Diversity in Neural Network Ensembles

Gavin Brown

A thesis submitted
to The University of Birmingham
for the degree of Doctor of Philosophy

School of Computer Science
The University of Birmingham
Birmingham B15 2TT
United Kingdom

January 2004

Abstract

We study the issue of error diversity in ensembles of neural networks. In ensembles of regression estimators, the *measurement* of diversity can be formalised as the Bias-Variance-Covariance decomposition. In ensembles of classifiers, there is no neat theory in the literature to date. Our objective is to understand how to precisely define, measure, and create diverse errors for both cases. As a focal point we study one algorithm, *Negative Correlation (NC) Learning* which claimed, and showed empirical evidence, to *enforce* useful error diversity, creating neural network ensembles with very competitive performance on both classification and regression problems. With the lack of a solid understanding of its dynamics, we engage in a theoretical and empirical investigation.

In an initial empirical stage, we demonstrate the application of an evolutionary search algorithm to locate the optimal value for λ, the configurable parameter in NC. We observe the behaviour of the optimal parameter under different ensemble architectures and datasets; we note a high degree of unpredictability, and embark on a more formal investigation.

During the theoretical investigations, we find that NC succeeds due to exploiting the *Ambiguity decomposition* of mean squared error. We provide a grounding for NC in a statistics context of bias, variance and covariance, including a link to a number of other algorithms that have exploited Ambiguity. The discoveries we make regarding NC are not limited to neural networks. The majority of observations we make are in fact properties of the mean squared error function. We find that NC is therefore best viewed as a *framework*, rather than an algorithm itself, meaning several other learning techniques could make use of it.

We further study the configurable parameter in NC, thought to be entirely problem-dependent, and find that one part of it can be analytically determined for any ensemble architecture. We proceed to define an upper bound on the remaining part of the parameter, and show considerable empirical evidence that a lower bound also exists. As the size of the ensemble increases, the upper and lower bounds converge, indicating that the optimal parameter can be determined exactly. We describe a number of experiments with different datasets and ensemble architectures, including the first comparisons to other popular ensemble methods; we find NC to be a competitive technique, worthy of further application.

Finally we conclude with observations on how this investigation has impacted our understanding of diversity in general, and note several possible new directions that are suggested by this work. This includes links to evolutionary computation, Mixtures of Experts, and regularisation techniques.

Acknowledgements

Thanks firstly go to my supervisor, the man who taught me to open the black box of AI and scrutinise the contents, who generously not only educated me in research but also in the art of pedagogy, who turned out to be a good mate too. Thanks Jeremy!

In addition to Jeremy, much gratitude is due to several learned others who were always willing to impart valuable scholastic nuggets, among whom are Xin Yao, Ela Claridge, Peter Tiňo, and Lucy Kuncheva.

I had the good fortune to get a couple of great officemates who turned into even better friends. Mark Roberts, a pal and a half, constantly amusing me, constantly challenging me, whether it was intellectually or just to an elastic band fight. Along with Mark, there was John "I could tell you a story about that" Woodward, a wealth of life experience and laughs, painstakingly educating me in mathematics, eternally generous with his time. Thanks guys, for all the stimulating debates, coffee breaks, life advice, my mathematics education, elastic bands, and general procrastination opportunities.

Aside of these two, I have benefited from numerous fellow PhD students, who luckily for me were wise beyond their years and helped me along the way: Elliot Smith, Rachel Harris, Adrian Hartley, Tim Kovacs and Stuart Reynolds to name but a few. Portions of the taxonomy presented in section 3.2 were written in cooperation with Rachel Harris.

To my oldest Birmingham buddies, always willing to provide me with a weekend away from the academic bubble, James Coltman and Steve Greene. The term "indispensable life-long friends" doesn't even come close.

To Leanne Sims, for support, confidence, and laughs, dragging me up through undoubtedly the most difficult time in this process. Though it looked like I never really took in what you were saying, I did, and without that wouldn't have made it. Thankyou.

To Lauren Foster, a great friend, laughs galore and inspiration abound, for introducing me to my secondary career in ComAc; and of course many thanks to all there, especially the kids of SOPS and DYC for keeping me sane, or at least on the edge.

Thanks also for countless enjoyable experiences over my time in Birmingham, to housemates, peers, and pals: Dave Fitz, Gareth, Iñigo & Egus, Hannah, Dean, James, Vineet, Dave G, Dave B, Padma, The Sims Family, Jack, Iain, Jon, and Cynth.

Finally, over all, this thesis is for my Mum, my Dad, and Ellen. Family is the one sure thing in life, and quite simply they put me where I am today.

-

Contents

Chapter 1

Introduction

When important decisions have to be made, society often places its trust in groups of people. We have parliaments, juries, committees, and boards of directors, whom we are happy to have make decisions for us. It is generally accepted that "two heads are better than one", and that bad choices can be avoided by encouraging people with different viewpoints to reach mutually agreeable decisions. The Condorcet Jury Theorem [156] proposed by the Marquis of Condorcet in 1784, studied the conditions under which a democracy as a whole is more effective than any of its constituent members. In 1818, Laplace [30] observed that a suitable combination of two probabilistic methods will perform better than either of the components. From these first insights, a number of issues have arisen which have been studied extensively by a number of different research groups, among which is the Machine Learning community, the target audience for this thesis.

In this chapter, we will firstly attempt to convey the motivation for the work contained in this thesis, including an outline of the problem to be addressed. This will be expanded upon later in the literature review. We will then state the thesis questions clearly, including reasons why they are important, followed by a chapter-by-chapter synopsis of the thesis contents.

1.1 Learning Machines

The notion of a *learning machine*, a system that is able to teach itself to solve a given task, is highly attractive. Even more attractive is the idea of systems that could automatically decompose very difficult tasks into smaller, more manageable sub-tasks, which could be solved using standard techniques. Such systems, the subject of so-called *meta-machine learning*, have received a lot of well-deserved attention in the past decade.

In one particular branch of meta-machine learning, some researchers have taken the approach of using multiple learners to solve *the same problem*, and combining their decisions in some manner. The term for these groups of learners is an *ensemble* or *committee machine*. There are many different ways to learn the machines that make up the ensemble, and many different ways to combine the decisions of those machines. The tie that binds all the techniques studied in this thesis is that the machines *are all attempting to solve the same problem.*

The term 'problem' refers to a task which in fact presents itself in a number of different guises, requiring the predictor to be able to provide decisions on several different instances of the same problem; for example learning to drive a car on a road, on a dirt track, on ice, or any other situation. Though they are all attempting to solve the same problem, each predictor within the ensemble is likely to have different strengths and weaknesses in different situations. A key challenge for ensemble research is how to 'manage the recognised limitations of the individual predictors' [124]. The word 'limitations' here refers to the situations in which the learner will not perform as well as we hope, i.e. when it will make errors. It is important to note, in a committee of *people*, the 'correct' decision, by which we would measure such errors, is often not known at the time of the decision-making; only time will tell whether an error was made. In machine learning however, we very often do have access to what is deemed the 'correct' choice for a small set of 'training' data. We would like to build a system from this information so that we can predict the correct choice in future situations that we have not yet encountered. This is known as *supervised learning* and will be discussed in section 2.1.

Given that each learner will have these 'limitations'—that it will make errors in some situation or another—the hope is that the strengths and weaknesses of the ensemble members can be recognised and managed, leading to the correct decision being taken overall. Several empirical results [66, 115, 116, 149] have supported the widespread view that for best performance on a task, the individuals should exhibit "diverse errors". The intuition here is that with a committee of people making decisions, we would not want them to all make the same bad judgements *at the same time.* However in order to implement these ideas in a computer system, they need to be far more formalised. What exactly do we mean by "diverse" errors? The answer to this question depends critically on whether our predictor outputs are real-valued numbers, or class labels. A real-valued number will be output if the problem is a *regression*, say for example predicting 83.74kg as the weight of a person given their height and shoe-size. A class label will be output if the problem is a *classification*, for example predicting YES as the guess of whether the person is over 80.0kg or not. The former situation, regression, is reasonably well understood (see section 3.1.1), however when "diversity" is referred to it is usually intended to imply the latter, *classification error diversity.* Intuitively, we know that the predictors (in this case, classifiers) should make different errors, yet formalising *how* they should do this has proved a difficult challenge. Though the predictors may exhibit very different errors, they may have sacrificed individual accuracy in order to do so; this shows that training an ensemble is a balancing act between error diversity and individual accuracy. This issue will be addressed in more detail in chapter 3.

We now make a distinction between two types of ensemble learning methods. Some researchers have taken the approach of incorporating a stochastic element during construction of the ensemble, randomly perturbing a learner's trajectory in the space of possible predictors, so that hopefully a group of "diverse" predictors will emerge at the end. Another approach is to deterministically *enforce* a measure deemed representative of error diversity. This distinction applies to both regression and classification problems, and we will discuss it more in section 3.2. It is this issue of how to understand and enforce error diversity (or more precisely, how to balance diversity against accuracy) that provides the focus for this thesis. The next obvious question is, *why?* What is our motivation to study such issues?

1.2 Why study Ensembles?

Ensemble learning is an appealing research topic for many reasons. For one, ensemble methods can be incredibly simple to implement, yet have such complex underlying dynamics that hundreds of hours can be spent in their study. A prime example of this is the *Bagging* algorithm [13] (see section 2.3.1). One of the most widely used techniques, Bagging can be implemented in just a few lines of code, yet has seen several researchers disputing the root of its success [20, 22, 32, 41, 110].

The wide applicability of these techniques is also an attractive prospect. The particular technique studied in this thesis, *Negative Correlation Learning* [85] can be applied to *any* function approximator that can minimise a quadratic error function. Many of the ensemble methods developed over the last decade of research can and will be applied to new learning techniques for many years to come. Working at such an abstracted level allows us to exist, interact and contribute at the junction of several disciplines: artificial intelligence, financial forecasting, and statistics to name but a few. As the latest machine learning fashions come and go, so ensemble techniques continue to be applied. From the current en vogue *Support Vector Machines* [142, 143], to the more established *Hidden Markov Models* [52], people find ways to extract that little bit more information with an ensemble technique.

