(x) = 0$ for all $h' \in \mathcal{H}, h' \neq h$. For example, in character recognition—a classifier environment—each image $x \in X$ is an example of only one character (it cannot be an image of both an "A" and a "Z", for example), or possibly an image of no character, so that at most one of the classifiers $h \in \mathcal{H}$ has $h(x) = 1$. A classifier environment $\mathcal{H}$ induces a partition of the input space X by

$$X = X_{\text{junk}} \bigcup_{h \in \mathcal{H}} X_h ,$$

where $X_h = \{x \in X : h(x) = 1\}$ and $X_{\text{junk}} = \{x \in X : h(x) = 0 \text{ for all } h \in \mathcal{H}\}$. If the environmental measure Q is assumed to be uniform over $\mathcal{H}$[2] then the metric ρ on X takes only three values: $\rho(x, x') = 0$ if $x, x' \in X_{\text{junk}}$ or if there exists $h \in \mathcal{H}$ such that $x, x' \in X_h$, $\rho(x, x') = 1/|\mathcal{H}|$ if one and only one of x, x' is in X_{junk} and $\rho(x, x') = 2/|\mathcal{H}|$ otherwise. If $\mathcal{H}$ has infinite cardinality then ρ will be indentically zero, however in any practical situation $\mathcal{H}$ will be effectively finite.

Suppose for the moment that the distribution P on X is such that X_{junk} is a set of measure zero, in which case ρ is effectively only two-valued. If the output functions $\mathcal{G}$ are assumed to be linear or thresholded linear then the formulae derived in the previous two examples for $\rho_{\mathcal{G}}$ can be substituted into the expression for the error measure (42), along with either $\hat{\rho}(x_i, x_j) = 0$ or $\hat{\rho}(x_i, x_j) = 2/|\mathcal{H}|$, depending on whether x_i and x_j have the same classification or not[3]. A representation $\hat{f}$ minimizing the resulting expression can then be found by gradient descent. An $\hat{f}$ with zero error would map each subset X_h to its own vector $v_h \in V$ such that $\rho_{\mathcal{G}}(v_h, v'_h) = c$ if $h \neq h'$. Thus if $\mathcal{G}$ is as in example 6.1 or 6.2, a perfect representation will map all inputs $x \in X$ to one of $|\mathcal{H}|$ vectors in V that are all either mutually separated by the same distance, or are the same angle apart. This places a lower bound of $|\mathcal{H}| - 1$ on the dimensionality of V because it must be possible to embed a tetrahedron with $|\mathcal{H}|$ vertices within V. If the dimensionality of V is just large enough then X (or at least the support of P) will be mapped onto the corners of a tetrahedron (if $\mathcal{G}$ is linear) or vectors pointing towards the corners of a tetrahedron (if $\mathcal{G}$ is thresholded linear). Note that if X_{junk} does have posisitve probability under P then it can be mapped to the centroid of the tetrahedron and although this does not quite give $\rho(x, x') = 1/|\mathcal{H}|$ for $x \in X_{\text{junk}}$ and $x' \in X_h$ for some $h \in \mathcal{H}$, it does give the correct *qualitative* behaviour in that $\rho(x, x')$ is at least *constant* in this case.

It may be that the exact space $\mathcal{G}$ used by the environment is not known to the learner. If so, the learner can still proceed in a somewhat less rigorous manner by selecting its own representation space $\mathcal{G}'$, guessing a form for the metric $\rho_{\mathcal{G}'}$ and then

[2] In practice the environmental measure Q will not be uniform, for example not every character has equal likelihood of being observed, however deviations from uniformity are usually small and hence a representation that is learnt under the assumption of uniformity would be quite likely to perform well in a non-uniform environment.

[3] In this case, because it is known that $\mathcal{H}$ is a set of classifiers and Q is uniform, $\hat{\rho}$ is known to be equal to ρ.

plugging this into formula (42). This approach can be used to avoid the quite rigid requirements that are imposed on the behaviour of a representation by the $\rho_{\mathcal{G}}$'s in examples 6.1 and 6.2, as follows. Rather than forcing the representation $\hat{f}$ to map the inputs exactly towards (or onto) the corners of a tetrahedron, it is sufficient if the representation simply maps all x with the same classification into the same *region* in V, in such a way that the distance between $\hat{f}(x)$ and the *centroid* of its region is smaller than the distance between $\hat{f}(x)$ and the centroid of any other region. The classification of any novel $x \in X$ can then be determined by computing $\hat{f}(x)$ and finding the region with the closest centroid to $\hat{f}(x)$. Learning a new classification category (i.e. a new task) using the same representation would simply be a matter of collecting enough examples of the new task to accurately estimate the location of its centroid. One way of achieving this is to take $\hat{\rho}(x, x') = 0$ if x and x' have the same classification (i.e. $x, x' \in X_h$ for some $h \in \mathcal{H}$) , $\hat{\rho}(x, x') = 1$ otherwise, and take

$$\rho_{\mathcal{G}}(\hat{f}(x), \hat{f}(x')) = 1 - \exp\left(-\frac{\|\hat{f}(x) - \hat{f}(x')\|^2}{T}\right).$$

The error (42) on a sample $x_1, \ldots, x_N$ then becomes,

$$E(\hat{f}) = \sum_{i,j=1}^{N} \left[1 - \exp\left(-\frac{\|\hat{f}(x_i) - \hat{f}(x_j)\|^2}{T}\right) - \hat{\rho}(x_i, x_j)\right]^2. \tag{43}$$

$E(\hat{f})$ will be minimal for a representation that draws inputs x with the same classification as close together as possible, and separates those with different classifications as far apart as possible. The exponential causes $\hat{f}(x)$ and $\hat{f}(x')$ that are more than a small Euclidean distance apart to be viewed, under $\rho_{\mathcal{G}}$, as having nearly unit separation. The parameter T controls the tightness of the clusters in V, and their relative separation. A small value for T encourages representations to form very tight clusters that are not necessarily separated very much, while a large T allows clusters to be looser, but forces them to be further apart. If necessary, rather than making an arbitrary choice for the value of T, it could be optimized along with the parameters of the representation.

6.4. Experiment

The error measure (43) was used to learn a representation for the translation-invariant environment of section 4.2.3. The objects of figure 4.2.4 were the classification tasks. The training data $x_1, \ldots, x_N$ was generated by selecting one of the four objects uniformly at random and placing it uniformly at random within the ten-pixel input space (wrapping at the edges was again permitted). The representations $f \in \mathcal{F}$ had no hidden layers, one output node and no squashing function, so that f was in fact a linear map. Four values of the parameter T were tried: $1, 0.1, 0.01$, and 0.001 with 0.01 giving the best results in that the fewest local minima were encountered with this value. $T = 0.01$ was the value used in the results reported below. $E(\hat{f})$ was minimized using conjugate gradient descent with stopping criteria of $E(\hat{f})/N < 10^{-7}$ or an L^∞ error[4] of less than 10^{-3}. If neither of these criteria were achieved and

[4] That is, the maximium absolute difference between $1 - \exp\left(-\frac{\|\hat{f}(x_i) - \hat{f}(x_j)\|^2}{T}\right)$ and $\hat{\rho}(x_i, x_j)$ for any distinct pair x_i, x_j.

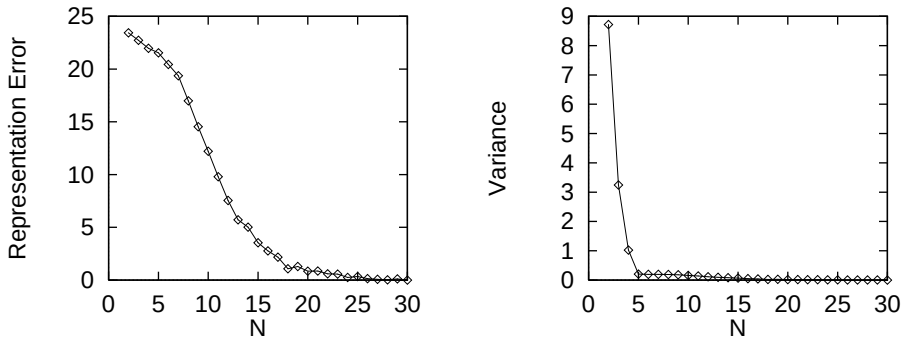

FIGURE 6.4.23. Representation error and variance: experiment 1.

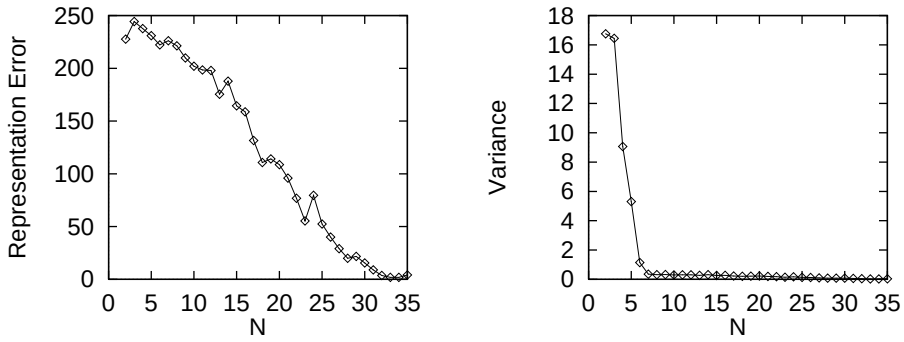

FIGURE 6.4.24. Representation error and variance: experiment 2.

there was a less than 0.01% improvement in the error over five iterations the network was deemed to have struck a local minima and was restarted with a new randomly generated set of weights. The weights were generated uniformly in the range $[0, 0.1]$.

A succesful representation $\hat{f}$ was tested by first computing the four centroids $c_i = \frac{1}{10}\sum_{j=1}^{10} \hat{f}(x_{ij})$, $1 \leq i \leq 4$, where x_{ij} is the input vector consisting of the ith object placed in the jth input position, and then counting up the number of inputs x_{ij} for which $\hat{f}(x_{ij})$ was not minimal distance from its corresponding centroid c_i. The average of the four within group variances,

$$\mathrm{var}_i = \frac{1}{10}\sqrt{\sum_{j=1}^{10}\sum_{k=1}^{10}\left[\hat{f}(x_{ij}) - \hat{f}(x_{ik})\right]^2}$$

was also computed. 200 independent simulations were performed with training sets consisting of from 2 to 40 examples[5]. The average of the error and variance results across all simulations is plotted in figure 6.4.23. A second set of simulations was performed with a thirty-pixel input space and with 10 objects consisting of from 1 to 10 active adjacent pixels. The results are plotted in figure 6.4.24. Due to the larger problem size the curves are the average of only 24 separate simulations in this case.

[5] The simulations took about two hours to run on a 66Mhz, 486DX notebook computer.

Note the very quick convergence to zero variance in both experiments. This is significant because it indicates that learning with a succsessful representation $\hat{f}$ would require only one example of an object to determine its centroid accurately, and hence only one example of the object to learn to recognise it perfectly. It is somewhat surprising that the representation error does not fall to zero as quickly as the variance. This is probably due to the fact that the small value of T caused clusters to be quite close together and hence required very small variance to ensure no misclassification. This needs to be investigated further.

This method also greatly reduced the number of restarts required—the average number in the first experiment was 0.3 and in the second 1.4 (compare this with the graphs in figure 4.2.18).

6.5. Discussion and Future Directions

At least for the toy problem examined here, the experimental results show that (43) is a good error measure to use for representation learning in classification problems. Although this particular form for the error measure may not be the best one to use in all problems, the general idea of training the representation to *match a metric* on its output space should be one that is widely applicable in practice. Intuitively, matching a metric, particularly in the "soft" manner allowed by (43), should be "easier" than explicitly demanding a particular output from a given input, as happens with ordinary learning and with the representation learning techniques of chapter 4.