The study of *why* these ensemble methods succeed is of fundamental importance. If we understand precisely why, when, and how particular ensemble methods can be applied successfully, we will have made progress toward a powerful new tool for machine learning: the ability to automatically exploit the strengths and weaknesses of different learning systems. The issue of diversity is therefore a topic of great draw and depth, offering wide-ranging implications and a fascinating community of researchers to work within. To contribute meaningful knowledge in such an interdisciplinary subject, a focal point for study within it must be quickly identified.

1.3 What is this Thesis about?

1.3.1 Thesis Questions

This thesis studies one specific ensemble learning technique, *Negative Correlation (NC) Learning.* NC has shown several empirical successes [18, 85, 86, 94, 152], often outperforming other ensemble learning methods on a variety of tasks, but as yet has had no explanations offered as to *why* it performs well when it does. As such, the primary thesis question is *"Why and how does NC Learning succeed as an ensemble method?"*. NC has a problem-dependent parameter, λ, which determines the amount of diversity among the ensemble members. This parameter can determine how well NC performs, yet currently there is little information on how to set it. A secondary thesis question is therefore *"How can we provide guidance for setting λ in the NC learning algorithm?"*.

These are important questions to ask, for two reasons. The first is from the viewpoint of NC as a stand-alone tool: anything that helps us understand its behaviour or configure its parameters, is useful. The second reason is from the viewpoint of gaining a better understanding of ensemble classification error "diversity". NC is a technique that was claimed to explicitly enforce this type of diversity; in his thesis Liu [85] shows that a lower correlation between classification errors is observed when using NC. Understanding why NC works should clarify, at least partially, what "diversity" means. If we can formalise "diversity", and then control it, we may be able to engineer better performing ensemble systems. A third thesis question is therefore *"Can we identify a good way to understand classifier error diversity, and can we explicitly control it?"*.

Our strategy for answering these questions is to engage in a study of the literature concerned with "diversity", combined with a theoretical and empirical analysis of NC learning.

1.3.2 Contributions of the Thesis

The main contribution of the thesis is a better understanding of the dynamics of NC learning. We first summarise these, then summarise how parts of this thesis have contributed to a wider range of the community. All of these will be discussed in more detail in the Conclusions chapter.

Contributions to the Understanding of NC Learning

- Empirical work demonstrating the use of an evolutionary algorithm to find the optimal NC strength parameter (chapter 4).
- A formal grounding for NC, explaining *why* it works in a statistics context. This also relates NC to a large body of literature, including identifying a family of techniques that all utilise the same Ambiguity metric [70] during their ensemble learning procedures (chapter 5).
- Derivation of an upper bound on the NC strength parameter (chapter 5).
- A thorough analysis of the error gradient during learning with NC. We identify four distinct gradient components which are present in the error of a simple ensemble, an NC learning ensemble, and a single estimator. We observe that NC learning succeeds because it can blend smoothly between the two extremes of a fully parallel ensemble and a single large estimator (chapter 5).
- Empirical work supporting the theoretical upper bound; also providing a characterisation of how the optimal strength parameter behaves under changes to the ensemble architecture (chapter 6).
- Empirical evidence suggesting that a lower bound exists for the strength parameter. As the ensemble size increases, the empirical lower bound is constant, while the proven upper bound converges down towards it. In practice, it seems with an ensemble size about 20 or more, the optimal parameter can be determined exactly (chapter 6).
- The first empirical comparison of NC to two of the most popular ensemble learning techniques, Boosting and Bagging (chapter 6).

Contributions in a Wider Scope

- A review of the literature on creating diversity in regression and classification ensembles. This includes suggestions on a possible new taxonomy of the different methods used to create error diversity (chapter 3).

- An alternative, simplified proof of the Ambiguity decomposition (chapter 3).
- An explanation of the exact relationship between Ambiguity and the Bias-Variance-Covariance decomposition (chapter 3).
- Though we could not pursue it, as it is outside the scope of the thesis, we present detailed suggestions on which directions may prove fruitful in understanding how to quantify *classification* error diversity (chapter 7).
- An observation that Negative Correlation learning [85] provides the missing link that unites ensembles, Mixtures of Experts and the Dyn-Co algorithm [48]. Though it is not entirely clear how the dynamics interact, we will show evidence that a smooth space of algorithms exists between these three systems (chapter 7).

1.3.3 Structure of the Thesis

In chapter 2, we formalise the challenge of building learning systems as the *Supervised Learning Problem*, and introduce one class of learning system, *neural networks*. In section 2.2 we review current techniques for combining the outputs of a set of predictors. In section 2.3 we review the most commonly used architectures for learning the predictors that are to be combined. This chapter identifies the wider body of literature to which this thesis contributes.

In chapter 3, we review the literature specifically relating to "diversity". We first attempt to explain precisely what is meant by this term in both regression and classification domains, and identify problems with formalising the concept in the classification domain. We then review several methods for enforcing diversity in an ensemble of predictors. In an overall effort to understand and create "diversity", we focus for the remainder of the thesis on one technique, Negative Correlation (NC) Learning [85], which has claimed and shown empirical evidence that it enforces the creation of a diverse ensemble using a regularisation term in the error function. We describe NC, noting the lack of a solid understanding of its dynamics, including how to set the regularisation strength parameter λ.

In chapter 4, we use an evolutionary algorithm to automatically set the λ parameter in NC; we present empirical data for the behaviour of the optimal parameter with various ensemble configurations and datasets. We find the optimal parameter quite unpredictable, and suggest this may be resolved with a more formal understanding of the algorithm.

In chapter 5, we perform a theoretical analysis of NC learning. We find that due to an assumption made in the original work, the λ parameter in fact is made up of two components, one of which can be determined analytically for any ensemble architecture. We give a clear account of the connections between NC, simple ensemble learning, single network learning, the Ambiguity decomposition, and the bias-variance and bias-variance-covariance decompositions. This gives NC a statistical interpretation, including a grounding in a large body of literature, which it did not previously have. We prove an upper bound on the strength parameter and show how the ensemble gradient can be understood as a sum of various components.

In chapter 6, we present a study supporting the theoretical work developed in chapter 5. We show empirical evidence suggesting that a lower bound exists for the strength parameter, and several experiments showing how the optimal value changes with ensemble architecture and noisy data, including a short comparison of NC with other popular ensemble techniques.

In chapter 7 we conclude the thesis, summarising what we have learnt about NC learning and how this has contributed to our understanding of diversity in general. In addition we summarise several future studies that are suggested by these investigations.

1.3.4 Publications Resulting from the Thesis

The work resulting from these investigations has been published in a number of papers:
[17] Gavin Brown and Jeremy Wyatt
"The Use of the Ambiguity Decomposition in Neural Network Ensemble Learning Methods", Twentieth International Conference on Machine Learning. Washington DC, August 2003.

[16] Gavin Brown and Jeremy Wyatt
"Negative Correlation Learning and The Ambiguity Family of Ensemble Methods", Fourth International Workshop on Multiple Classifier Systems. Guildford, UK, June 2003

[19] Gavin Brown, Xin Yao, Jeremy Wyatt, Heiko Wersing and Bernhard Sendhoff
"Exploiting Ensemble Diversity for Automatic Feature Extraction", 9th International Conference on Neural Information Processing, Singapore, 2002.

[18] Gavin Brown and Xin Yao
"On The Effectiveness of Negative Correlation Learning", First UK Workshop on Computational Intelligence, Edinburgh, September 2001.

[152] Xin Yao, Manfred Fischer and Gavin Brown
"Neural Network Ensembles and Their Application to Traffic Flow Prediction in Telecommunications Networks", International Joint Conference on Neural Networks, Washington DC, July 2001

Chapter 2

Combinations of Learning Machines

In the previous chapter, we introduced the concept of a computer system that could learn how to perform a task by presentation of example inputs and desired outputs. In section 2.1 we formalise the challenge of creating such systems as the *Supervised Learning Problem*, and introduce one class of learning system, *neural networks*. Several authors [66, 115, 116] have noted that significant performance improvements can be obtained by creating a *group* of learning systems, and combining their outputs in some fashion. In section 2.2 we review current techniques for combining the outputs of a set of given predictors. In section 2.3 we review techniques specifically for learning the predictors that are to be combined.

2.1 Supervised Learning and Neural Networks

Imagine we are building a security system for a new hi-tech building. We would like to build in a face-recognition system, so the building can choose whether to admit or not admit certain personnel to a top secret area when they arrive at the door. The problem is, a person naturally changes their appearance from day to day; how can our system make allowances for this? Also, if we have several members of staff, how are we going to create the system in the first place? It is not a desirable prospect to sit for many hours meticulously describing each member of staff to the computer system. An alternative would be to provide example pictures of the staff members, and have it *learn* for itself what they look like, and how to compensate for when they change appearance slightly. We should also provide it with at least a few examples of people who are *not* allowed into the secret area, so we can be sure it will refuse admittance to non-authorised staff. This type of problem for a learning system is known as a *classification problem*, the answer it provides *classifies* each input into different categories. Another problem would be to predict the weight in kilograms of the person from a photograph; this provides a real-valued output, for example 80.53kg or 67.901kg. This is a *regression problem*.

Our security system is only one of many possible tasks for a learning system. Others may be for example to predict the amount of snow that will fall tonight given the current weather conditions, or to predict the average house price in an area given some demographic information (in fact, exactly this, the house pricing problem, is tackled in chapter 4). We need a general mathematical framework that will encompass all of these and enable us to discuss them in a formal manner; we develop this in the following section.

2.1.1 Error Measures

We will first consider the formalisms required for regression problems, then toward the end of section 2.1.2 extend the notation to deal with classification. Consider an unknown function $\phi : X \rightarrow Y$, which maps from an input space $X \subseteq \mathbb{R}^m$ to an output space $Y \subseteq \mathbb{R}$. Here we assume a single dimensional output space, but all results are easily generalised to the multi-dimensional case [10, 51]. In the problem context we have discussed so far, the input space corresponds to the photograph of the person, the current weather conditions and the demographic information; the output space corresponds to the weight of the person, the amount of snow and the average house price. The overriding principle we work to is that we only have access to a limited, random, data sample of ϕ's behaviour, which we must use to construct a system that can generalise and predict correctly on future inputs. The data sample is the set $t = \{(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), ..., (\mathbf{x}_N, d_N)\}$, where each *input* vector $\mathbf{x}_n \in X$, and each *target* variable $d_n = \phi(\mathbf{x}_n) + \epsilon$ where ϵ is a random scalar value, assumed to be Gaussian distributed with zero mean. This definition for d reflects that our source for this data sample t is not entirely representative of ϕ, it is *noisy*[1]. The set t is drawn randomly according to the distribution of the random variable T, defined over all possible sets, of $(\mathbf{x}, d)$ pairs, of fixed size N.

We have a parameterised function $f(\mathbf{x}; \mathbf{w})$ which we will use as a predictor—our estimated model of ϕ. When we do not wish to refer to any specific input $\mathbf{x}$ or parameter vector $\mathbf{w}$, we will use simply f, or f_i to refer to a specific predictor. For any input $\mathbf{x} \in X$, the predictor will map it to an output in Y. The accuracy of its mapping (i.e. how closely it models ϕ) is determined by two factors: firstly the exact functional form of f, and secondly by the contents of the parameter vector $\mathbf{w} \in \mathbb{R}^k$. We will consider the form of the function in the next section; for the moment imagine it to be any arbitrary form, with the parameter vector $\mathbf{w}$ also set arbitrarily; $f(\cdot; \mathbf{w})$ is now our *predictor*. For a particular $(\mathbf{x}, d)$ pair chosen from t, we can quantify how well our predictor models ϕ with an error measure:

$$(f(\mathbf{x}; \mathbf{w}) - d)^2 \tag{2.1}$$

This is the squared (or *quadratic*) loss of our prediction from the target variable d. Is this a fair assessment of our predictor? No, because there is a random influence at work here that may have masked the true ability of f to model ϕ — in the random choice of *which* input vector $\mathbf{x}$ that we measure the error on, and built into this is what exact amount of random noise is present in the corresponding d. We would like to eliminate these random influences from our error measurements. As such we would ideally like to minimise the following integral:

$$e = \frac{1}{2} \int \int \{(f(\mathbf{x}; \mathbf{w}) - d)^2\} p(d|\mathbf{x}) p(\mathbf{x}).dd.d\mathbf{x} \tag{2.2}$$

[1]We have presumed here that the noise component exists because of a randomness in our measuring device, though it could easily be that ϕ itself is inherently stochastic. In this latter case we would have $d = \phi(\mathbf{x}) = h(\mathbf{x}) + \epsilon$, where $h(\mathbf{x})$ is the deterministic (predictable) part, and ϵ is the non-deterministic (unpredictable) part. In either case, they are mathematically equivalent and we would like to recover the deterministic part, in spite of the noise.

The $\frac{1}{2}$ is a result of assuming the noise component is Gaussian distributed (see [10, p196] for details). This is the weighted average squared error for every possible $\mathbf{x}$ weighted by its probability of occurrence, taking into account every possible noise value d weighted by its probability of occurrence given the $\mathbf{x}$ value. We now define the following conditional averages of the target variable:

$$\langle d|\mathbf{x}\rangle = \int dp(d|\mathbf{x}).dd \tag{2.3}$$

$$\langle d^2|\mathbf{x}\rangle = \int d^2 p(d|\mathbf{x}).dd \tag{2.4}$$

As a shorthand notation when we do not wish to refer to any specific $\mathbf{x}$ we use $\langle d\rangle$. Equation 2.2 can be now simplified (see [10, p202] for details) to give us the following:

$$\begin{aligned} e &= \frac{1}{2}\int \{(f(\mathbf{x};\mathbf{w}) - \langle d|\mathbf{x}\rangle)^2 + (\langle d^2|\mathbf{x}\rangle - \langle d|\mathbf{x}\rangle^2)\}p(\mathbf{x})d\mathbf{x} \\ &= \frac{1}{2}\int \{(f(\mathbf{x};\mathbf{w}) - \langle d|\mathbf{x}\rangle)^2 + \sigma_\epsilon^2\}p(\mathbf{x})d\mathbf{x} \end{aligned} \tag{2.5}$$

where σ_ϵ^2 is the variance of the noise component. Since this variance is independent of our parameter vector $\mathbf{w}$, it is out of our control and therefore is regarded as irreducible. Since the noise is assumed to have zero mean, equation (2.5) is minimised when:

$$f(\mathbf{x};\mathbf{w}) = \langle d|\mathbf{x}\rangle = \phi(\mathbf{x}) \tag{2.6}$$

for all $\mathbf{x}$. The infinite integral in (2.5) can be approximated with a summation over our finite dataset t, where we have presumed that each input is equally likely. The error of predictor $f(\cdot;\mathbf{w})$, on dataset t is defined as:

$$e(f(\cdot;\mathbf{w});t) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}(f(\mathbf{x}_n;\mathbf{w}) - d_n)^2 \tag{2.7}$$

This quantity is known as the *mean squared error* over the set t. With larger N we should be able to get a better approximation to the integral.

We have now established a relatively fair way to assess the abilities of our predictor, eliminating random influences. So far we have assumed that our parameter vector $\mathbf{w}$ has been set arbitrarily, enabling us to issue predictions for any given $\mathbf{x}$, though as we mentioned, how closely the predictor models ϕ depends significantly on this parameter vector. In the following section we will consider the problem of how to determine the optimal parameter vector that minimises error.

2.1.2 The Supervised Learning Problem

Defining the Problem

The *Supervised Learning Problem* is, given a finite *training* set, t, to update $\mathbf{w}$ so as to minimise equation (2.5). It can be seen that this is the challenge of reconstructing the map

ϕ from the set t, in spite of the fact that it is a limited, noisy sample of ϕ's behaviour. We have $\mathbf{w}' \leftarrow \mathcal{L}(f, t, \mathbf{w})$, which indicates that our *learning algorithm* $\mathcal{L}$ has updated the parameter vector $\mathbf{w}$. For the class of predictors we will be considering, it is common for an input to the learning algorithm to be a random initialisation of the parameter vector, drawn according to the distribution of the random variable W, which for this thesis we define as uniform over $[-0.5, 0.5]^k$.

The "trained" function $f(\cdot; \mathbf{w})$ is dependent on two random variables, T and W. Once trained, for any input from the space X, the function will now map[2] it to a point in the output space Y; if it received a different training set or parameter initialization, it may have performed a different mapping. As previously mentioned, when calculating any measure based on our predictor, we would like to eliminate these random influences. For a *particular* input vector $\mathbf{x}$, the *expected value* of the predictor f is defined as:

$$E_{TW}\{f(\mathbf{x};\cdot)\} = \int\int \{f(\mathbf{x};\mathbf{w}')|\mathbf{w}' \leftarrow \mathcal{L}(f,t,\mathbf{w})\} p(t)p(\mathbf{w}).dt.d\mathbf{w} \tag{2.8}$$

This is the weighted average output of the predictor f on the input $\mathbf{x}$, where each output is determined after training f with a *particular* training set t and parameter initalization $\mathbf{w}$, and weighted according to the probability of that training set and that parameter initialization. The two independent random variables T and W form a joint random variable TW. Wherever the expectation operator $E\{\cdot\}$ is used in this thesis, unless explicitly stated, it can be assumed the expectation is calculated with respect to the distribution of TW.

Similarly to (2.8) we define the *expected error* (or *generalization error*) of the predictor on a single input pair $(\mathbf{x}, d)$ as:

$$E_{TW}\{(f(\mathbf{x};\cdot) - d)^2\} = \int\int \{(f(\mathbf{x};\mathbf{w}') - d)^2|\mathbf{w}' \leftarrow \mathcal{L}(f,t,\mathbf{w})\} p(t)p(\mathbf{w}).dt.d\mathbf{w} \tag{2.9}$$

It can be seen that what we are really doing is evaluating the ability of our learning algorithm $\mathcal{L}$ to update the parameter vector toward the optimal vector, given all possible training sets and random parameter initializations. Just as we approximated (2.5) with (2.7), we can approximate the expected error at a single *testing* point $\mathbf{x}$ with an average over several training sets and several random parameter initializations:

$$e(f(\cdot;\cdot)) = \frac{1}{t_{trials}} \sum_t \frac{1}{w_{trials}} \sum_{\mathbf{w}} \left[(f_t(\mathbf{x};\mathbf{w}') - d)^2 | \mathbf{w}' \leftarrow \mathcal{L}(f,t,\mathbf{w}) \right] \tag{2.10}$$

where the number of training sets is t_{trials} and the number of parameter initializations is w_{trials}. This can of course then be averaged over several testing points $(\mathbf{x}, d)$, as in equation (2.7). By designing our learning algorithm such that it can minimise this quantity, we hope to be able to minimise error on future unseen data. Part of the Supervised Learning problem is to update the parameters so the predictor can *generalise* to new data—to use t in such a way as to achieve low error on an unseen *testing* set t'. Practically, a testing set could be obtained by splitting the original set t into two parts, t and t'. If we perform

[2]The mapping it performs is a *hypothesis*: a guess at what the true map ϕ might be. The set of all possible hypotheses is called the *hypothesis space*. The set of all possible configurations of the parameter vector $\mathbf{w}$ is the *parameter space*. It should be noted that more than one point in parameter space can map to a single point in hypothesis space.

well on t but not on t' we are said to have *overfitted* the function—we have fine-tuned our predictor so much that now it only works on the training set t, and is consequently not a good representation of ϕ. This issue will be further discussed in section 2.1.3.