Metric error measures also hold promise as a means of defining an error measure on *all* layers of a representation directly, not just the final one as in (43). Although I do not yet have any explicit method for this, the general idea is to note that if an arbitrary metric ρ is selected for the input space X, say the Euclidean metric, and the metric $\rho_{\mathcal{G}}$ is known for the output space V, then the successive layers of a representation f can be thought of as mapping X through a series of intermediary spaces $V_1, V_2, \ldots$ in such a way that at each stage it becomes "easier" for the remaining layers to produce behaviour at their output nodes that matches $\rho_{\mathcal{G}}$. If the idea of an early layer mapping that makes things "easier" for later layers could be quantified, it could be used as an error measure to recursively train the first hidden layer, then the second and so on. To finish on an even more speculative note, if there was some way of measuring the distance between ρ and $\rho_{\mathcal{G}}$[6] then it might be possible to make an estimate of how many successive "neural" layers of processing would be required to convert ρ into $\rho_{\mathcal{G}}$ and possibly even the number of nodes needed in each layer.

This is by far the most speculative and unfinished part of this thesis. However, I believe the general framework presented here is a most promising one for further investigation.

[6] A metric on metrics!

APPENDIX A

Measurability and Permissibility

The definitions and results ensuring measurability of all the necessary functions and sets in chapters 2 and 3 and appendix B are presented in this appendix.

DEFINITION A.1. Given any set of functions $\mathcal{H}: Z \to [0, M]$, where Z is any set, let $\sigma_{\mathcal{H}}$ be the σ-algebra on Z generated by all inverse images under functions in $\mathcal{H}$ of all open balls (under the usual topology on R) in $[0, M]$. Let $\mathcal{P}_{\mathcal{H}}$ denote the set of all probability measures on $\sigma_{\mathcal{H}}$.

This definition ensures measurability of any function $h \in \mathcal{H}$ with respect to any measure $P \in \mathcal{P}_{\mathcal{H}}$. For some of the results stated elsewhere (e.g. theorem B.6), functions formed by taking suprema over $\mathcal{H}$ also have to be measurable. This is assured if $\mathcal{H}$ is *permissible*. The following two definitions are taken (with minor modifications) from Pollard (1984).

DEFINITION A.2. A family, $\mathcal{H}$, of $[0, M]$-valued functions on a set Z is *indexed* by the set T if there exists a function $f: Z \times T \to [0, M]$ such that

$$\mathcal{H} = \{f(\,\cdot\,, t): t \in T\}\,.$$

DEFINITION A.3. The set $\mathcal{H}$ is *permissible* if it can be indexed by a set T such that

(1) T is an *analytic* subset of a Polish space $\overline{T}$, and
(2) the function $f: Z \times T \to [0, M]$ indexing $\mathcal{H}$ by T is measurable with respect to the product σ-algebra $\sigma_{\mathcal{H}} \otimes \sigma(T)$, where $\sigma(T)$ is the Borel σ-algebra induced by the topology on T.

An analytic subset T of a Polish space $\overline{T}$ is simply the continuous image of a Borel subset X of another Polish space $\overline{X}$. The analytic subsets of a Polish space include the Borel sets. They are important because projections of analytic sets are analytic, and can be measured in a complete measure space whereas projections of Borel sets are not necessarily Borel, and hence cannot be measured with a Borel measure. For more details see Dudley (1989), section 13.2.

DEFINITION A.4. Let $\mathcal{H}_1, \ldots, \mathcal{H}_n$ be n sets of functions mapping Z into $[0, M]$. For all $h_1 \in \mathcal{H}_1, \ldots, h_n \in \mathcal{H}_1$, define $h_1 \oplus \cdots \oplus h_n: Z^n \to [0, M]$ by

$$h_1 \oplus \cdots \oplus h_n(\vec{z}) = \frac{1}{n} \sum_{i=1}^{n} h_i(z_i),$$

for all $\vec{z} = (z_1, \ldots, z_n) \in Z^n$. Denote the set of all such functions by $\mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_n$. Occasionally $(h_1, \ldots, h_n)$ and $\vec{h}$ will also be used to denote the function $h_1 \oplus \cdots \oplus h_n$.

LEMMA A.1. *$\mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_n : Z^n \to [0, M]$ is permissible if $\mathcal{H}_1, \ldots, \mathcal{H}_n$ are all permissible.*

PROOF. Let each $\mathcal{H}_i$ be indexed by $f_i : Z \times T_i \to [0, M]$ where T_i is an analytic subset of the Polish space $\overline{T}_i$. Then $\mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_n$ is indexed by $f : Z^n \times T_1 \times \cdots \times T_n \to [0, M]$ where

$$f(\vec{z}, t_1, \ldots, t_n) = \frac{1}{n} \sum_{i=1}^{n} f_i(z_i, t_i),$$

for all $\vec{z} = (z_1, \ldots, z_n) \in Z^n$ and $t_i \in T_i, 1 \leq i \leq n$. $\overline{T}_1 \times \cdots \times \overline{T}_n$ with product topology is itself a Polish space (Dudley (1989), section 13.2) and it is easily verified that $T_1 \times \cdots \times T_n$ is an analytic subset of $\overline{T}_1 \times \cdots \times \overline{T}_n$. Also f is clearly measurable with respect to the product σ-algebra $\sigma_{\mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_n} \otimes \sigma_{T_1 \times \cdots \times T_n}$. The result follows. □

For the representation learning results of chapter 3 and particularly the more general results of section 3.6 to hold, the concept of permissibility must be introduced for *hypothesis space families*, that is, sets $H = \{\mathcal{H}\}$ where each $\mathcal{H} \in H$ is itself a set of functions mapping Z into $[0, M]$.

DEFINITION A.5. H is *f-permissible*[1] if there exist sets S and T that are analytic subsets of Polish spaces $\overline{S}$ and $\overline{T}$ respectively, and a function $f : Z \times T \times S \to [0, M]$, measurable with respect to $\sigma_{H_\sigma} \otimes \sigma(T) \otimes \sigma(S)$, such that

$$H = \big\{ \{ f(\cdot, t, s) : t \in T \} : s \in S \big\}.$$

DEFINITION A.6. For any hypothesis space family H, define $H_\sigma = \{h : h \in \mathcal{H} : \mathcal{H} \in H\}$ and define $H^n = \{\mathcal{H} \oplus \cdots \oplus \mathcal{H} : \mathcal{H} \in H\}$. For all $h \in H_\sigma$ define $\overline{\boldsymbol{h}} : \mathcal{P}_{H_\sigma} \to [0, M]$ by $\overline{\boldsymbol{h}}(P) = \langle h \rangle_P$. Let $\overline{H}_\sigma = \{\overline{\boldsymbol{h}} : h \in H_\sigma\}$ and for all $\mathcal{H} \in H$ let $\overline{\mathcal{H}} = \{\overline{\boldsymbol{h}} : h \in \mathcal{H}\}$. For all $\mathcal{H} \in H$ define $\mathcal{H}^* : \mathcal{P}_{H_\sigma} \to [0, M]$ by $\mathcal{H}^*(P) = \inf_{h \in \mathcal{H}} \overline{\boldsymbol{h}}(P)$. Let $H^* = \{\mathcal{H}^* : \mathcal{H} \in H\}$. For any probability measure $Q \in \mathcal{P}_{\overline{H}_\sigma}$, define the probability measure Q_Z on Z by $Q_Z(S) = \int_{\mathcal{P}_{H_\sigma}} P(S)\, dQ(P)$, for all $S \in \sigma_{H_\sigma}$.

LEMMA A.2. *Let H be a family of hypothesis spaces mapping Z into $[0, M]$. Take the σ-algebra on Z to be σ_{H_σ}, the set of probability measures on Z to be $\mathcal{P}_{H_\sigma}$ and the σ-algebra on $\mathcal{P}_{H_\sigma}$ to be $\sigma_{\overline{H}_\sigma}$. With these choices, if H is f-permissible then*

(1) *H_σ is permissible.*
(2) *H^n is f-permissible.*
(3) *$[H^n]_\sigma$ is permissible.*
(4) *$\overline{H}_\sigma$ is permissible.*
(5) *$\overline{\mathcal{H}}$ is permissible for all $\mathcal{H} \in H$.*
(6) *$\mathcal{H}$ is permissible for all $\mathcal{H} \in H$.*
(7) *$\mathcal{H}^*$ is measurable for all $\mathcal{H} \in H$.*
(8) *H^* is permissible.*
(9) *For all $Q \in \mathcal{P}_{\overline{H}_\sigma}$, $Q_Z \in \mathcal{P}_{H_\sigma}$.*

[1] "f" for *family*.

PROOF. Let H be indexed by $f: Z \times T \times S \to [0, M]$ in the appropriate way. By taking the product topology on $T \times S$, the same construction used in the proof of lemma A.1 can be used to show (1) and (2). (3), (4) and (5) are then direct consequences of (1) and (2), while (6) follows immediately from f-permissibility of H. (7) follows from (5).

(8) is more problematic. To begin with it must be further assumed that $(\mathcal{P}_{H_\sigma}, \sigma_{\overline{H}_\sigma}, Q)$ is a complete probability space. If this is not the case, simply add all sets of Q-measure zero to $\sigma_{\overline{H}_\sigma}$ and generate a new σ-algebra from the result. Let (X, σ_X, μ) be any measure space and T be an analytic subset of a Polish space. Let $\mathcal{A}(X)$ denote the analytic subsets of any set X (note that the analytic sets actually depend on σ_X, but it seems to be convention to write $\mathcal{A}(X)$). The following three facts about analytic sets are stated in Pollard (1984), appendix C.

(a) If (X, σ_X, μ) is complete then $\mathcal{A}(X) = \sigma_X$.

(b) $\mathcal{A}(X \times T)$ contains the product σ-algebra $\sigma_X \otimes \sigma(T)$.

(c) For any set Y in $\mathcal{A}(X \times T)$, the projection $\pi_X Y$ of Y onto X is in $\mathcal{A}(X)$.

Now, define $g: \mathcal{P}_{H_\sigma} \times T \times S \to [0, M]$ by $g(P, t, s) = \int_Z f(z, t, s)\, dP(z)$ where f indexes H as above. By Fubini's theorem and the construction of $\sigma_{\overline{H}_\sigma}$, g is a $\sigma_{\overline{H}_\sigma} \otimes \sigma(T) \otimes \sigma(S)$-measurable function. Let $G: \mathcal{P}_{H_\sigma} \times S \to [0, M]$ be defined by $G(P, s) = \inf_{t \in T} g(P, t, s)$. G indexes H^* in the appropriate way for H^* to be permissible, as long as it can be shown that G is $\sigma_{\overline{H}_\sigma} \otimes \sigma(S)$-measurable. This is where analyticity becomes important. Let $g_\alpha = \{(P, t, s): g(P, t, s) > \alpha\}$. By property (b) above, $\mathcal{A}(\mathcal{P}_{H_\sigma} \times T \times S)$ contains g_α. The set $G_\alpha = \{(P, s): G(P, s) > \alpha\}$ is the projection of g_α onto $\mathcal{P}_{H_\sigma} \times S$, which by property (c) is also analytic. As $(\mathcal{P}_{H_\sigma}, \sigma_{\overline{H}_\sigma}, Q)$ is assumed complete, G_α is measurable, by property (a). Thus G is a measurable function and the permissibility of H^* follows.

For (9) note that for all $h \in H_\sigma$,

$$\int_{\mathcal{P}_{H_\sigma}} \overline{\boldsymbol{h}}(P)\, dQ(P) = \int_Z h(z)\, dQ_Z(z),$$

and so each $h \in H_\sigma$ is Q_Z-measurable. □

APPENDIX B

The Fundamental Theroem

The fundamental theorem (B.6) bounds the probability of large deviation between empirical estimates of the error of functions $h_1 \oplus \cdots \oplus h_n$ (recall definition A.4) and their true error. The theorem is crucial for deriving bounds on the sample size required for good generalisation in representation learning, hence the description *fundamental*. It is used heavily in sections 3.5 and 3.6, although only implicitly in the latter section. The fundamental theorem is a generalisation of a similar theorem for ordinary learning derived by Haussler (1992), and the same techniques used by Haussler in deriving his theorem are used here.

Before the theorem can be stated some preliminary definitions must be introduced, which in many cases are repetitions of definitions given in chapters 2 and 3. They are reproduced here for clarity and smoothness of exposition.