Classification Problems

We mentioned that our output space is continuous, $Y \subseteq \mathbb{R}$, this means the reconstruction of the map ϕ is a *regression* problem. If the output space Y is discrete, $Y = \{c_1, c_2, c_3, ...\}$, where each c_i is a possible 'class' to which each element of X can be mapped, we have a *classification* problem. Classification problems are assessed by a 0-1 (zero-one) loss function; as such, analagously to (2.7), we have:

$$e(f(\cdot;\mathbf{w});t) = \frac{1}{N}\sum_{n=1}^{N} L(f(\mathbf{x}_n;\mathbf{w}), d_n) \tag{2.11}$$

where $L(a,b)$ is 1 if $a \neq b$ and 0 otherwise. This error measure is referred to as the *classification error rate*; it can also be thought of as the percentage of misclassified examples.

Stable and Unstable Predictors

We have discussed how error measures play a role in the Supervised Learning problem and assessing the predictors we work with. We now make a distinction between two classes of predictors, *unstable* and *stable.* An unstable predictor is one that has a strong dependency on its training data, therefore the hypothesis it forms during learning depends to a large degree on what data it received. Examples of unstable predictors are decision trees [34] and neural networks (see section 2.1.4). A stable predictor is one that has no such strong dependency on the training data; examples of stable predictors are k-nearest neighbour classifiers and the Fischer linear discriminant [34]. Unstable classifiers exhibit what is known as *high variance*, while stable classifiers exhibit *low variance* (see [13] for more information). It is this property that brings us to choose unstable classifiers as the focus of the thesis. *Variance* is one half of the well known *bias-variance* decomposition of an error function, which we will now discuss.

2.1.3 Bias and Variance

The most important theoretical tool in machine learning research is the *bias-variance* decomposition [45]. The original decomposition by Geman et al [45] applies to quadratic loss error functions, and states that the generalisation error can be broken into separate components each with their own interpretation. However as mentioned in section 2.1.1, different tasks may require different error measures, not necessarily using quadratic loss. Several authors have proposed decompositions for 0-1 loss [14, 42, 68, 67], each with their own shortcomings and assumptions. Most recently Domingos [33] and James [58] provide equivalent, unified definitions for any symmetric[3] loss function. This thesis is only concerned with the quadratic loss decomposition, though in section 3.1.2 and the Conclusions chapter we do discuss certain possible extensions using the 0-1 loss function.

[3]A symmetric loss function L is one such that $L(a,b) = L(b,a)$

The bias-variance decomposition for quadratic loss states that the generalisation error of an estimator can be broken down into two components: bias and variance. These two usually work in opposition to each other: attempts to reduce the bias component will cause an increase in variance, and vice versa. Techniques in the machine learning literature are often evaluated on how well they can optimize the trade-off between these two components [142, 144]. The bias can be characterised as a measure of how close, *on average over several different training sets*, your estimator is to its target. The variance is a measure of how 'stable' the solution is: given slightly different training data, an estimator with high variance will tend to produce wildly varying performance. Training an estimator for a long time tends to decrease bias, but slowly increase variance; at some point there will be an optimal trade-off that minimises the generalisation error. This is the *bias-variance dilemma*; a typical training session is shown in figure 2.1.

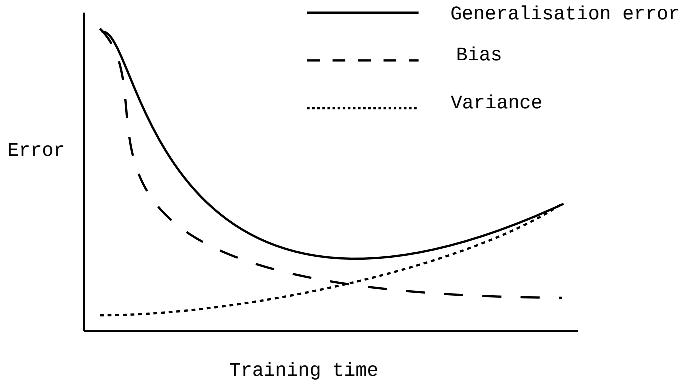

Figure 2.1: Illustration of the trade-off between bias and variance as a function of training time.

To understand these concepts further, consider our unknown function ϕ, a sample training set t, and two estimators, C and S. The C estimator is so named because it is *complicated* — it has a large number of configurable parameters $\mathbf{w}$. The S estimator is so named because it is *simple*, it has relatively few parameters $\mathbf{w}$. Assume now that t is outside the representable function space of S, but within the corresponding space of C. We know that C will be able to fit the function, but also that it will probably *overfit* to the training dataset t, and not be a good estimate of the true function ϕ. For a different random sample of training data, the complex estimator may again overfit, so producing a different hypothesis for each different set of training data presented. This is *high variance*, with regard to the training sets, but because it fits each one so well, it is said to have *low bias*. The simple estimator S will not be able to fit the data t, so will have *high bias*. However, it will respond in almost the same way (*low variance*) for different instances of t, since its representable function space is so small. Formally, the bias-variance decomposition can be proved as follows:

$$\begin{aligned}
E_T\{(f-d)^2\} &= E_T\{(f - E_T\{f\} + E_T\{f\} - d)^2\} \\
&= E_T\{[(f - E_T\{f\}) + (E_T\{f\} - d)]^2\} \\
&= E_T\{(f - E_T\{f\})^2 + (E_T\{f\} - d)^2 + 2(f - E_T\{f\})(E_T\{f\} - d)\} \\
E_T\{(f-d)^2\} &= E_T\{(f - E_T\{f\})^2\} + (E_T\{f\} - d)^2 \\
MSE(f) &= var(f) + bias(f)^2 \qquad (2.12)
\end{aligned}$$

where d is the target value of an arbitrary testing datapoint; the decomposition is therefore a property of the *generalisation* error. It should be noted that for brevity of notation, and without loss of generality, we have presumed a noise level of zero in the testing data[4]. The expectation operator in the decomposition, $E_T\{\cdot\}$, is with respect to all possible training sets of fixed size N, and all possible parameter initialisations. We define this formally as a random sequence T, which is of length N, where each element is an independent observation from an unknown joint distribution $p(\mathbf{x}, d)$. For brevity of notation, we leave the averaging over possible parameter initialisations implicit.

This decomposition is a very widely cited result; a lesser-known result [141] extends this to take account of the possibility that the estimator could be an *ensemble* of estimators. This extension will be discussed in the next chapter. For now, we will consider the exact *type* of learning machine that we will be using throughout this thesis: *neural networks.*

2.1.4 Neural Networks

While we could study combinations of any type of learning machine, the focus of this thesis is on combinations of *multi-layer perceptrons* (MLP), which is a type of *neural network* [10, 51]. An MLP is a directed graph, arranged in layers. Each node in the graph performs a particular function on numeric values that it receives as input from the nodes that connect into it from the previous layer. The node calculates this function, then passes the outcome to each node in the next layer.

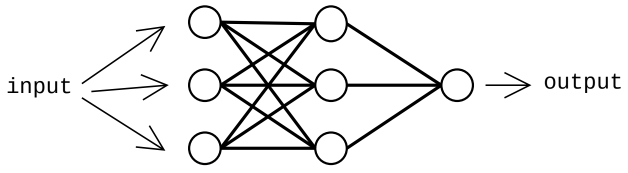

Figure 2.2: A Multi-Layer Perceptron

The layers are termed ‘input’, ‘hidden’ and ‘output’ layers. A layer is ‘hidden’ if it has no *direct* connection to the input or output of the network, so the network above has one hidden layer containing three hidden nodes. The pattern of connections between nodes can vary; in this thesis we only consider *fully-connected feedforward* networks, as in figure 2.2. Each node passes its output forward to other nodes that it is connected to; information flow occurs from left to right in the diagram shown. The nodes in the hidden and output layers calculate their *activation*, which is a weighted summation of the inputs they receive, then pass this through an *activation function.* The activation of node j in a particular layer is:

$$a_j = \sum_i w_{ij} o_i \tag{2.13}$$

where o_i is the output of node i in the previous layer, and w_{ij} is the weighting on the connection from node i to the current node j. The output o_j is produced by the activation

[4]In the case of a non-zero noise component, d in the decomposition would be replaced by its expected value $\langle d \rangle$, and a constant (irreducible) term σ_ϵ^2 would be added, representing the variance of the noise.

function:

$$o_j = \frac{1}{1 + exp(-a_j)} \tag{2.14}$$

This is the *logistic function*, and is used throughout the thesis. The connection pattern and activation functions together define the *network architecture.* In the notation we have established, f is the function implemented by the network architecture, and $\mathbf{w}$ is a vector containing all the weights on connections between the nodes. The weights are real-valued numbers and usually initialised randomly in a small range, for example $[-1, 1]$. For given input and output vectors, there exists a set of weight values which will minimise the mean squared error function[5]. The values of the weights can be discovered by several methods [10, 51]. The most popular method is *gradient descent*, which we will now discuss.

Gradient Descent and the Backpropagation Algorithm

A common algorithm for updating the weights is *backpropagation*, which is used throughout this thesis. Backpropagation [92] is a *gradient descent* technique, meaning weight changes are made in proportion to the gradient of the error function. The error of a network output f, on a single input pattern, is:

$$e_{net} = \frac{1}{2}(f - d)^2 \tag{2.15}$$

Updates to a weight w_{ij} within the network that produces output f are accomplished by:

$$\begin{aligned} w_{ij}^{new} &= w_{ij}^{old} + \Delta_{w_{ij}} \qquad (2.16) \\ \Delta_{w_{ij}} &= -\alpha \frac{\partial e_{net}}{\partial w_{ij}} \end{aligned}$$

where α is a step size or *learning rate.* The updates are performed iteratively, making small updates to each weight in the network until a minimum is reached in the error function. This is *plain* (or *vanilla*) backpropagation. Several modifications can be made to the algorithm to improve performance, such as adding a *momentum* term, which can aid in escape from local minima, or *weight decay* to improve generalisation [51]. In this work we only concern ourselves with plain backpropagation.

[5] In fact many other error functions could be used but MSE is popular in the neural networks literature as it has several nice properties, including the *bias-variance decomposition* as we have shown.

2.2 Methods for Combining a Set of Predictors

The combination strategy for a group of predictors is of fundamental importance, and can decide the performance of the whole system. Sharkey [124] states that it can be thought of as *"managing the recognized limitations of the component estimators"*. In this section we review two types of combination strategy: *linear* and *non-linear*.

2.2.1 Linear Combinations

The simplest possible combination strategy is to take a linearly weighted summation of the individual predictor outputs. The output of the combination is:

$$f_{ens} = \sum_{i=1}^{M} w_i f_i \tag{2.17}$$

where M is the number of predictors, f_i is the output of the ith predictor, and w_i is the corresponding non-negative real-valued combination weight. This is also commonly referred to as the *sum* fusion strategy. Specialisations of the linear combination are the *convex* combination, where weights can be non-uniform but have the constraint that they sum to one: $\sum_{i=1}^{M} w_i = 1.0$; and also the *uniform* combination in eq.(2.18), where all weights are equal at $w_i = \frac{1}{M}$.

$$\bar{f} = \frac{1}{M} \sum_{i=1}^{M} f_i \tag{2.18}$$

A set of predictors combined by a uniform weighting is referred to as the *Simple ensemble*. But which of these strategies should we use—uniform weighting or non-uniform weighting?

Regression Problems

The first[6] major study on combining regression estimators was by Perrone [109]. Given a target pattern $(\mathbf{x}, d)$, define the *misfit* of a predictor $f_i(\mathbf{x})$ as $m_i(\mathbf{x}) = d - f_i(\mathbf{x})$. Define the correlation matrix $\mathbf{C}$ with elements indexed as $C_{ij} = E\{m_i(\mathbf{x})m_j(\mathbf{x})\}$. Perrone [109], and independently Hashem [50], showed equivalent solutions on how to determine optimal weights. It can be shown that the optimal weights are:

$$w_i = \frac{\sum_{j=1}^{M} (\mathbf{C}^{-1})_{ij}}{\sum_{k=1}^{M} \sum_{j=1}^{M} (\mathbf{C}^{-1})_{kj}} \tag{2.19}$$

In order to calculate this, we have to estimate the correlation matrix $\mathbf{C}$. This matrix, and therefore the optimal weights, *cannot computed analytically* but can be *approximated* with a validation dataset [50],[10, p368], as follows:

$$C_{ij} \approx \frac{1}{N} \sum_{n=1}^{N} \Big[(d_n - f_i(\mathbf{x}_n))(d_n - f_j(\mathbf{x}_n)) \Big] \tag{2.20}$$

We calculate this value on a dataset of size N, in (2.20), determine optimal weights with (2.19), then calculate the ensemble output with (2.17).

[6]This was the first study in the machine learning literature, but the topic has been covered in other research communities for several years, e.g. in financial forecasting, Clemen [23], and Bates and Granger [6].

Classification Problems

Tumer and Ghosh [140] (see [138] or [139] for fuller explanations) provided a theoretical framework for analysing the simple averaging combination rule when our predictor outputs are estimates of the posterior probabilities of each class. First, we note that the total error of a predictor is equal to the Bayes error plus some 'added' amount of error [138]:

$$E_{tot} = E_{add} + E_{bay} \tag{2.21}$$

Now, given a one-dimensional feature vector x, the ith individual predictor approximates the posterior probability of class a as:

$$\hat{P}_i(a|\mathbf{x}) = P(a|\mathbf{x}) + \eta_i(a|\mathbf{x}) \tag{2.22}$$

$P(a|\mathbf{x})$ is the true posterior probability of class a and $\eta_i(a|\mathbf{x})$ is the estimation error. Let us assume the estimation errors on different classes a and b are independent and identically distributed random variables [139, p5] with zero mean and variance $\sigma^2_{\eta_i}$. Now the expected added error of the ith predictor in distinguishing between the two classes a and b is

$$E_{add,i} = \frac{2\sigma^2_{\eta_i}}{P'(a|\mathbf{x}) - P'(b|\mathbf{x})} \tag{2.23}$$

where $P'(a|\mathbf{x})$ is the derivative of the true posterior probability of class a. This framework and its consequences will be discussed more in chapter 3. It is interesting here because Roli and Fumera [43, 114] recently extended it to allow non-uniform weighting. They show that in the case of the classifiers having uncorrelated estimation errors η_i and η_j, then the optimal weights for the convex combination are:

$$w_i = \frac{1}{E_{add,i}}\Big[\sum_{j=1}^{M} \frac{1}{E_{add,j}}\Big]^{-1} \tag{2.24}$$

For classifiers with *correlated* estimation errors (the most likely situation) they suggest that it is a very difficult problem to compute optimal weights analytically, and to date no solution has been shown in the literature.

Classification problems can be viewed as a special case of regression problems, where the targets are always discrete values—for a two-class problem, targets are 1 and 0. Jimenez [60] used this principle in his *"Dynamically Averaged Networks"* (DAN), and compared approximating optimal weights by cross-validation [10, 50, 108] as we have described using equation (2.20), against his own method using the 'certainty' of an estimator in its answer. This used a neural network as the predictor; a single sigmoidal output unit was thresholded at 0.5 to distinguish between class 0 and class 1. The certainty of a network f_i on an input x was:

$$cert(f_i(\mathbf{x})) = \begin{cases} f_i(\mathbf{x}) & \text{if } f_i(\mathbf{x}) > \frac{1}{2}; \\ 1 - f_i(\mathbf{x}) & otherwise. \end{cases} \tag{2.25}$$

and the convex combination weights were given by:

$$w_i = \frac{cert(f_i(\mathbf{x}))}{\sum_{j=1}^{M} cert(f_j(\mathbf{x}))} \tag{2.26}$$

This DAN method was tested on four real world datasets, and found to outperform or equal the performance of using either uniform weights or optimal weights estimated by the validation set procedure.

Generally, we would expect to weight the classifiers inversely proportional to how well we expect them to perform; a very good classifier gets a high weight, while a not so good classifier gets a small weight. With this in mind, some authors have developed their own heuristics for weighting the ensemble members. For binary classification problems, Freund et al [39] weight predictors exponentially with respect to their training error. Tresp and Taniguchi [135] investigate non-constant weighting functions, where the weightings are determined anew for each testing pattern, by calculating a probability for how likely a given estimator is to have seen data in a region of the input space close to a new input pattern.

2.2.2 Non-Linear Combinations

When we have a classification problem, and our learner outputs a discrete class label rather than a real-valued number, a widely used combination rule is a *majority vote* among the labels predicted by each member of the ensemble [7, 115]. Kittler and Alkoot [65] theoretically analysed the relationship between the sum and vote fusion strategies. They found that when the errors in our estimate of the posterior probability of a class have a normal distribution, sum always outperforms vote, whereas for heavy tail distributions, vote may outperform sum; this was confirmed with empirical data. Kuncheva et al [72] derive theoretical upper bounds on the maximum achievable accuracy when using a majority voting combination.

Bahler and Navarro [5] conduct a large empirical study into using different combination rules. They found that when accuracy is approximately balanced across the estimators, majority voting performs as well as any more complex combination rule. When accuracy is imbalanced, majority vote tends to decrease in reliability, while more complex methods which take account of the individual performances, like Dempster-Schafer theory of evidence [91] and Bayesian methods [2], retain their performance.

Tumer and Ghosh [136] introduce *order statistics* as combination rules for classification tasks. For this we take M estimators, ordered such that the outputs for a particular class have the property $\{f_1 \leq f_2 ... \leq f_M\}$. The *min, max* and *median* rules are:

$$f_{min}(\mathbf{x}) = f_1(\mathbf{x}) \tag{2.27}$$

$$f_{max}(\mathbf{x}) = f_M(\mathbf{x}) \tag{2.28}$$

$$f_{med}(\mathbf{x}) = \begin{cases} f_{\frac{M}{2}}(\mathbf{x}) & \text{if} \quad M \text{ is even} \\ f_{\frac{M+1}{2}}(\mathbf{x}) & \text{if} \quad M \text{ is odd} \end{cases} \tag{2.29}$$

For example, consider the situation if we had an ensemble of four neural networks on a two-class problem. Each has a single sigmoidal output unit, if the output is above 0.5, we say it is class A, and class B otherwise. For a new pattern the networks give outputs $0.2, 0.6, 0.7$ and 0.99. The *min* rule would predict class B, whereas the *max* and *med* rules would predict class A. Roli and Fumera [114] presents empirical evidence supporting the arguments by Tumer and Ghosh [137] stating that order statistics are most beneficial as combination rules when the classifiers are highly imbalanced in accuracy.

An alternative to the static combination methods discussed so far is to choose a single predictor from the ensemble, dependent on what input pattern is received. The DCS-LA

(Dynamic Classifier Selection with Local Accuracy) algorithm by Woods et al [150] uses estimates of local accuracy in the input space to select a single classifier to respond to a new pattern; their experiments plus extensive testing by Kuncheva [71] establish DCS-LA as a very robust technique for combining estimators that have different individual accuracies.

A novel approach was adopted by Langdon [81, 82, 83] using a Genetic Programming [69] system, to find a highly non-linear method of combining the outputs of the ensemble. This combination rule was applied to a medical dataset and Receiver-Operator Characteristics (ROC) [158] were calculated. The system was found to improve performance in ROC space *beyond* a previously suggested theoretical upper bound [122]. A similarly inspired evolutionary approach was taken by Zhou et al [159], using a Genetic Algorithm to select a good set of networks to include in the ensemble.

Other techniques that are commonly used include Bayesian formulations [134, 24], Dempster-Schafer Theory of Evidence [91], and the Behaviour Knowledge Space method [112]. Suen and Lam [131] provide an excellent review of a number of different rules. Several authors have compared the effectiveness of these and other combination rules in different situations; for example, Duin and Tax [35], Bauer and Kohavi [8], and Ali [2]. A point of agreement seems to be that no one combination rule is overall better than another, and that no rule at all can be a substitute for a well-trained set of predictors in the first place. So how do you learn a good set of predictors?