B.1. Miscellaneous definitions

For any set Z, denote the set of $m \times n$ matrices over Z by $Z^{(m,n)}$ Matrices will always be denoted by lower-case, bold-face characters,

$$\mathbf{z} = \begin{matrix} z_{11} & \cdots & z_{1n} \\ \vdots & \ddots & \vdots \\ z_{m1} & \cdots & z_{mn}. \end{matrix}$$

The *rows* of $\mathbf{z}$ will always be denoted by $\vec{z}_1, \ldots, \vec{z}_m$

Let $\mathcal{H}$ be any set of functions mapping Z^n into $[0, M]$. For all $h \in \mathcal{H}$ and $\mathbf{z} \in Z^{(m,n)}$, define

$$\langle h \rangle_{\mathbf{z}} = \frac{1}{m} \sum_{i=1}^{m} h(\vec{z}_i).$$

If P is any probability measure on Z^n then for all $h \in \mathcal{H}$ define

$$\langle h \rangle_P = \int_{Z^n} h(\vec{z}) \; dP(\vec{z}).$$

$\langle h \rangle_{\mathbf{z}}$ is the *empirical error* of h on the sample $\mathbf{z}$ and $\langle h \rangle_P$ is the *true error* of h with respect to P.

The main theorem of this section is about bounding the probability of large deviation between $\langle h \rangle_{\mathbf{z}}$ and $\langle h \rangle_P$. Such deviations are measured with the d_ν metric.

DEFINITION B.1. For all $x, y \in \mathrm{R}^+$ and all $\nu > 0$, let

$$d_\nu(x,y) = \frac{|x-y|}{\nu + x + y}.$$

LEMMA B.1. *The d_ν metric has the following three properties:*

(1) $0 \le d_\nu(r,s) \le 1$.
(2) *For all* $r \le s \le t$, $d_\nu(r,s) \le d_\nu(r,t)$ *and* $d_\nu(s,t) \le d_\nu(r,t)$.
(3) *For* $0 \le r, s \le M$, $\frac{|r-s|}{\nu+2M} \le d_\nu(r,s) \le \frac{|r-s|}{\nu}$.

Following Haussler, the second property is referred to as the *compatibility of d_ν with the ordering on the reals.*

B.2. Psuedo-metric spaces and ε-covers.

DEFINITION B.2. A *pseudo-metric space* (X, ρ) is a metric space without the constraint that $\rho(x,y) = 0 \Rightarrow x = y$. An *$\varepsilon$-cover* of a pseudo-metric space (X, ρ) is a set $T \subseteq X$ such that for all $x \in X$, there exists $t \in T$ such that $\rho(x,t) \le \varepsilon$. A set $T \subseteq X$ is called *ε-separated* if for all distinct $s, t \in T$, $\rho(s,t) > \varepsilon$. Denote the smallest ε-cover of a pseudo-metric space (X, ρ) by $\mathcal{N}(\varepsilon, X, \rho)$. Denote the largest ε-separated subset by $\mathcal{M}(\varepsilon, X, \rho)$. $\mathcal{M}(\varepsilon, X, \rho)$ is also referred to as a *packing number.* X is *totally bounded* if it has a finite ε-cover for all $\varepsilon > 0$.

The following four lemmas are the only ones about pseudo-metrics and ε-covers that are needed to prove the results in the rest of this work. They are all quite trivial. The first is due to Kolmogorov and Tihomirov (1961).

LEMMA B.2. *If T is a totally bounded subset of a pseudo-metric space (X, ρ), then for all $\varepsilon > 0$,*

$$\mathcal{M}(2\varepsilon, T, \rho) \le \mathcal{N}(\varepsilon, T, \rho) \le \mathcal{M}(\varepsilon, T, \rho).$$

LEMMA B.3.

$$\mathcal{M}(2\varepsilon, T, \rho) = \infty \Rightarrow \mathcal{N}(\varepsilon, T, \rho) = \infty.$$

LEMMA B.4. *Let (X, ρ) and (Y, σ) be totally bounded pseudo-metric spaces. Let $\psi : X \to Y$ be surjective. If ψ is a* contraction, *i.e for all $x, x' \in X$*

$$\sigma(\psi(x), \psi(x')) \le \rho(x, x'),$$

then for all $\varepsilon > 0$

$$\mathcal{N}(\varepsilon, Y, \sigma) \le \mathcal{N}(\varepsilon, X, \rho),$$

with equality always if ψ is an isometry.

LEMMA B.5. *For all $x > 0$,*

$$\mathcal{N}(\varepsilon, \mathcal{F}, x\rho) = \mathcal{N}\left(\frac{\varepsilon}{x}, \mathcal{F}, \rho\right).$$

B.2.1. Capacity.

DEFINITION B.3. Given a set of functions $\mathcal{H}$ from any set Z into $[0, M]$ and a probability measure P on Z, define the pseudo-metric d_P on $\mathcal{H}$ by,

$$d_P(f, g) = \int_Z |f(z) - g(z)| \, dP(z),$$

for all $f, g \in \mathcal{H}$.

Note that d_P is only a pseudo-metric because f and g could differ on a set of measure zero and still be equal under d_P.

DEFINITION B.4. Given a set of functions $\mathcal{H}$ from any set Z into $[0, M]$, for all $\varepsilon > 0$ let

$$\mathcal{C}(\varepsilon, \mathcal{H}) = \sup_{P \in \mathcal{P}_{\mathcal{H}}} \mathcal{N}(\varepsilon, \mathcal{H}, d_P)$$

when the supremum exists and $\mathcal{C}(\varepsilon, \mathcal{H}) = \infty$ otherwise (recall definition A.1 for the definition of the set of probability measures $\mathcal{P}_{\mathcal{H}}$ on Z). Call $\mathcal{C}(\varepsilon, \mathcal{H})$ the *ε-capacity* of $\mathcal{H}$.

B.3. The Fundamental Theorem

THEOREM B.6. *Let $\mathcal{H} \subseteq \mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_n$ be a permissible set of functions mapping Z^n into $[0, M]$. Let $\mathbf{z} \in Z^{(m,n)}$ be generated by $m \geq \frac{2M}{\alpha^2 \nu}$ independent trials from Z^n according to some product probability measure $\vec{P} \in \mathcal{P}_{\mathcal{H}}$, $\vec{P} = P_1 \times \cdots \times P_n$. For all $\nu > 0$, $0 < \alpha < 1$,*

$$\Pr\left\{\mathbf{z} \in Z^{(m,n)} : \exists \vec{h} \in \mathcal{H} : d_\nu\left(\langle \vec{h} \rangle_{\mathbf{z}}, \langle \vec{h} \rangle_{\vec{P}}\right) > \alpha\right\} \leq 4\mathcal{C}(\alpha\nu/8, \mathcal{H}) \, e^{-\frac{\alpha^2 \nu n m}{8M}}.$$

The restriction $\vec{P} \in \mathcal{P}_{\mathcal{H}}$ is only required to ensure measurability of all functions in $\mathcal{H}$ with respect to $\vec{P}$ (recall definition A.1). The restriction of this result to $n = 1$ is used several times elsewhere so for convenience it is stated here as a corollary. The corollary is also a direct consequence of Haussler's theorem 3.

COROLLARY B.7. *Let $\mathcal{H}$ be a permissible set of functions from Z into $[0, M]$. Let $\vec{z} \in Z^m$ be generated by m independent trials from Z according to some probability measure $P \in \mathcal{P}_{\mathcal{H}}$. For all $\nu > 0$, $0 < \alpha < 1$,*

$$\Pr\{\vec{z} \in Z^m : \exists h \in \mathcal{H} : d_\nu(\langle h \rangle_{\vec{z}}, \langle h \rangle_P) > \alpha\} \leq 4\mathcal{C}(\alpha\nu/8, \mathcal{H}) e^{-\frac{\alpha^2 \nu m}{8M}}.$$

It is the exponent of the exponential in these two results that is most important. In the general theorem, the probablity decays exponentially in the *product* mn, whilst in the $n = 1$ case it simply decays exponentially in m. Ultimately, it is the appearance of the product that makes representation learning feasible. Note also that the restriction $m \geq 2M/(\alpha^2\nu)$ in the fundamental theorem does not actually have any effect on the results presented elsewhere in the work because the bound is always automatically exceeded (look for example at the bound on m in theorem 3.8).

The proof of theorem B.6 is quite involved. The path followed here differs only where necessary from that taken by Haussler in proving his theorem 3, although some delicate footwork is required in translating from the $n = 1$ case to the higher dimensional case. In what follows, where a lemma is an immediate consequence of a corresponding result in Haussler, the proof has been omitted and the number of Haussler's result listed in parentheses.

DEFINITION B.5. Let $\mathcal{H} \subseteq \mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_n$ be a set of real-valued functions on Z^n. For all $\vec{h} = h_1 \oplus \cdots \oplus h_n \in \mathcal{H}$ and all $\mathbf{z} \in Z^{(m,n)}$, define $\vec{h}(\mathbf{z}) \in \mathrm{R}^{(m,n)}$ by

$$\vec{h}(\mathbf{z}) = \begin{matrix} h_1(z_{11}) & \dots & h_n(z_{1n}) \\ \vdots & \ddots & \vdots \\ h_1(z_{m1}) & \dots & h_n(z_{mn}), \end{matrix}$$

and let $\mathcal{H}_{|\mathbf{z}} = \left\{ \vec{h}(\mathbf{z}) : \vec{h} \in \mathcal{H} \right\}$ be the set of all such points. Note that the notation is being overloaded here because the functions $\vec{h} \in \mathcal{H}$ are now defined on two separate spaces (Z^n and $Z^{(m,n)}$) and moreover have different ranges (R and $\mathrm{R}^{(m,n)}$)in each case. However this is not a problem as it will always be clear from the context which version of $\vec{h}$ is meant. Denote the function $\mathbf{z} \mapsto \mathcal{H}_{|\mathbf{z}}$ for all $\mathbf{z} \in Z^{(m,n)}$ by $\mathcal{H}_{|(m,n)}$.

DEFINITION B.6. For all $\mathbf{z} \in Z^{(2m,n)}$, let $\mathbf{z}(1)$ be the top half of $\mathbf{z}$ and $\mathbf{z}(2)$ be the bottom half, viz:

$$\mathbf{z}(1) = \begin{matrix} z_{11} & \dots & z_{1n} \\ \vdots & \ddots & \vdots \\ z_{m1} & \dots & z_{mn} \end{matrix} \qquad \mathbf{z}(2) = \begin{matrix} z_{m+11} & \dots & z_{m+1n} \\ \vdots & \ddots & \vdots \\ z_{2m1} & \dots & z_{2mn} \end{matrix}$$

DEFINITION B.7. For all integers $m, n \geq 1$ let $\Gamma_{(2m,n)}$ denote the set of all permutations σ of the sequence of pairs of integers $\{11, \dots, 1n, \dots, 2m\,1, \dots, 2m\,n\}$ such that for all i, $1 \leq i \leq m$, either $\sigma(ij) = m+i\,j$ and $\sigma(m+ij) = ij$ or $\sigma(ij) = ij$ and $\sigma(m+i\,j) = m+i\,j$. Thus the permutations in $\Gamma_{(2m,n)}$ swap selected indices in the first half of the sequence $\{11, \dots, 2m\,n\}$ with corresponding indices in the second half. For any $\mathbf{x} \in \mathrm{R}^{(2m,n)}$ and $\sigma \in \Gamma_{(2m,n)}$, let

$$\mu_1(\mathbf{x}, \sigma) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{x}_{\sigma(ij)},$$

$$\mu_2(\mathbf{x}, \sigma) = \frac{1}{mn} \sum_{i=m+1}^{2m} \sum_{j=1}^{n} \mathbf{x}_{\sigma(ij)}.$$

Given any metric d on R^+, extend it to a metric $\boldsymbol{d}$ on $(\mathrm{R}^+)^{(2m,n)}$ by

$$\boldsymbol{d}(\mathbf{x}, \mathbf{y}) = \max_{\sigma \in \Gamma_{(2m,n)}} d(\mu_1(\mathbf{x}, \sigma), \mu_1(\mathbf{y}, \sigma)) + d(\mu_2(\mathbf{x}, \sigma), \mu_2(\mathbf{y}, \sigma)),$$

for all $\mathbf{x}, \mathbf{y} \in (\mathrm{R}^+)^{(2m,n)}$.