2.3 Architectures for Learning a Set of Predictors

In this section we review the two main architectures for learning a set of predictors, *ensembles* and *Mixtures of Experts*. These two are very often seen as opposites, one "modular" and one "non-modular". The supposed boundary here can easily be blurred as we illustrate a third technique which is claimed to blend smoothly between the two; we discuss this claim and the relationship between all three systems.

2.3.1 Ensembles

The characteristic feature of an ensemble is *redundancy*; each predictor could perform the task on its own, but better generalization performance is obtained from the combination. The idea of an ensemble of neural networks can be traced back to Nilsson [99] who arranged a group of perceptrons in a layer, and combined their outputs with a second, vote-taking, layer. In this section we will review *Bagging* and *Boosting*, two of the most widely used techniques for creating an ensemble.

In Bagging [13] (short for *B*ootstrap *Agg*regation Learn*ing*) we randomly generate a new training set, based on the original set, for each ensemble member. Given a training set of size N, from this we uniformly at random sample N items *with replacement*, then train a new ensemble member with this resample. We repeat this for any new ensemble members we wish to create. The resampled sets are often termed *bootstrap replicates* [36]; Breiman showed that on average 63.2% of the original training set will present in each replicate. Figure 2.3 shows the algorithm.

Bagging has proved to be a popular technique, applicable to many problems, but the explanation for its success remains controversial. Friedman [42] suggests that Bagging succeeds by reducing the variance component of the error and leaving the bias unchanged;

```
1. Let $M$ be the final number of predictors required.
2. Take a training set $t = \{(\mathbf{x}_1, d_1), ..., (\mathbf{x}_N, d_N)\}$.
3. For $i = 1$ to $M$ do :
    • Make a new training set $t_{bag}$ by sampling $N$ items uniformly at random with
      replacement from $t$.
    • Train an estimator $f_i$ with this set $t_{bag}$ and add it to the ensemble.

For any new testing pattern $\mathbf{x}$, the Bagged ensemble output is given by:

    $f_{bag}(\mathbf{x}) = \frac{1}{M} \sum_i f_i(\mathbf{x})$
```

Figure 2.3: The Bagging Algorithm

while [46] shows evidence that Bagging can in fact converge *without* reducing variance. Domingos [32] gives a Bayesian account, and investigates two hypotheses: Bagging succeeds because (1) it is an approximation to Bayesian Model Averaging [26], and (2) it shifts the prior distribution of the combined estimator to a more appropriate region of the model space; the empirical results on 26 real and artificial datasets all support the second hypothesis and contradict the first.

Bagging resamples the dataset randomly with a uniform probability distribution; an alternative would be to have a non-uniform distribution, as we have in *Boosting* [119]. Since the initial work by Schapire [119], a large family of Boosting algorithms have developed, including cost-sensitive versions [37, 63] and those which can provide confidence estimates in their predictions [121]. AdaBoost [40] is probably the most widely investigated variant. The algorithm proceeds in rounds, with a new network being trained at each round. A network is trained initially with equal emphasis on all training patterns. At the end of each round, mis-classified patterns are identified and have their emphasis increased in a new training set which is then fed back into the start of the next round, and a new network is trained. After the desired number of networks have been trained, they are combined by a weighted vote, based on their training error. Figure 2.4 shows the algorithm; the version shown is one of several ways *AdaBoost*ing could be conducted. Kuncheva [78, 77] provides in-depth studies of how subtle changes to the probability distribution updates can lead to very different behaviours.

1. Let M be the final number of predictors required.
2. Take a training set $t = \{(x_1, d_1), ..., (\mathbf{x}_n, d_N)\}$, where $\forall n, d_n \in \{-1, +1\}$. Define a distribution D_i, initially uniform over elements of t, so $\forall n, D_i(n) = \frac{1}{N}$
3. For $i = 1$ to M do :
   - Make a new training set t_{boost} by sampling N items at random *with replacement* from t, according to the distribution D_i.
   - Train an estimator f_i with this set t_{boost} and add it to the ensemble.
   - Calculate $\eta_i = \sum_{(\mathbf{x}_n, d_n) \in t} I(f_i(\mathbf{x}_n), d_n)$, where $I(\cdot, \cdot)$ returns $D_i(n)$ if the pattern is classified *incorrectly* and 0 otherwise.
   - Set $\alpha_i = \frac{1}{2} ln\left(\frac{\eta_i}{1-\eta_i}\right)$
   - For each pattern $(\mathbf{x}_n, d_n) \in t$, update $D_{i+1}(n) = \frac{D_i(n) exp(-\alpha_i d_n f_i(\mathbf{x}_n))}{Z_i}$ where Z_i is a normalization factor to ensure $D_{i+1}(n)$ is a distribution.

For any new testing pattern $\mathbf{x}$, the Boosted ensemble output is given by:

$$f_{boost}(\mathbf{x}) = sign(\sum_{i=1}^{M} \alpha_i f_i(\mathbf{x}))$$

Figure 2.4: The AdaBoost Algorithm

The combination in AdaBoost is a linear (not necessarily convex) combination of the network outputs, with weights α based on the training error but are *fixed* with respect to the input patterns. The obvious extension to this is to allow the weights to vary according to the input pattern, so the α values are re-calculated by some method for every new x value. Several authors, including Schapire and Singer [118], Meir et al [96] and Moerland [98], do precisely this with versions of "localized" boosting. Their architectures bear a number of similarities to the *Mixture of Experts*, which will be covered in the next section. Although we have so far discussed Boosting in the context of classification problems, Avnimelech and Intrator [3] showed an extension of AdaBoost to boosting with regression estimators. Schapire [120] conducts a review of recent theoretical analysis on AdaBoost, describing links to game theory, and extensions to handle multi-class problems.

Maclin and Opitz [90] and also Bauer [8] compare Bagging and Boosting methods in a large empirical study. Their findings show that although Bagging almost always produces an ensemble which is better than any of its component classifiers, and is relatively impervious to noise, it is on average not significantly better than a simple ensemble. They find Boosting to be a powerful technique, usually beating Bagging, but is susceptible to noise in the data and can quickly overfit; similar problems with overfitting in AdaBoost have been observed by a number of authors. Most recently, Jin et al [61] use a confidence-based regularisation term when combining the Boosted learners; if learners early on in the Boosting chain are confident in their predictions, then the contribution of learners later on in the chain is down-played; this technique shows significantly improved tolerance to noisy datasets.

2.3.2 Mixtures of Experts

The Mixtures of Experts architecture is a widely investigated paradigm for creating a combination of estimators [54, 55]. The principle underlying the architecture is that certain estimators will be able to 'specialise' to particular parts of the input space. A *Gating network* receives the same inputs as the component estimators, but its outputs are used as the combination weights. The Gating network is responsible for learning the appropriate weighted combination of the specialised estimators (experts) for any given input. In this way the input space is 'carved-up' between the experts, increasing and decreasing their weights for particular patterns. Figure 2.5 illustrates the basic architecture.

For M experts, the gating network consists of a single layer of M neurons, which each neuron assigned to a specific expert. The activation of the ith neuron on a single pattern $\mathbf{x}$ is:

$$c_i = \sum_j x_j w_{ij} \tag{2.30}$$

where the sum is over all of the input features $x_j \in \mathbf{x}$, and w_{ij} is the weight on the connection from input feature j to this neuron i. The output of the gating network is normalised so that the gates on the experts sum to one. The *softmax* activation function is used, thus the output of the *ith* gate is:

$$g_i = \frac{exp(c_i)}{\sum_{j=1}^{M} exp(c_j)} \tag{2.31}$$

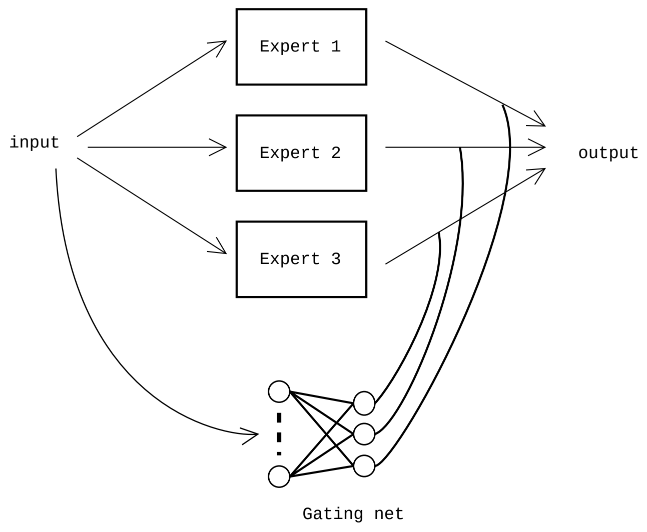

Figure 2.5: The Mixtures of Experts architecture

The output of the whole system is:

$$f_{mix} = \sum_{j=1}^{M} g_j f_j \tag{2.32}$$

For a single pattern $(\mathbf{x}, d)$, the error function to be minimised for the system is:

$$e_{mix} = \frac{1}{2}(f_{mix} - d)^2 \tag{2.33}$$

If using gradient descent to train the system, the weight updates for the experts will be calculated using the partial derivative:

$$\frac{\partial e_{mix}}{\partial f_i} = g_i(\sum_{j=1}^{M} g_j f_j - d) \tag{2.34}$$

and the updates to the gating network weights with:

$$\frac{\partial e_{mix}}{\partial g_i} = f_i(\sum_{j=1}^{M} g_j f_j - d) \tag{2.35}$$

A huge number of variants of this paradigm have developed since the original concept paper [54]. In a paper published later in the same year [57] the ideas were extended and formalised. In [62] the ideas were given a theoretical grounding in associative Gaussian mixture models, and the Expectation-Maximisation (EM) algorithm was shown to be useful for learning the weights of the networks.

2.3.3 Dyn-Co : Blending between Mixtures and Ensembles?

Hansen's Dyn-Co algorithm [48] uses the same overall architecture as the original Mixtures of Experts method [54], but using a different error function. The error for the ith expert is:

$$e_{dynco,i} = \frac{1}{2}(\sum_{j=1}^{M} g_j f_j - d)^2 + \gamma\frac{1}{2}(g_i - \frac{1}{M})^2 \tag{2.36}$$

The update for the experts remains the same, using a partial derivative identical to equation 2.34, but the update for the gating network now contains an regularisation element which constrains the values of the gates. The update to the gating network weights uses the partial derivative:

$$\frac{\partial e_{dynco,i}}{\partial g_i} = f_i(\sum_{j=1}^{M} g_j f_j - d) + \gamma g_i(g_i - \sum_{j=1}^{M} g_i^2) \tag{2.37}$$

When the scaling parameter γ is zero, Dyn-Co is equivalent to a standard Mixtures of Experts model. When γ approaches infinity, the outputs of the gating net are forced to be equal, at $\frac{1}{M}$. Hansen's claim [48, p86] was that when $\gamma = 0$ we have a 'pure' Mixtures method, and when $\gamma \rightarrow \infty$, Dyn-Co is a 'pure' ensemble method.

If all the gates $g_i = \frac{1}{M}$, then the error gradient for the experts becomes:

$$\frac{\partial e_{dynco,i}}{\partial f_i} = \frac{1}{M}(\bar{f} - d) \tag{2.38}$$

However, the conventional definition for an ensemble training method is to train ensemble members *independently* of one another. The error function for a single network f_i is:

$$e_{net} = \frac{1}{2}(f_i - d)^2 \tag{2.39}$$

and the error gradient:

$$\frac{\partial e_{net}}{\partial f_i} = (f_i - d) \tag{2.40}$$

As $\gamma \rightarrow \infty$, the error gradient approaches equation 2.38. Since equation 2.40 is not equal to equation 2.38, we can see that for Dyn-Co with large γ, the training procedure is not exactly equivalent to the more widely accepted definition of an ensemble.

2.3.4 Ensembles, Mixtures, and Dyn-Co

In introductory texts that mention combinations of predictors, it is common to draw a bold distinction between "modular" and "ensemble" systems [51]. This can be regarded as very misleading, many authors have also commented on the similarities between the two [38, 124], and as we have seen Dyn-Co seems to provide a link of sorts. In the Conclusions chapter, we will demonstrate that the technique we study as the focus of this thesis, Negative Correlation learning [85] provides the missing link that unites these. Though it is not entirely clear how the dynamics interact, we will show evidence that a smooth space of algorithms exists between these three paradigms.

2.4 Chapter Summary

We have now reviewed the main issues encountered when combining predictors in groups. We first introduced the *Supervised Learning* problem and described the *multilayer perceptron*, the predictor to be used in this thesis. We then reviewed some of the methods for combining a given set of predictors, followed by a summary of architectures for learning the predictors that are to be combined. Among these architectures is the paradigm of *ensemble learning*, where each predictor attempts to solve the same problem and errors can potentially be reduced by *averaging* their predictions. Many of these techniques will be described in more detail in a taxonomy we propose in the next chapter.

We introduced an important theoretical result for supervised learning, the *bias-variance* decomposition of the generalisation error. This decomposition is extended in the following chapter, we show how it becomes an extremely useful result for the ensemble learning literature, and is fundamental to some of the later results in this thesis.

This chapter has served to identify the wider body of literature to which the thesis contributes. In the following chapter we will focus the discussion a little more and review the concept of ensemble error "diversity".

Chapter 3

Defining Diversity

In chapter 1 we introduced the idea that an ensemble of predictors can be improved if the ensemble members exhibit 'diverse' errors. In this chapter we further analyse the concept of error "diversity". We review existing explanations of why ensembles with diverse errors perform well, for both regression and classification contexts. In the process of this we clarify a subtle point, often overlooked, to do with quantifying classifier ensemble diversity and the inherent non-ordinality of the predictor outputs. We proceed with a review of the literature on attempts to create diversity in both forms, commenting on the structure of the field and proposing a novel way to categorise different diversity creation techniques.

3.1 Why do Ensembles perform better than Single Networks?

3.1.1 In a Regression Context

First, as an illustrative scenario, consider a single neural network approximating a sine wave; our network has been supplied with a limited set of datapoints to train on, the inputs chosen randomly at uniform from $[-\pi, \pi]$, and a small amount of Gaussian noise added to the outputs. Now, consider a single testing datapoint, to find the value of $sin(2)$. The true answer is ~ 0.909, yet we know our network may possibly overpredict or underpredict that value. The way in which it makes errors will follow a distribution dependent on the random training data sample it received, and also on the random initialisation of the weights. The mean of this distribution is the expectation value $E_{TW}\{f\}$, and f is a network trained with a particular dataset and a particular weight initialisation. Figure 3.1 illustrates a typical error distribution, with the target value d shown. The four crosses marked are estimates of $sin(2)$ from a hypothetical network; the estimates differ only in that the network was started from different initial weights each time.

From this viewpoint we can immediately see the parallels to an ensemble system. Each member of an ensemble is a realisation of the random variable defined by this distribution over all possible training datasets and weight initialisations. The ensemble output is the average of this set of realisations; all our diversity promoting mechanisms are there to encourage our sample mean $\bar{f}$ to be a closer approximation to $E_{TW}\{f\}$. If we have a large ensemble, we have a large sample from this space; consequently with a large sample we can expect that we have a good approximation to the mean, $E_{TW}\{f\}$. If we have a smaller ensemble, we cannot expect this: our sample mean may be upward or downward

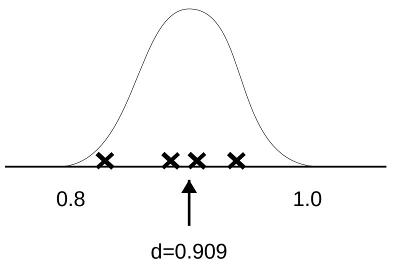

Figure 3.1: Typical error distribution of an unbiased estimator approximating $sin(2)$

biased. In order to correct this, some methods, such as Bagging (see section 2.3.1), construct our networks from different training datasets, allowing us to sample a more representative portion of the space. This illustration assumes that the expected value of our estimator is equal to the true target value, i.e. an *unbiased* estimator. If this is not the case, we may have the situation in figure 3.2.

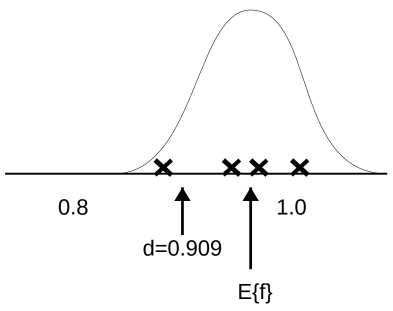

Figure 3.2: Typical error distribution of a biased estimator approximating $sin(2)$

Here, our estimator is upward biased, i.e. $E\{f\} \neq d$, its expected value $E\{f\}$ is high of the target d. In this case, even if we sample many times from this distribution, we will not be able to estimate the target accurately with a simple average combination as the simple average will converge to $E\{f\}$ as we add more networks. We would need to non-uniformly weight the outputs, giving higher weights to the networks predicting lower values. This is of course a purely hypothetical scenario, we could not look this closely at every single datapoint to manually set the weights for the combination, but it does serve to illustrate that the chance of an error could be reduced by using a combination of several predictors rather than one. This intuition can be more formalised as the *Ambiguity Decomposition.*

The Ambiguity Decomposition

Krogh and Vedelsby [70] proved that *at a single datapoint the quadratic error of the ensemble estimator is guaranteed to be less than or equal to the average quadratic error of the component estimators*:

$$(f_{ens} - d)^2 = \sum_i w_i(f_i - d)^2 - \sum_i w_i(f_i - f_{ens})^2 \tag{3.1}$$

where f_{ens} is a convex combination of the component estimators:

$$f_{ens} = \sum_i w_i f_i \tag{3.2}$$

This result turns out to be very important for the focus of this thesis, so we present a full proof. The details of the original proof from [70] were omitted for the authors' space considerations; we present the full details of their proof in appendix A. However, this can in fact be shown more simply by the same manipulations as used in the bias-variance decomposition (see section 2.1.3), reflecting a strong relationship between the two decompositions. We present this alternative version here:

$$\begin{aligned}
\sum_i w_i(f_i - d)^2 &= \sum_i w_i(f_i - f_{ens} + f_{ens} - d)^2 \\
&= \sum_i w_i[(f_i - f_{ens})^2 + (f_{ens} - d)^2 + 2(f_i - f_{ens})(f_{ens} - d)] \\
&= \sum_i w_i(f_i - f_{ens})^2 + (f_{ens} - d)^2 \\
(f_{ens} - d)^2 &= \sum_i w_i(f_i - d)^2 - \sum_i w_i(f_i - f_{ens})^2 \qquad (3.3)
\end{aligned}$$

This was a very encouraging result for ensemble research, providing a very simple expression for the effect due to error correlation in an ensemble. In section 3.2.4 we will see how a number of researchers have exploited this result in various ways for learning in ensembles. The significance of the Ambiguity decomposition is that it shows us that if we have any given set of predictors, the error of the convex-combined ensemble will be less than or equal to the average error of the individuals. Of course, one of the individuals may in fact have lower error than the average, and lower than even the ensemble, on a particular pattern. But, given that we have no criterion for identifying that best individual, all we could do is pick one at random. One way of looking at the significance of the Ambiguity decomposition is that it tells us that taking the combination of several predictors would be better *on average over several patterns*, than a method which selected one of the predictors at random.

The decomposition is made up of two terms. The first, $\sum_i w_i(f_i - d)^2$, is the weighted average error of the individuals. The second, $\sum_i w_i(f_i - f_{ens})^2$ is the *Ambiguity term*, measuring the amount of variability among the ensemble member answers for this pattern. Since this is always positive, it is subtractive from the first term, meaning the ensemble is guaranteed lower error than the average individual error. The larger the Ambiguity term, the larger the ensemble error reduction. However, as the variability of the individuals rises, so does the value of the first term. This therefore shows that diversity itself is not enough, we need to get the right balance between diversity (the Ambiguity term) and individual accuracy (the average error term), in order to achieve lowest overall ensemble error.