DEFINITION B.8. Define the metric $\boldsymbol{d}_{L^1}$ on $(\mathrm{R}^+)^{(m,n)}$ by,

$$\boldsymbol{d}_{L^1}(\mathbf{x}, \mathbf{y}) = \frac{1}{mn} \sum_{i=1}^{m} \left| \sum_{j=1}^{n} (x_{ij} - y_{ij}) \right|,$$

for all $\mathbf{x}, \mathbf{y} \in (\mathrm{R}^+)^{(m,n)}$. Note that this is *not* the extension described in the previous definition of the L^1 metric on R^+ to $(\mathrm{R}^+)^{(m,n)}$ (which would, in any case, be denoted by $\mathbf{L}^1$.).

LEMMA B.8 (10). *For any metric d on R^+, for all $\mathbf{x}, \mathbf{y} \in (\mathrm{R}^+)^{(2m,n)}$ and any $\sigma \in \Gamma_{(2m,n)}$,*

$$d\left(\mu_1(\mathbf{x}, \sigma), \mu_2(\mathbf{x}, \sigma)\right) \leq d\left(\mu_1(\mathbf{y}, \sigma), \mu_2(\mathbf{y}, \sigma)\right) + \boldsymbol{d}(\mathbf{x}, \mathbf{y}).$$

LEMMA B.9. *For any $\mathbf{x}, \mathbf{y} \in (\mathrm{R}^+)^{(2m,n)}$ and $\nu > 0$,*

$$\boldsymbol{d}_\nu(\mathbf{x}, \mathbf{y}) \leq \frac{2}{\nu} \boldsymbol{d}_{L^1}(\mathbf{x}, \mathbf{y}).$$

PROOF. For any $\sigma \in \Gamma_{(2m,n)}$ and any $\mathbf{x}, \mathbf{y} \in (\mathrm{R}^+)^{(2m,n)}$,

$$\begin{aligned}
&d_\nu\left(\mu_1(\mathbf{x}, \sigma), \mu_1(\mathbf{y}, \sigma)\right) + d_\nu\left(\mu_2(\mathbf{x}, \sigma), \mu_2(\mathbf{y}, \sigma)\right) \\
&= \frac{\left|\sum_{i=1}^{m} \sum_{j=1}^{n} \left(\mathbf{x}_{\sigma(ij)} - \mathbf{y}_{\sigma(ij)}\right)\right|}{\nu mn + \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\mathbf{x}_{\sigma(ij)} + \mathbf{y}_{\sigma(ij)}\right)} + \frac{\left|\sum_{i=m+1}^{2m} \sum_{j=1}^{n} \left(\mathbf{x}_{\sigma(ij)} - \mathbf{y}_{\sigma(ij)}\right)\right|}{\nu mn + \sum_{i=m+1}^{2m} \sum_{j=1}^{n} \left(\mathbf{x}_{\sigma(ij)} + \mathbf{y}_{\sigma(ij)}\right)} \\
&\leq \frac{\sum_{i=1}^{2m} \left|\sum_{j=1}^{n} \left(\mathbf{x}_{\sigma(ij)} - \mathbf{y}_{\sigma(ij)}\right)\right|}{\nu mn} \\
&= \frac{2}{\nu} \boldsymbol{d}_{L^1}(\mathbf{x}, \mathbf{y}).
\end{aligned}$$

□

LEMMA B.10 (11). *If a permutation $\sigma \in \Gamma_{(2m,n)}$ is chosen uniformly at random, then for all $\nu > 0$, $0 < \alpha < 1$, and $\mathbf{x} \in [0, M]^{(2m,n)}$,*

$$\Pr\left\{\sigma \in \Gamma_{(2m,n)} \colon d_\nu\left(\mu_1(\mathbf{x}, \sigma), \mu_2(\mathbf{x}, \sigma)\right) > \alpha\right\} \leq 2e^{-\frac{2\alpha^2 \nu mn}{M}}.$$

The following lemma relates the probability of large deviation between the empirical and true error of a function h to the probability of large deviation between two independent empirical estimates of h's error.

LEMMA B.11 (12). *Let $\mathcal{H}$ be a permissible set of functions from Z^n into $[0, M]$ and $P \in \mathcal{P}_\mathcal{H}$ be a probability measure on Z^n. For all $\nu > 0, 0 < \alpha < 1$ and $m \geq \frac{2M}{\alpha^2 \nu}$,*

$$\begin{aligned}
&\Pr\left\{\mathbf{z} \in Z^{(m,n)} \colon \exists h \in \mathcal{H} \colon d_\nu\left(\langle h \rangle_\mathbf{z}, \langle h \rangle_P\right) > \alpha\right\} \\
&\qquad \leq 2 \Pr\left\{\mathbf{z} \in Z^{(2m,n)} \colon \exists h \in \mathcal{H} \colon d_\nu\left(\langle h \rangle_{\mathbf{z}(1)}, \langle h \rangle_{\mathbf{z}(2)}\right) > \frac{\alpha}{2}\right\}.
\end{aligned}$$

Recalling definition B.5, note that for all $\varepsilon > 0$, $\mathcal{N}\left(\varepsilon, \mathcal{H}_{|(2m,n)}, \boldsymbol{d}_\nu\right)$ is a random variable on $Z^{(2m,n)}$, where $\boldsymbol{d}_\nu$ is the $(\mathrm{R}^+)^{(2m,n)}$ extension of d_ν given in definition B.7. The following lemma bounds the probability in theorem B.6 in terms of the expected value of this random variable.

LEMMA B.12. *Let $\mathcal{H} \subseteq \mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_n$ be a permissible set of functions from Z^n into $[0, M]$. Suppose that $\mathbf{z} \in Z^{(m,n)}$ is generated by $m \geq \frac{2M}{\alpha^2 \nu}$ independent trials from Z^n according to some product probability measure $\vec{P} \in \mathcal{P}_\mathcal{H}$, $\vec{P} = P_1 \times \cdots \times P_n$. Then for all $\nu > 0$ and $0 < \alpha < 1$,*

$$\begin{aligned}
&\Pr\left\{\mathbf{z} \in Z^{(m,n)} \colon \exists \vec{h} \in \mathcal{H} \colon d_\nu\left(\langle \vec{h} \rangle_{\vec{P}}, \langle \vec{h} \rangle_\mathbf{z}\right) > \alpha\right\} \\
&\qquad \leq 4 \langle \mathcal{N}\left(\alpha/4, \mathcal{H}_{|(2m,n)}, \boldsymbol{d}_\nu\right) \rangle_{\vec{P}^{2m}} e^{-\frac{\alpha^2 \nu n m}{8M}}.
\end{aligned}$$

PROOF. By lemma B.11,

$$\Pr\left\{\mathbf{z}\in Z^{(m,n)}\colon \exists\vec{h}\in\mathcal{H}\colon d_\nu\left(\langle\vec{h}\rangle_{\vec{P}},\langle\vec{h}\rangle_{\mathbf{z}}\right)>\alpha\right\}$$
$$\leq 2\Pr\left\{\mathbf{z}\in Z^{(2m,n)}\colon \exists\vec{h}\in\mathcal{H}\colon d_\nu\left(\langle\vec{h}\rangle_{\mathbf{z}(1)},\langle\vec{h}\rangle_{\mathbf{z}(2)}\right)>\frac{\alpha}{2}\right\},$$

so only the latter quantity needs bounding.

First, for any $\mathbf{z}\in Z^{(2m,n)}$ and any $\sigma\in\Gamma_{(2m,n)}$, let

$$\mathbf{z}_\sigma=\begin{matrix} z_{\sigma(11)} & \cdots & z_{\sigma(1n)}\\ \vdots & \ddots & \vdots\\ z_{\sigma(m1)} & \cdots & z_{\sigma(mn)}.\end{matrix}$$

For any fixed function $\vec{h}\in\mathcal{H}$ and fixed $\mathbf{z}\in Z^{(2m,n)}$, if $\sigma\in\Gamma_{(2m,n)}$ is chosen uniformly at random, lemma B.10 implies

$$\Pr\left\{\sigma\in\Gamma_{(2m,n)}\colon d_\nu\left(\langle\vec{h}\rangle_{\mathbf{z}_\sigma(1)},\langle\vec{h}\rangle_{\mathbf{z}_\sigma(2)}\right)>\frac{\alpha}{2}\right\}\leq 2e^{-\frac{\alpha^2\nu mn}{2M}}.$$

Now suppose H is an $\alpha/4$-cover for $\left(\mathcal{H}_{|\mathbf{z}},\boldsymbol{d}_\nu\right)$, and suppose $\sigma\in\Gamma_{(2m,n)}$ is such that

$$d_\nu\left(\langle\vec{h}\rangle_{\mathbf{z}_\sigma(1)},\langle\vec{h}\rangle_{\mathbf{z}_\sigma(2)}\right)>\frac{\alpha}{2}. \tag{44}$$

As H is an $\alpha/4$-cover for $\left(\mathcal{H}_{|\mathbf{z}},\boldsymbol{d}_\nu\right)$, for all $\vec{h}\in\mathcal{H}$, there exists $\vec{h}'(\mathbf{z})\in H$ such that

$$\boldsymbol{d}_\nu\left(\vec{h}(\mathbf{z}),\vec{h}'(\mathbf{z})\right)\leq\alpha/4. \tag{45}$$

Equations 44 and 45, along with lemma B.9, imply,

$$d_\nu\left(\langle\vec{h}'\rangle_{\mathbf{z}_\sigma(1)},\langle\vec{h}'\rangle_{\mathbf{z}_\sigma(2)}\right)>\frac{\alpha}{4}.$$

Thus

$$\Pr\left\{\sigma\in\Gamma_{(2m,n)}\colon\exists\vec{h}\in\mathcal{H}\colon d_\nu\left(\langle\vec{h}\rangle_{\mathbf{z}_\sigma(1)},\langle\vec{h}\rangle_{\mathbf{z}_\sigma(2)}\right)>\frac{\alpha}{2}\right\}$$
$$\leq\Pr\left\{\sigma\in\Gamma_{(2m,n)}\colon\exists\vec{h}'(\mathbf{z})\in\mathcal{H}_{|\mathbf{z}}\colon d_\nu\left(\langle\vec{h}'\rangle_{\mathbf{z}_\sigma(1)},\langle\vec{h}'\rangle_{\mathbf{z}_\sigma(2)}\right)>\frac{\alpha}{4}\right\}$$
$$\leq 2\mathcal{N}\left(\alpha/4,\mathcal{H}_{|\mathbf{z}},\boldsymbol{d}_\nu\right)e^{-\frac{\alpha^2\nu mn}{8M}},$$

the last inequality following from lemma B.10. If $\mathbf{z}\in Z^{(2m,n)}$ is also selected at random according to the measure $\vec{P}$ in the statement of the lemma, then

$$\Pr\left\{\sigma\in\Gamma_{2mn},\mathbf{z}\in Z^{(2m,n)}\colon\exists\vec{h}\in\mathcal{H}\colon d_\nu\left(\langle\vec{h}\rangle_{\mathbf{z}_\sigma(1)},\langle\vec{h}\rangle_{\mathbf{z}_\sigma(2)}\right)>\frac{\alpha}{2}\right\}$$
$$\leq 2\langle\mathcal{N}\left(\alpha/4,\mathcal{H}_{|\mathbf{z}},\boldsymbol{d}_\nu\right)\rangle_{\vec{P}^{2m}}e^{-\frac{\alpha^2\nu mn}{8M}}.$$

As all the components of $\mathbf{z}\in Z^{(2m,n)}$ are selected independently (although not identically) and σ only ever swaps components of $\mathbf{z}$ from "under" the same probability

measure P_i[1], the effect of choosing $\sigma \in \Gamma_{(2m,n)}$ uniformly can be integrated out of the above bound, which can then be written simply as

$$\Pr\left\{\mathbf{z} \in Z^{(2m,n)} : \exists \vec{h} \in \mathcal{H} : d_\nu\left(\langle \vec{h} \rangle_{\mathbf{z}(1)}, \langle \vec{h} \rangle_{\mathbf{z}(2)}\right) > \frac{\alpha}{2}\right\}.$$

The result follows. □

One more lemma is needed to prove the fundamental theorem.

LEMMA B.13. *For all $\alpha > 0$ and all $\mathbf{z} \in Z^{(m,n)}$,*

$$\mathcal{N}\left(\alpha, \mathcal{H}_{|\mathbf{z}}, \boldsymbol{d}_\nu\right) \leq \mathcal{N}\left(\alpha\nu/2, \mathcal{H}, d_{P_\mathbf{z}}\right),$$

where in the right hand side $\mathbf{z}$ should be viewed as an m-fold sample from Z^n and $P_\mathbf{z}$ as the corresponding empirical measure[2] *on Z^n.*

PROOF. By lemmas B.9, B.4 and B.5,

$$\mathcal{N}\left(\alpha, \mathcal{H}_{|\mathbf{z}}, \boldsymbol{d}_\nu\right) \leq \mathcal{N}\left(\alpha\nu/2, \mathcal{H}_{|\mathbf{z}}, \boldsymbol{d}_{L^1}\right),$$

and observe that the map $\psi : (\mathcal{H}, d_{P_\mathbf{z}}) \to \left(\mathcal{H}_{|\mathbf{z}}, \boldsymbol{d}_{L^1}\right)$, where $\psi(\vec{h}) = \vec{h}(\mathbf{z})$ for all $\vec{h} \in \mathcal{H}$, is an isometry. Thus by lemma B.4 again the result follows. □

The proof of the fundamental theorem (B.6) follows immediately from the last two lemmas.

[1] This is the only point where the fact that $\vec{P}$ is a *product* measure on Z^n has been used. However it should be clear that it is crucial in ensuring the argument carries through.

[2] That is, for any set $S \subseteq Z^n$, $P_\mathbf{z}(S) = |S \cap \mathbf{z}|/|\mathbf{z}|$, where $\mathbf{z}$ is interpreted as a subset of Z^n by collecting all its rows together, $\mathbf{z} \equiv \{\vec{z}_1, \ldots, \vec{z}_m\}$.

APPENDIX C

Bounding $\mathcal{C}(\varepsilon, \mathcal{H})$

For the fundamental theorem to be useful bounds need to be found for the capacity $\mathcal{C}(\varepsilon, \mathcal{H})$ when $\mathcal{H}$ consists of classes of functions common in machine learning applications. Haussler (1992) showed how to do this for classes constructed out of compositions and free products of certain simple function classes. His results can be used to determine the capacity of artificial neural networks (see his paper, section 7). One slightly unsatisfactory aspect of his method is that somewhat arbitrary metrics must be introduced on the simple function classes for the analysis to carry through. In this section it is shown that such a restriction is unnecessary: there is a *natural* metric—generated by the function class as a whole—that can be defined for any small function class embedded within a larger composition or product. This leads to simpler and more general formulae than those given in Haussler, section 6. As an application, the capacity of the function space $l_{\mathcal{G}^n \circ \overline{\mathcal{F}}}$ (see section 3.5.3) is calculated in terms of the capacity of $\mathcal{G}$ and $\mathcal{F}$. To begin with some preliminary definitions are needed in order that a fundamental and elegant theorem of Pollard's (1984)—central to Haussler's approach—may be understood.

C.1. Pseudo-Dimension

The capacity of a collection of real-valued functions can be shown to depend ultimately on just a single parameter known as the *pseudo-dimension*. The pseudo-dimension is a real-valued extension of a similar concept called the *VC-dimension* introduced by Vapnik and Chervonenkis (1971) to cover uniform convergence of empirical estimates over Boolean-valued function classes. The pseudo-dimension was named as such by Haussler (1992), but essentially defined by Pollard (1984).

DEFINITION C.1. The notion of quadrant in R^2 and octant in R^3 generalises naturally to the concept of *orthant* in R^d. That is, the orthants in R^d are the regions in which the sign of the coordinates of each point are all the same (take the sign of zero be positive). Clearly there are 2^d orthants in R^d.

DEFINITION C.2. Given a class of functions $\mathcal{H}: Z \to \mathrm{R}$ and any finite subset Y of Z, define $\mathcal{H}_{|Y} \subset \mathrm{R}^{|Y|}$ by (for a similar notion recall definition B.5)

$$\mathcal{H}_{|Y} = \left\{ (h(y_1), \ldots, h(y_{|Y|})) : h \in \mathcal{H} \right\}.$$

DEFINITION C.3. A set of real-valued functions $\mathcal{H}: Z \to \mathrm{R}$ is said to *shatter* a finite set $Y \subseteq Z$ if there exists a *translation* of $\mathcal{H}_{|Y}$ by $x \in \mathrm{R}^{|Y|}$ such that $\mathcal{H}_{|Y} + x$ has a point in each orthant of $\mathrm{R}^{|Y|}$.

Note that "splatter" would be a more appropriate term here as the points of Y are being *spread out* over all orthants by the action of $\mathcal{H}$, however "shatter" appears to have stuck.

DEFINITION C.4. The *pseudo-dimension* of a set of real valued functions $\mathcal{H}: Z \to \mathrm{R}$ is the size of the largest subset $Y \subseteq Z$ shattered by $\mathcal{H}$. If there is no largest such set then set the pseudo-dimension of $\mathcal{H}$ equal to infinity. Following Haussler the pseudo-dimension of $\mathcal{H}$ will be denoted by $\mathrm{dimp}(\mathcal{H})$.

The term pseudo-dimension is justified by the following theorem due to Dudley (1978).

THEOREM C.1. *If $\mathcal{H}$ is a d-dimensional vector space of functions from any set Z into* R, $\mathrm{dimp}(\mathcal{H}) = d$.

Another useful result due to Wenocur and Dudley (1981) :

THEOREM C.2. *Let $\mathcal{F}$ be any family of functions from Z into some interval $I \subseteq \mathrm{R}$. Let h be a fixed monotone function from I into* R *and let $\mathcal{H} = \{h \circ f: f \in \mathcal{F}\}$. Then* $\mathrm{dimp}(\mathcal{H}) \leq \mathrm{dimp}(\mathcal{F})$ *with equality if h is strictly increasing or strictly decreasing.*

The following elegant theorem due to Pollard (1984) illustrates the significance of the pseudo-dimension in machine learning applications. (Recall definitions B.2 and B.3 for the definitions of the packing numbers $\mathcal{M}(\varepsilon, \mathcal{H}, d_P)$ and the pseudo-metric d_P).

THEOREM C.3. *Suppose the family of functions $\mathcal{H}: Z \to [0, M]$ has finite pseudo-dimension, d. Then for all $0 < \varepsilon \leq M$ and any probability measure P on Z,*

$$\mathcal{M}(\varepsilon, \mathcal{H}, d_P) < 2\left(\frac{2eM}{\varepsilon} ln \frac{2eM}{\varepsilon}\right)^d.$$

By lemma B.2, $\mathcal{N}(\varepsilon, \mathcal{H}, d_P) \leq \mathcal{M}(\varepsilon, \mathcal{H}, d_P)$ and so

$$\mathcal{C}(\varepsilon, \mathcal{H}) < 2\left(\frac{2eM}{\varepsilon} ln \frac{2eM}{\varepsilon}\right)^d.$$

Hence if the pseudo-dimension of a class of functions is known, its capacity can immediately be bounded. Thus the capacities of the simple classes mentioned in theorems C.1 and C.2 can be bounded already. Most function classes used in practice–artificial neural networks, radial basis function networks, etc.–are constructed from compositions and products of such simple classes. Techniques for bounding the capacity of products and compositions are introduced below. However before embarking on that project one more section is required introducing Haussler's decision-theoretic formulation of machine learning which enables results bounding capacities to apply in a multitude of different learning scenarios.

C.2. Loss Functions

Without going into too much detail as to why (see section 2.1 for some motivation and Haussler section 1.1 for still more), rather than dealing with a set of maps $\mathcal{H} : Z \to [0, M]$ the following structure is used:

$$\begin{array}{lll} X \xrightarrow{\mathcal{H}} & A & \\ & \times & \xrightarrow{l} [0, M] \\ & Y & \end{array}$$

The space A is called the *action* space and X and Y are called the *input* and *output* spaces respectively. The sample space Z is now defined to be $X \times Y$. l is called the *loss function*. This formulation is derived from statistical decision theory (see for example Keifer (1987)). The idea of this construction is that when provided with an input x, the learner produces some action $a \in A$ and, depending on the "state of the environment" $y \in Y$ corresponding to x, receives a certain *loss*, $l(a, y)$. The learner tries to find a *decision rule* or *hypothesis* $h \in \mathcal{H}$ that minimizes its expected loss. In ordinary classification problems using neural networks Y is typically the set $\{0, 1\}$, A is the interval $[0, 1]$ and l is the mean-squared error $l(y, a) = (y - a)^2$. Minimal (zero) loss then corresponds to a decision rule that is a perfect classifier for the problem. The main purpose of this construction is that it allows results concerning the capacity of $\mathcal{H}$ to apply, with little modification, when $\mathcal{H}$ is used in more general ways than simply as a set of classifiers.

Note that for any function $h \in \mathcal{H}$ a function $l_h : Z \to [0, M]$ may be defined by $l_h(z) = l(h(x), y)$ for all $z = (x, y) \in Z$. Denote the set of all such functions by $l_{\mathcal{H}}$ The next few sections show how to bound the capacity of $l_{\mathcal{H}}$ when $\mathcal{H}$ consists of compositions and products of simpler function spaces.

C.3. Capacity of Compositions

Suppose the function space $\mathcal{H} : X \to A$ is of the form $\mathcal{H} = \mathcal{G} \circ \mathcal{F}$ where the functions in $\mathcal{F}$ map X into some set V, and the functions in $\mathcal{G}$ map from V into A. Denote this structure by

$$X \xrightarrow{\mathcal{F}} V \xrightarrow{\mathcal{G}} A.$$

For any function $g \in \mathcal{G}$ the function $l_g : V \times Y \to [0, M]$ can be defined in analogy with the definition for l_h, i.e. $l_g(v, y) = l(g(v), y)$ for all $(v, y) \in V \times Y$. Definition B.4 may then be used directly to give a definition for the capacity of $l_{\mathcal{G}}$:

$$\mathcal{C}(\varepsilon, l_{\mathcal{G}}) = \sup_{P \in \mathcal{P}_{l_{\mathcal{G}}}} \mathcal{N}\left(\varepsilon, l_{\mathcal{G}}, d_P\right).$$

Note that $\mathcal{P}_{l_{\mathcal{G}}}$ is a set of probability measures on $V \times Y$, rather than $Z = X \times Y$.

To define the capacity of $\mathcal{F}$ an appropriate pseudo-metric for $\mathcal{F}$ needs to be found. This is done by pulling back the loss function so that it is effectively a map from $V \times A$ into $[0, M]$, rather than a map from $Y \times A \to [0, M]$.

Definition C.5. Given the structure $X \xrightarrow{\mathcal{F}} V \xrightarrow{\mathcal{G}} A$ and any probability measure $P \in \mathcal{P}_{l_{\mathcal{G}} \circ \mathcal{F}}$, define the pseudo metric $d_{[P, l_{\mathcal{G}}]}$ on $\mathcal{F}$ by

$$d_{[P, l_{\mathcal{G}}]}(f, f') = \sup_{g \in \mathcal{G}} \int_Z |l_{g \circ f}(z) - l_{g \circ f'}(z)| \, dP(z),$$

for all $f, f' \in \mathcal{F}$.

Its clear that $d_{[P,l_\mathcal{G}]}$ is symmetric and $d_{[P,l_\mathcal{G}]}(f,f) = 0$ for all $f \in \mathcal{F}$. The triangle inequality is also pretty trivial but it is proved here for completeness. For any $f_1, f_2, f_3 \in \mathcal{F}$,

$$\begin{aligned} d_{[P,l_\mathcal{G}]}(f_1, f_2) &+ d_{[P,l_\mathcal{G}]}(f_2, f_3) \\ &= \sup_{g\in\mathcal{G}} \int_Z |l_{g\circ f_1}(z) - l_{g\circ f_2}(z)|\, dP(z) + \sup_{g\in\mathcal{G}} \int_Z |l_{g\circ f_2}(z) - l_{g\circ f_3}(z)|\, dP(z) \\ &\geq \sup_{g\in\mathcal{G}} \int_Z |l_{g\circ f_1}(z) - l_{g\circ f_2}(z)| + |l_{g\circ f_2}(z) - l_{g\circ f_3}(z)|\, dP(z) \\ &\geq \sup_{g\in\mathcal{G}} \int_Z |l_{g\circ f_1}(z) - l_{g\circ f_3}(z)|\, dP(z) \\ &= d_{[P,l_\mathcal{G}]}(f_1, f_3). \end{aligned}$$

Thus $d_{[P,l_\mathcal{G}]}$ is indeed a pseudo-metric.

DEFINITION C.6. For the structure $X \xrightarrow{\mathcal{F}} V \xrightarrow{\mathcal{G}} A$, define the capacity of $\mathcal{F}$ by

$$C_{l_\mathcal{G}}(\varepsilon, \mathcal{F}) = \sup_{P\in\mathcal{P}_{l_{\mathcal{G}\circ\mathcal{F}}}} \mathcal{N}\left(\varepsilon, \mathcal{F}, d_{[P,l_\mathcal{G}]}\right).$$

DEFINITION C.7. For any probability measure $P \in \mathcal{P}_{l_{\mathcal{G}\circ\mathcal{F}}}$ on $X \times Y$ and any function $f\colon X \to V$, define the measure P_f on $V \times Y$ by

$$P_f(S) = P(f^{-1}(S))$$

for any $S \in \sigma_{l_\mathcal{G}}$[1].

LEMMA C.4. *Let $\mathcal{H}\colon X \to A$ be of the form $\mathcal{H} = \mathcal{G} \circ \mathcal{F}$ where $X \xrightarrow{\mathcal{F}} V \xrightarrow{\mathcal{G}} A$. For all $\varepsilon_1, \varepsilon_2 > 0$,*

$$\mathcal{C}(\varepsilon_1 + \varepsilon_2, l_\mathcal{H}) \leq \mathcal{C}_{l_\mathcal{G}}(\varepsilon_1, \mathcal{F})\mathcal{C}(\varepsilon_2, l_\mathcal{G}).$$

PROOF. The idea for this proof comes from Haussler's lemma 8. Fix $P \in \mathcal{P}_{l_{\mathcal{G}\circ\mathcal{F}}}$ and let F be a minimum size ε_1-cover for $(\mathcal{F}, d_{[P,l_\mathcal{G}]})$. By definition $|F| \leq \mathcal{C}_{l_\mathcal{G}}(\varepsilon_1, \mathcal{F})$. For each $f \in F$ let G_f be a minimum size ε_2-cover for $(\mathcal{G}, d_{P_f})$. By definition again, $|G_f| \leq \mathcal{C}(\varepsilon_2, \mathcal{G})$ for each $f \in F$. Let $H = \{g \circ f\colon f \in F \text{ and } g \in G_f\}$. Let l_H be defined in the usual way. Note that $|l_H| \leq |H| \leq \mathcal{C}_{l_\mathcal{G}}(\varepsilon_1, \mathcal{F})\mathcal{C}(\varepsilon_2, \mathcal{G})$ and so the lemma will be proved if l_H can be shown to be an $\varepsilon_1 + \varepsilon_2$-cover of $l_\mathcal{H}$. Hence, given any $l_{g\circ f} \in l_\mathcal{H}$ choose $f' \in F$ such that $d_{[P,l_\mathcal{G}]}(f, f') \leq \varepsilon_1$ and $g' \in \mathcal{G}_{f'}$ such that $d_{P_{f'}}(g, g') \leq \varepsilon_2$. Now,

$$\begin{aligned} d_P(l_{g\circ f}, l_{g'\circ f'}) &= \int_Z |l_{g\circ f} - l_{g'\circ f'}|\, dP(z) \\ &\leq \int_Z |l_{g\circ f} - l_{g\circ f'}| + |l_{g\circ f'} - l_{g'\circ f'}|\, dP(z) \\ &\leq d_{[P,l_\mathcal{G}]}(f, f') + d_{P_{f'}}(l_g, l_{g'}) \\ &\leq \varepsilon_1 + \varepsilon_2. \end{aligned}$$

[1] This definition implicitly assumes f is a measurable function from $\left(X, \sigma_{l_{\mathcal{G}\circ\mathcal{F}}}\right)$ into $\left(V, \sigma_{l_\mathcal{G}}\right)$. This is guaranteed by the definition of $\sigma_{l_\mathcal{G}}$ and $\sigma_{l_{\mathcal{G}\circ\mathcal{F}}}$.

Thus l_H is an $\varepsilon_1 + \varepsilon_2$-cover for $l_{\mathcal{H}}$ and so the result follows. □

For multiple compositions this lemma may easily be extended by induction to give the following lemma which is the corresponding result within this framework to Haussler's lemma 8.

LEMMA C.5. *Let* $\mathcal{H} = \mathcal{H}_q \circ \cdots \circ \mathcal{H}_1$ *and* $\mathcal{G}_i = \mathcal{H}_q \circ \cdots \circ \mathcal{H}_{i+1}$, *for* $1 \leq i \leq q-1$. *For all* $\varepsilon_1, \ldots, \varepsilon_q > 0$ *let* $\varepsilon = \sum_{i=1}^{q} \varepsilon_i$. *Then*

$$\mathcal{C}(\varepsilon, l_{\mathcal{H}}) \leq \mathcal{C}(\varepsilon_q, l_{\mathcal{H}_q}) \prod_{i=1}^{q-1} \mathcal{C}_{l_{\mathcal{G}_i}}(\varepsilon_i, \mathcal{H}_i).$$

C.4. Capacity of Products

In this section the problem of bounding the capacity of a *product* of function spaces is tackled. As in the previous section a lemma is proved bounding the capacity of a product in terms of the capacities of its individual components.

Let $X_1, \ldots, X_n$, $Y_1, \ldots, Y_n$ and $A_1, \ldots, A_n$ be n be separate input, output and action spaces respectively. For each i, $1 \leq i \leq n$, let $l_i \colon Y_i \times A_i \to [0, M_i]$ be a loss function and let $Z_i = X_i \times Y_i$. Define $X = X_1 \times \cdots \times X_n, Y = Y_1 \times \cdots \times Y_n$, $A = A_1 \times \cdots \times A_n$ and $Z = Z_1 \times \cdots \times Z_n$. Define a loss function l on $Y \times A$ by

$$l(\vec{y}, \vec{a}) = \frac{1}{n} \sum_{i=1}^{n} l_i(y_i, a_i),$$

for all $\vec{y} = (y_1, \ldots, y_n) \in Y$ and $\vec{a} = (a_1, \ldots, a_n) \in A$. There is a natural identification between $X \times Y = X_1 \times \cdots \times X_n \times Y_1 \times \cdots \times Y_n$ and $Z = X_1 \times Y_1 \times \cdots \times X_n \times Y_n$ by $(x_1, \ldots, x_n, y_1, \ldots, y_n) \leftrightarrow (x_1, y_1, \ldots, x_n, y_n)$, hence Z can equivalently be written as $Z = X \times Y$. Note also that the range of l is $[0, M]$ where $M = \frac{1}{n} \sum_{i=1}^{n} M_i$.

The function space $\mathcal{H} \colon X \to A$ is assumed to be a subset of a product of function spaces, i.e. $\mathcal{H} \subseteq \mathcal{H}_1 \times \cdots \times \mathcal{H}_n$ where for each i, $1 \leq i \leq n$, $\mathcal{H}_i \colon X_i \to A_i$ and the action of $\vec{h} = (h_1, \ldots, h_n) \in \mathcal{H}$ is defined by

$$\vec{h}(x_1, \ldots, x_n) = (h_1(x_1), \ldots, h_n(x_n)).$$

To tackle the problem in its full generality assume $\mathcal{H}$ has the structure $\mathcal{H} = \mathcal{G} \circ \mathcal{F}$, where $X \xrightarrow{\mathcal{F}} V \xrightarrow{\mathcal{G}} A$ and assume that $\mathcal{G}$ and $\mathcal{F}$ are also subsets of product function spaces: $\mathcal{G} \subseteq \mathcal{G}_1 \times \ldots \mathcal{G}_n$ and $\mathcal{F} \subseteq \mathcal{F}_1 \times \cdots \times \mathcal{F}_n$. Given $\vec{g} \circ \vec{f} \in \mathcal{G} \circ \mathcal{F}$ and a loss function $l = \frac{1}{n} \sum_{i=1}^{n} l_i$ as above, define $l_{\vec{g} \circ \vec{f}}$ in the usual way[2] and let $l_{\mathcal{G} \circ \mathcal{F}}$ denote the set of all such functions.

With these definitions, for all *product probability measures*[3] $\vec{P} = P_1 \times \cdots \times P_n \in \mathcal{P}_{l_{\mathcal{H}}}$, the pseudo-metric $d_{[\vec{P}, l_{\mathcal{G}}]}$ can be defined on $\mathcal{F}$ as in definition C.5,

$$d_{[\vec{P}, l_{\mathcal{G}}]}(\vec{f}, \vec{f}') \sup_{\vec{g} \in \mathcal{G}} \int_Z \left| l_{\vec{g} \circ \vec{f}}(\vec{z}) - l_{\vec{g} \circ \vec{f}'}(\vec{z}) \right| d\vec{P}(\vec{z}).$$

The pseudo-metric $d_{[P_i, l_{\mathcal{G}_i}]}$ on $\mathcal{F}_i$ also follows definition C.5,

$$d_{[P_i, l_{\mathcal{G}_i}]}(f, f') = \sup_{g \in \mathcal{G}_i} \int_{Z_i} |l_{g \circ f}(z) - l_{g \circ f'}(z)| \, dP_i(z),$$

[2] That is, $l_{\vec{g} \circ \vec{f}}(\vec{z}) = l(\vec{g} \circ \vec{f}(\vec{x}), \vec{y})$ where $\vec{z} = (\vec{x}, \vec{y})$. Note that $l_{\vec{g} \circ \vec{f}}(\vec{z}) = \frac{1}{n} \sum_{i=1}^{n} l_i(g_i \circ f_i(x_i), y_i)$.

[3] $\mathcal{P}_{l_{\mathcal{H}}}$ contains more than just product probability measures but they will be ignored.

for all $f, f' \in \mathcal{F}_i$[4].

LEMMA C.6. *Given the function space $\mathcal{H} = \mathcal{G} \circ \mathcal{F}$ with $X \xrightarrow{\mathcal{F}} V \xrightarrow{\mathcal{G}} A$, $\mathcal{F} \subseteq \mathcal{F}_1 \times \cdots \times \mathcal{F}_n$ and $\mathcal{G} \subseteq \mathcal{G}_1 \times \cdots \times \mathcal{G}_n$, then for any pair of functions $\vec{f} = (f_1, \ldots, f_n), \vec{f}' = (f'_1, \ldots, f'_n)$ in $\mathcal{F}$ and any product probability measure $\vec{P} = P_1 \times \cdots \times P_n \in \mathcal{P}_{l_\mathcal{H}}$,*

$$d_{[\vec{P}, l_\mathcal{G}]}(\vec{f}, \vec{f}') \leq \frac{1}{n} \sum_{i=1}^{n} d_{[P_i, l_{\mathcal{G}_i}]}(f_i, f'_i). \tag{46}$$

If instead $\vec{P}$ is product probability measure in $\mathcal{P}_{l_\mathcal{G}}$, then for all $\vec{g} = (g_1, \ldots, g_n)$, $\vec{g}' = (g'_1, \ldots, g'_n) \in \mathcal{G}$,

$$d_{\vec{P}}(\vec{g}, \vec{g}') \leq \frac{1}{n} \sum_{i=1}^{n} d_{P_i}(g_i, g'_i).$$

PROOF. Let $\vec{f}, \vec{f}'$ and $\vec{P}$ be as in the first inequality.

$$\begin{aligned}
&d_{[\vec{P}, l_\mathcal{G}]}(\vec{f}, \vec{f}') \\
&\quad = \sup_{\vec{g} \in \mathcal{G}} \int_Z \left| l_{\vec{g} \circ \vec{f}}(\vec{z}) - l_{\vec{g} \circ \vec{f}'}(\vec{z}) \right| \, d\vec{P}(\vec{z}) \\
&\quad = \sup_{(g_1, \ldots, g_n) \in \mathcal{G}} \int_{z_1 \in Z_1} \cdots \int_{z_n \in Z_n} \left| \frac{1}{n} \sum_{i=1}^{n} \left(l_{g_i \circ f_i}(z_i) - l_{g_i \circ f'_i}(z_i) \right) \right| \, dP_1(z_1) \ldots dP_n(z_n) \\
&\quad \leq \sup_{(g_1, \ldots, g_n) \in \mathcal{G}} \int_{z_1 \in Z_1} \cdots \int_{z_n \in Z_n} \frac{1}{n} \sum_{i=1}^{n} \left| l_{g_i \circ f_i}(z_i) - l_{g_i \circ f'_i}(z_i) \right| \, dP_1(z_1) \ldots dP_n(z_n) \\
&\quad = \frac{1}{n} \sum_{i=1}^{n} \sup_{g_i \in \mathcal{G}_i} \int_{Z_i} \left| l_{g_i \circ f_i}(z) - l_{g_i \circ f'_i}(z) \right| \, dP_i(z) \\
&\quad = \frac{1}{n} \sum_{i=1}^{n} d_{[P_i, l_{\mathcal{G}_i}]}(f_i, f'_i).
\end{aligned}$$

The second inequality is proved in a similar way. □

Recalling definition C.6 for the capacity $\mathcal{C}_{l_\mathcal{G}}(\varepsilon, \mathcal{F})$, the last lemma gives the following lemma.

LEMMA C.7. *Given the function space $\mathcal{H} = \mathcal{G} \circ \mathcal{F}$ with $X \xrightarrow{\mathcal{F}} V \xrightarrow{\mathcal{G}} A$, $\mathcal{F} \subseteq \mathcal{F}_1 \times \cdots \times \mathcal{F}_n$ and $\mathcal{G} \subseteq \mathcal{G}_1 \times \cdots \times \mathcal{G}_n$, then*

$$\mathcal{C}_{l_\mathcal{G}}(\varepsilon, \mathcal{F}) \leq \prod_{i=1}^{n} \mathcal{C}_{l_{\mathcal{G}_i}}(\varepsilon, \mathcal{F}_i)$$

$$\mathcal{C}(\varepsilon, l_\mathcal{G}) \leq \prod_{i=1}^{n} \mathcal{C}(\varepsilon, l_{\mathcal{G}_i}).$$

[4] Strictly speaking $l_{\mathcal{G}_i}$ and $l_{g \circ f}$ should be written as $l_{i\mathcal{G}_i}$ and $l_{i\,g \circ f}$ respectively, however notation like that is just too ugly. Hopefully there should not be any confusion.

PROOF. For the first inequality fix $\vec{P} = P_i \times \cdots \times P_n \in \mathcal{P}_{l_{\mathcal{H}}}$ and let F_i be an ε-cover for $(\mathcal{F}_i, d_{[P_i, l_{\mathcal{G}_i}]})$, $1 \leq i \leq n$. Let $F = F_1 \times \cdots \times F_n$. As $|F| = \prod_{i=1}^{n} |F_i|$ the inequality will be proved if it can be shown that F is an ε-cover for $(\mathcal{F}, d_{[\vec{P}, l_{\mathcal{G}}]})$. Let $\vec{f} = (f_1, \ldots, f_n)$ be any function in $\mathcal{F}$. For each $1 \leq i \leq n$ choose $f_i' \in F_i$ so that $d_{[P, l_{\mathcal{G}_i}]}(f_i, f_i') \leq \varepsilon$. Let $\vec{f}' = (f_1', \ldots, f_n')$. By lemma C.6

$$d_{[\vec{P}, l_{\mathcal{G}}]}(\vec{f}, \vec{f}') \leq \frac{1}{n} \sum_{i=1}^{n} d_{[P_i, l_{\mathcal{G}_i}]}(f_i, f_i') \leq \varepsilon.$$

Thus F is an ε-cover for $\mathcal{F}$. The second inequality is proved similarly. □

C.5. Diagonal Function Spaces

DEFINITION C.8. Given a function space $\mathcal{F}\colon X \to V$, define $\overline{\mathcal{F}} \subseteq \mathcal{F}^n$ by $\overline{\mathcal{F}} = \{(f, \ldots, f) : f \in \mathcal{F}\}$. Denote $(f, \ldots, f)$ by $\bar{f}$. $\bar{f}$ acts on X^n as per usual: $\bar{f}(x_1, \ldots, x_n) = (f(x_1), \ldots, f(x_n))$. For any product probability measure $\vec{P} = P_1 \times \cdots \times P_n$ on Z^n, define the *average* measure $\bar{P}$ on Z by $\bar{P} = \frac{1}{n} \sum_{i=1}^{n} P_i$. For the structure $X \xrightarrow{\mathcal{F}} V \xrightarrow{\mathcal{G}} A$ and for any probability measure $P \in \mathcal{P}_{l_{\mathcal{G} \circ \mathcal{F}}}$, define the pseudo-metric $d^*_{[P, l_{\mathcal{G}}]}$ on $\mathcal{F}$ by

$$d^*_{[P, l_{\mathcal{G}}]}(f, f') = \int_Z \sup_{g \in \mathcal{G}} |l_{g \circ f}(z) - l_{g \circ f'}(z)| \, dP(z)$$

for all $f, f' \in \mathcal{F}$. Define

$$\mathcal{C}^*_{l_{\mathcal{G}}}(\varepsilon, \mathcal{F}) = \sup_{P \in \mathcal{P}_{l_{\mathcal{G} \circ \mathcal{F}}}} \mathcal{N}(\varepsilon, \mathcal{F}, d^*_{[P, l_{\mathcal{G}}]}).$$

LEMMA C.8. *Given the structure* $X^n \xrightarrow{\overline{\mathcal{F}}} V^n \xrightarrow{\mathcal{G}^n} A^n$ *and any product probability measure* $\vec{P} = P_1 \times \cdots \times P_n \in \mathcal{P}_{l_{\mathcal{G}^n \circ \overline{\mathcal{F}}}}$,

$$d_{[\vec{P}, l_{\mathcal{G}^n}]}(\bar{f}, \bar{f}') \leq d^*_{[\bar{P}, l_{\mathcal{G}}]}(f, f')$$

for all $f, f' \in \mathcal{F}$.

PROOF.

$$\begin{aligned} d_{[\vec{P}, l_{\mathcal{G}^n}]}(\bar{f}, \bar{f}') &\leq \frac{1}{n} \sum_{i=1}^{n} d_{[P_i, l_{\mathcal{G}}]}(f, f') \qquad \text{(by lemma C.6)} \\ &= \frac{1}{n} \sum_{i=1}^{n} \sup_{g \in \mathcal{G}} \int_Z |l_{g \circ f}(z) - l_{g \circ f'}(z)| \, dP_i(z) \\ &\leq \frac{1}{n} \sum_{i=1}^{n} \int_Z \sup_{g \in \mathcal{G}} |l_{g \circ f}(z) - l_{g \circ f'}(z)| \, dP_i(z) \\ &= d^*_{[\bar{P}, l_{\mathcal{G}}]}(f, f'). \end{aligned}$$

□

The following lemma follows immediately from this lemma, lemma B.5 and lemma B.4.

LEMMA C.9. *For the structure* $X^n \xrightarrow{\overline{\mathcal{F}}} V^n \xrightarrow{\mathcal{G}^n} A^n$ *and any product probability distribution* $\vec{P} \in \mathcal{P}_{l_{\mathcal{G}^n \circ \overline{\mathcal{F}}}}$ *on* Z^n,

$$\mathcal{C}_{l_{\mathcal{G}^n}}(\varepsilon, \overline{\mathcal{F}}) \leq \mathcal{C}^*_{l_{\mathcal{G}}}(\varepsilon, \mathcal{F}).$$

Now the most important capacity of all can be calculated.

THEOREM C.10. *For the structure*

$$X^n \xrightarrow{\overline{\mathcal{F}}} V^n \xrightarrow{\mathcal{G}^n} A^n,$$

a loss function $l\colon Y \times A \to [0, M]$, *and all* $\varepsilon, \varepsilon_1, \varepsilon_2 > 0$ *such that* $\varepsilon_1 + \varepsilon_2 = \varepsilon$,

$$\mathcal{C}(\varepsilon, l_{\mathcal{G}^n \circ \overline{\mathcal{F}}}) \leq \mathcal{C}(\varepsilon_1, l_{\mathcal{G}})^n \, \mathcal{C}^*_{l_{\mathcal{G}}}(\varepsilon_2, \mathcal{F}).$$

PROOF. The result is immediate from lemmas C.9, C.5 and C.7. □

APPENDIX D

Neural Network bounds.

It is shown how the results of the previous chapter may be used to determine, following Haussler again, the capacity of certain classes of continuous functions that can be defined over *graphs*. As an application the capacity of neural networks is determined as a function of the number of weights in the network, using some more techniques from Haussler (1992). The capacity bounds parallel similar results given in Haussler, in fact his main result (theorem 11) can be derived as a special case.

DEFINITION D.1. A *feedforward continuous function network* is a continuous class of functions defined on a directed acyclic graph. Nodes in the graph with no incoming edges are called *input* nodes and those with no outgoing edges are called *output* nodes. The rest are called *hidden* nodes. The hidden nodes and output nodes are known collectively as *computational* nodes. To each non-input node n is attached a class of continuous functions $\mathcal{H}$ mapping $\mathrm{R}^{\mathrm{indegree}(n)}$ into R where indegree(n) is the number of incoming edges to n. The *depth* of a node is defined to be the length of the longest path from any input node to that node. By composing all the function classes of the graph in the obvious way, the entire graph can be viewed as a class of functions mapping $\mathrm{R}^{n_{\mathrm{in}}}$ into $\mathrm{R}^{n_{\mathrm{out}}}$ where n_{in} and n_{out} are the number of input and output nodes of the graph respectively.

By possibly adding dummy nodes that only compute the identity function, a continuous function network can be viewed as a series of *layers* in which the input nodes lie in the first layer, the output nodes in the last layer and each hidden layer contains nodes that only compute functions of the outputs of the nodes in the immediately preceeding layer. Hence the class of functions $\mathcal{H}$ computed by the network may be viewed as a composition $\mathcal{H} = \mathcal{H}_{\mathrm{out}} \circ \mathcal{H}_k \circ \cdots \circ \mathcal{H}_1$ where k is the depth of the deepest hidden node in the network, $\mathcal{H}_i$ is the space of functions computed by layer k and $\mathcal{H}_{\mathrm{out}}$ is the space of functions formed by the output nodes. Denoting the number of computational nodes in hidden layer i by n_i and the number of dummy nodes by m_i, the entire structure of the network can be written

$$\mathrm{R}^{n_{\mathrm{in}}} \xrightarrow{\mathcal{H}_1} \mathrm{R}^{n_1+m_1} \xrightarrow{\mathcal{H}_2} \mathrm{R}^{n_2+m_2} \longrightarrow \cdots \longrightarrow \mathrm{R}^{n_{k-1}+m_{k-1}} \xrightarrow{\mathcal{H}_k} \mathrm{R}^{n_k+m_k} \xrightarrow{\mathcal{H}_{\mathrm{out}}} \mathrm{R}^{n_{\mathrm{out}}}.$$

To apply the formulae of the previous section (particulary lemma C.5), assume that there is some loss function $l\colon \mathrm{R}^{n_{\mathrm{out}}} \times Y \to \mathrm{R}$ measuring the network's performance.

Lemma C.5 immediately yields

$$\mathcal{C}\left(\varepsilon, l_{\mathcal{H}}\right) \leq \mathcal{C}(\varepsilon_{\text{out}}, l_{\mathcal{H}_{\text{out}}}) \prod_{i=1}^{k} \mathcal{C}_{l_{\mathcal{G}_i}}\left(\varepsilon_i, \mathcal{H}_i\right), \tag{47}$$

where $\mathcal{G}_i = \mathcal{H}_{\text{out}} \circ \mathcal{H}_k \cdots \circ \mathcal{H}_{i+1}$ for $1 \leq i \leq k$, $\varepsilon = \varepsilon_{\text{out}} + \sum_{i=1}^{k} \varepsilon_i$ and $\varepsilon_i > 0$ for all $1 \leq i \leq k$ and $\varepsilon_{\text{out}} > 0$.

To determine the capacity of each of the hidden layers $\mathcal{H}_i$ and the output layer $\mathcal{H}_{\text{out}}$ the following ideas from Haussler may be used. Consider the general problem of determining the capacities of $\mathcal{F}$ and $\mathcal{G}$ in the structure

$$\mathrm{R}^p \xrightarrow{\mathcal{F}} \mathrm{R}^q \xrightarrow{\mathcal{G}} \mathrm{R}^r.$$

To begin with, for any function space $\mathcal{H}\colon \mathrm{R}^p \to \mathrm{R}^q$ and any probability measure P on R^p, define the pseudo-metric $d_{L^1(P)}$ on $\mathcal{H}$ by,

$$d_{L^1(P)}(h, h') = \int_{\mathrm{R}^p} |h(x) - h'(x)| dP(x),$$

for all $h, h' \in \mathcal{H}$. Let

$$\mathcal{C}\left(\varepsilon, \mathcal{H}, d_{L^1}\right) = \sup_P \mathcal{N}\left(\varepsilon, \mathcal{H}, d_{L^1(P)}\right),$$

where the supremum is over all probablity measures P on R^p[1]. Next, given a loss function $l\colon Y \times A \to [0, M]$, define the pseudo-metric ρ_l on A by

$$\rho_l(a, b) = \sup_{y \in Y} |l(y, a) - l(y, b)|$$

for all $a, b \in A$. If the action space A is a subset of Euclidean space of some dimension, then for many common machine learning scenarios it can be shown that there exists a constant $c_l > 0$ such that $\rho_l \leq c_l L^1$ where L^1 is the usual L^1 metric on A (see Haussler, section 7). Assuming such a bound holds, it is then trivial to show for $\mathcal{G}$ in the above structure that

$$\mathcal{C}(\varepsilon, l_{\mathcal{G}}) \leq \mathcal{C}\left(\frac{\varepsilon}{c_l}, \mathcal{G}, d_{L^1}\right). \tag{48}$$

Furthermore, if $b_{\mathcal{G}} > 0$ is a uniform Lipschitz bound on $\mathcal{G}$ (that is, $L_1(g(u), g(v)) \leq b_{\mathcal{G}} L^1(u, v)$ for all $g \in \mathcal{G}$ and $u, v \in \mathrm{R}^q$), then the following bounds can be shown to hold for $\mathcal{F}$:

$$\mathcal{C}_{l_{\mathcal{G}}}\left(\varepsilon, \mathcal{F}\right) \leq \mathcal{C}\left(\frac{\varepsilon}{c_l b_{\mathcal{G}}}, \mathcal{F}, d_{L^1}\right) \tag{49}$$

$$\mathcal{C}^*_{l_{\mathcal{G}}}\left(\varepsilon, \mathcal{F}\right) \leq \mathcal{C}\left(\frac{\varepsilon}{c_l b_{\mathcal{G}}}, \mathcal{F}, d_{L^1}\right). \tag{50}$$

The above formulae may now be directly applied to (47) to yield

$$\mathcal{C}\left(\varepsilon, l_{\mathcal{H}}\right) \leq \prod_{i=1}^{k+1} \mathcal{C}\left(\frac{\varepsilon_i}{c_l b_{\mathcal{G}_i}}, \mathcal{H}_i, d_{L^1}\right), \tag{51}$$

[1] To be definite, assume all functions in $\mathcal{H}$ are measurable with respect to the Borel σ-algebras on R^p and R^q and take the supremum in the definition of $C\left(\varepsilon, \mathcal{H}, d_{L^1}\right)$ to be over all probability measures on the Borel σ-algebra on R^p.

where $b_{\mathcal{G}_i}$ is a uniform Lipschitz bound for each $\mathcal{G}_i = \mathcal{H}_{k+1} \circ \cdots \circ \mathcal{H}_{i+1}, 1 \leq i \leq k+1$. The formula has been simplified by writing $\mathcal{H}_{k+1}$ for $\mathcal{H}_{\text{out}}$, setting $b_{\mathcal{G}_{k+1}} = 1$ and $\varepsilon_{k+1} = \varepsilon_{\text{out}}$.

Let $\mathcal{F}_1, \ldots, \mathcal{F}_p$ be p classes of functions mapping R^q into R. Define the *free product* of $\mathcal{F}_1$ through $\mathcal{F}_p$ to be the class of functions

$$\mathcal{F} = \{(f_1, \ldots, f_p) \colon f_i \in \mathcal{F}_i, 1 \leq i \leq p\},$$

where $(f_1, \ldots, f_p) \colon \mathrm{R}^q \to \mathrm{R}^p$ is the function defined by

$$(f_1, \ldots, f_p)(x) = (f_1(x), \ldots, f_p(x)).$$

Note that the free product differs from the product used elsewhere in the thesis which has always been the Cartesian product. Observe that each layer $\mathcal{H}_i$ in a continuous function network is in fact a free product of n_i computational nodes and m_i identity functions. Haussler's lemma 7 states that the capacity of a free product is less than the product of the capacities of its individual components, which along with the fact that the capacity of the identity is one[2] gives

$$\mathcal{C}(\varepsilon, l_{\mathcal{H}}) \leq \prod_{i=1}^{k+1} \prod_{j=1}^{n_i} \mathcal{C}\left(\frac{\varepsilon_i}{c_l b_{\mathcal{G}_i}}, \mathcal{H}_{ij}, d_{L^1}\right) \tag{52}$$

where $\mathcal{H}_{ij}$ is the function class computed by the j'th node in layer i.

Equation (52) applies to any continuous function network whatsoever and there is no further reduction to be made without knowing more about the function classes $\mathcal{H}_{ij}$. In the case of neural networks, $\mathcal{H}_{ij}$ is often formed by composing a fixed monotone "squashing" function with any linear combination of its inputs. Without weakening the following results it may be assumed that $\mathcal{H}_{ij}$ is the composition of a monotone function with *any* finite dimensional vector space of functions (not just those that can be formed by linear combinations of the inputs). Denoting the dimension of this space by W_{ij} and assuming that the range of all functions in $\mathcal{H}_{ij}$ is $[0, M]$, theorems C.1, C.2 and C.3 may be applied to the capacity of $\mathcal{H}_{ij}$ to give

$$\mathcal{C}\left(\frac{\varepsilon_i}{c_l b_{\mathcal{G}_i}}, \mathcal{H}_{ij}, d_{L^1}\right) \leq 2\left[\frac{c b_{\mathcal{G}_i}}{\varepsilon_i} \ln \frac{c b_{\mathcal{G}_i}}{\varepsilon_i}\right]^{W_{ij}}$$

where $c = 2eMc_l$. Substituting this into (52), setting $W_i = \sum_{j=1}^{n_i} W_{ij}$ (the total number of "parameters" in layer i) and $W = \sum_{i=1}^{k+1} W_i$ (the total number of parameters altogether), yields

$$\mathcal{C}(\varepsilon, l_{\mathcal{H}}) \leq 2^W \prod_{i=1}^{k+1} \left[\frac{c b_{\mathcal{G}_i}}{\varepsilon_i} \ln \frac{c b_{\mathcal{G}_i}}{\varepsilon_i}\right]^{W_i}. \tag{53}$$

To obtain the best possible bound, $\varepsilon_1, \ldots, \varepsilon_{k+1}$ should be chosen so that the right hand side of (53) is minimal, subject to the condition $\sum_{i=1}^{k+1} \varepsilon_i = \varepsilon$. Using Lagrange multipliers, each ε_i can be shown to satisfy

$$\frac{W_i}{\varepsilon_i}\left(1 + \frac{1}{\ln \frac{c b_{\mathcal{G}_i}}{\varepsilon_i}}\right) = \lambda.$$

[2] As is the capacity of any class consisting of just a single function.

Although this cannot be solved exactly for ε_i[3], a reasonable approximation to this system of equations for small ε_i is

$$\frac{W_i}{\varepsilon_i} = \lambda.$$

Using $\sum_{i=1}^{k+1} \varepsilon_i = \varepsilon$ gives,

$$\varepsilon_i = \frac{W_i \varepsilon}{W}.$$

Hence (53) becomes

$$\mathcal{C}(\varepsilon, l_{\mathcal{H}}) \leq 2^W \prod_{i=1}^{k+1} \left[\frac{c b_{\mathcal{G}_i} W}{W_i \varepsilon} \ln \frac{c b_{\mathcal{G}_i} W}{W_i \varepsilon} \right]^{W_i}. \tag{54}$$

Denoting the average Lipschitz bound of the nodes in layer i by b_i, observe that each Lipschitz bound $b_{\mathcal{G}_i}$ must be less than or equal to $\prod_{j=i+1}^{k+1} b_j$. Thus if $b_i > 1$ for each i then all $b_{\mathcal{G}_i}$ are bounded by $\prod_{j=2}^{k+1} b_j$. If all the layers in the network have the same number of parameters then $W/W_i = k + 1 = d$—the *depth* of the network. Using these assumptions and the fact that $2 \ln x < x$, formula (54) becomes

$$\mathcal{C}(\varepsilon, l_{\mathcal{H}}) \leq \left[\frac{2eMd \prod_{j=2}^{d} b_j}{\varepsilon} \right]^{2W}$$

which is identical to the formula in Haussler's theorem 11 (if one replaces $c_2 - c_1$ in his formula by M).

One problem about the bounds in this section is their dependence on Lipschitz bounds. By deriving similar bounds for hard-limiting neural networks (which don't have Lipschitz bounds), Baum and Haussler (1989) showed that such dependence is not always necessary. It remains an open problem to determine precisely the conditions under which the Lipschitz bounds can be ignored.

[3] At least not to the knowledge of the author.

Bibliography

The numbers in square brackets following each bibliography entry are the pages on which the entry is cited.

[1] Baum E. and Haussler D. What size net gives valid generalization? *Neural Comput.*, 1:151–160, 1989. [110].

[2] Cover T. M. and Thomas J. A. *Elements of Information Theory*. John Wiley & Sons, Inc., New York, 1991. [27,73,74].

[3] Dudley R. M. Central limit theorems for empirical measures. *Ann. Probab.*, 6:899–929, 1978. [100].

[4] Dudley R. M. *Real Analysis and Probability*. Wadsworth & Brooks/Cole, California, 1989. [87,88]

[5] Geman S., Bienenstock E. and Doursat R. Neural networks and the bias/variance dilemma. *Neural Comput.*, 4:1–58, 1992. [5]

[6] Haussler D. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Inform. Comput.*, 100:78–150, 1992. [Everywhere]

[7] Haussler D., Kearns M., Seung H. Sebastian and Tishby N. Rigorous learning curve bounds from statistical mechanics. Deposited in the Neuroprose archive., 1994. [39]

[8] Keifer J. C. *Introduction to Statistical Inference*. Springer-Verlag, New York, 1987. [101]

[9] Le Cun Y., Boser B., Denker J. S., Henderson D., Howard R. E., Hubbard W. and Jackel L. D. Backpropagation applied to handwritten zip-code recognition. *Neural Comput.* 1:541-551. [67]

[10] Pollard D. *Convergence of Stochastic Processes*. Springer-Verlag, New York, 1984. [19,87,89,99,99,100]

[11] Rumelhart D. E., Hinton G. E. and Williams R. J. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. [54]

[12] Valiant L. G. A theory of the learnable. *Comm. ACM*, 27:1134–1142, 1984. [10,15,20]

[13] Vapnik V. N. and Chervonenkis A. Ya. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971. [99]

[14] Vapnik V. N. *Estimation of Dependencies based on Empirical Data*. Springer-Verlag, New York, 1982. [28]

[15] Weiss Y. and Edelman S. Representation with receptive fields: gearing up for recognition. Technical Report CS-TR 93-09, Weizmann Institute, 1993. [25]

[16] Wenocur R. S. and Dudley R. M. Some special Vapnik-Chervonenkis classes. *Discrete Math.*, 33:313–318, 1981. [100]