The Ambiguity decomposition holds for convex combinations, and is a property of an ensemble trained on a single dataset. Unlike the bias-variance decomposition, it does not take into account the distribution over possible training sets or possible weight initialisations. What we are interested in, of course, is the expected error on future datapoints given these distributions. The Bias-Variance-Covariance decomposition [141] takes exactly this into account.

Bias, Variance and Covariance

The concept of ensemble diversity can be understood further if we re-examine the bias-variance decomposition for an ensemble. If the ensemble is a uniformly weighted convex combination, the variance term breaks down further, and we have the *Bias-Variance-Covariance decomposition* [141]. However, from this point forward, it should be noted that the expectations are subtly different to that in the original bias-variance decomposition. The ith estimator is trained using a training set drawn from the random sequence T_i. Each $T_i, i = \{1..M\}$ has the same distribution, but they are not necessarily mutually independent[1]. The expectations used below are therefore either over a particular T_i, over a subset of them, or over all of them, $T_1, ..., T_M$. We denote this as:

$$\begin{aligned} E_{T_i}\{\cdot\} &= E_i\{\cdot\} \\ E_{T_i,T_j}\{\cdot\} &= E_{i,j}\{\cdot\} \\ E_{T_1,...,T_M}\{\cdot\} &= E\{\cdot\} \end{aligned}$$

It should be noted that although the decomposition presented below does hold for non-uniformly weighted ensembles, we restrict our analysis to the uniform case, as it corresponds to the simple average ensemble combination technique used commonly in practice. To aid our exposition now, we define three concepts. The first concept is $\overline{bias}$, the averaged bias of the ensemble members:

$$\overline{bias} = \frac{1}{M}\sum_i (E_i\{f_i\} - d) \tag{3.4}$$

The second is $\overline{var}$, the averaged variance of the ensemble members:

$$\overline{var} = \frac{1}{M}\sum_i E_i\{(f_i - E_i\{f_i\})^2\} \tag{3.5}$$

The third is $\overline{covar}$, the averaged covariance of the ensemble members:

$$\overline{covar} = \frac{1}{M(M-1)}\sum_i \sum_{j \neq i} E_{i,j}\{(f_i - E_i\{f_i\})(f_j - E_j\{f_j\})\} \tag{3.6}$$

We have the original bias-variance decomposition from equation (2.12):

$$E_T\{(f-d)^2\} = (E_T\{f\} - d)^2 + E_T\{(f - E_T\{f\})^2\}$$

if the f here is a *uniformly weighted combination* of other components, then the variance component can be broken down further:

$$\begin{aligned} Var(\frac{1}{M}\sum_i f_i) &= E\{(\frac{1}{M}\sum_i f_i - E\{\frac{1}{M}\sum_i f_i\})^2\} \\ &= E\{(\frac{1}{M}\sum_i f_i - E\{\frac{1}{M}\sum_i f_i\})(\frac{1}{M}\sum_i f_i - E\{\frac{1}{M}\sum_i f_i\})\} \end{aligned}$$

[1]See [141] for details of the independent case.

$$
\begin{aligned}
&= E\{\frac{1}{M}\sum_i (f_i - E_i\{f_i\})\frac{1}{M}\sum_j (f_j - E_j\{f_j\})\} \\
&= E\{\frac{1}{M^2}\sum_i\sum_j (f_i - E_i\{f_i\})(f_j - E_j\{f_j\})\} \\
&= E\{\frac{1}{M^2}\sum_i\sum_{j\neq i} (f_i - E_i\{f_i\})(f_j - E_j\{f_j\})\} + E\{\frac{1}{M^2}\sum_{j=i}(f_j - E_j\{f_j\})^2\} \\
&= (1-\frac{1}{M})\frac{1}{M(M-1)}\sum_i\sum_{j\neq i} E_{i,j}\{(f_i - E_i\{f_i\})(f_j - E_j\{f_j\})\} \\
&\quad +\frac{1}{M}\frac{1}{M}\sum_{j=i} E_j\{(f_j - E_j\{f_j\})^2\} \\
&= (1-\frac{1}{M})\overline{covar} + \frac{1}{M}\overline{var}
\end{aligned}
$$

Similarly, the bias component can be broken down:

$$
\begin{aligned}
bias(\frac{1}{M}\sum_i f_i)^2 &= \Big(E\{\frac{1}{M}\sum_i f_i\} - d\Big)^2 \\
&= \Big(\frac{1}{M}\sum_i (E_i\{f_i\} - d)\Big)^2
\end{aligned}
$$

so we have the mean squared error of the ensemble estimator $\bar{f}$ as:

$$
E\{(\bar{f}-d)^2\} = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1-\frac{1}{M}\right)\overline{covar} \tag{3.7}
$$

We can see that the error of an *ensemble* of estimators depends *critically* on the amount of error correlation *between* them, quantified in the covariance term. We would ideally like to decrease the covariance, without causing any increases in the $\overline{bias}^2$ or $\overline{var}$ terms.

The Connection Between Ambiguity and Covariance

We now show the exact link between the two decompositions we have just described. Assuming a uniform weighting, we substitute the right hand side of equation (3.1) into the left hand side of equation (3.7):

$$
E\{\frac{1}{M}\sum_i (f_i - d)^2 - \frac{1}{M}\sum_i (f_i - \bar{f})^2\} = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1-\frac{1}{M}\right)\overline{covar} \tag{3.8}
$$

It would be interesting to understand what portions of the bias-variance-covariance decomposition correspond to the ambiguity term. After some manipulations (proof in appendix B) we can show:

$$
E\{\frac{1}{M}\sum_i (f_i - \bar{f})^2\} = \frac{1}{M}\sum_i E\{(f_i - E\{\bar{f}\})^2\} - E\{(\bar{f} - E\{\bar{f}\})^2\}
$$

$$
\begin{aligned}
&= \frac{1}{M}\sum_i \left[E_i\{(f_i - E_i\{f_i\})^2\} + (E_i\{f_i\} - E\{\bar{f}\})^2\right] - E\{(\bar{f} - E\{\bar{f}\})^2\} \\
&= \overline{var} + \text{“}deviations\text{”} - var(\bar{f}) \\
&= \overline{var} + \text{“}deviations\text{”} - \frac{1}{M}\overline{var} - \left(1 - \frac{1}{M}\right)\overline{covar} \qquad (3.9)
\end{aligned}
$$

This is the average variance of the networks, plus a term measuring the average deviations of the individual expectations from the ensemble expectation, minus the variance of the ensemble output. The Ambiguity term makes up only one part of the ensemble MSE, the other part is the average MSE of the individual members. The expected value of this part is equal to:

$$
\begin{aligned}
E\{\frac{1}{M}\sum_i (f_i - d)^2\} &= \left(\frac{1}{M}\sum_i (E_i\{f_i\} - d)\right)^2 + \frac{1}{M}\sum_i E\{(f_i - E\{\bar{f}\})^2\} \\
&= \left(\frac{1}{M}\sum_i (E_i\{f_i\} - d)\right)^2 + E\{(\bar{f} - E\{\bar{f}\})^2\} + \frac{1}{M}\sum_i (E_i\{f_i\} - E\{\bar{f}\})^2 \\
&= \overline{bias}^2 + \overline{var} + \text{“}deviations\text{”} \qquad (3.10)
\end{aligned}
$$

Again here we see the interaction terms—the average variance and the ‘deviations’ term. The expected average MSE and the expected Ambiguity contain these as common terms. These terms constitute an interaction between the two sides. However, when we combine them by subtracting the Ambiguity in equation (3.9), from the average MSE in equation (3.10), the interaction terms cancel out, and we get the original bias-variance-covariance decomposition back, as in the RHS of equation (3.8). The fact that the interaction exists illustrates why we cannot simply maximise Ambiguity without affecting the bias component.

In this Section we have clearly identified how “diversity” is quantified for simple-averaging ensembles, in terms of two separate but related error decompositions. It should be noted that these two decompositions have told us merely how to *quantify* diversity, not how to *achieve it* in this type of ensemble. In addition we have said nothing yet about ensemble schemes involving other types of combination, such as voting or other non-linear schemes; we will address this in the next section.

3.1.2 In a Classification Context

We know understand that in a regression context, we can rigorously define why and how differences between individual predictor outputs contribute toward overall ensemble accuracy. In a classification context, there is no such neat theory. There is a subtle point here, often overlooked. Difficulties in quantifying classification error diversity *is not intrinsic* to ensembles tackling classification problems. It is possible to reformulate any classification problem as a regression one by choosing to approximate the class posterior probabilities; this allows the theory we have already discussed to apply, and work is progressing in this area, notably[2] Tumer and Ghosh [140] and Roli and Fumera [44, 114]. For the regression context discussed in the previous section, the question can be clearly phrased as *“how can*

[2]It is interesting to note that some of the calculations in Tumer and Ghosh’s work rely fundamentally on the bias-variance-covariance decomposition.

we quantify diversity when our predictors output real-valued numbers and are combined by a convex combination?". For the case that Tumer and Ghosh [140] study, the question is the same, just that the "real-valued" numbers are probabilities. A much harder question appears when we are restricted such that our predictors *can only output discrete class labels*, as we have with Decision Trees or k-nearest neighbour classifiers. In this case, the outputs have *no intrinsic ordinality* between them, and so the concept of "covariance" is not so simple. This non-ordinality also implies that we have to change our combination function—a popular one is *majority voting* between the individual votes. The harder question can therefore be phrased as, *"how can we quantify diversity when our predictors output non-ordinal values and are combined by a majority vote?".*

A step toward understanding this question can be taken by considering where the bias-variance-covariance decomposition comes from: it falls neatly out of the bias-variance decomposition of the ensemble error. However, when our classification of a datapoint is either correct or incorrect, we have a zero-one loss function (instead of the usual quadratic loss function we used for the regression context). A number of authors have attempted to define a bias-variance decomposition for zero-one loss functions [68, 67, 14, 42], each with their own assumptions and shortcomings. Most recently Domingos [33] and James [58] propose general definitions which include the original quadratic loss function as a special case. This leads us naturally to ask the question, *does there exist an analogue to the bias-variance-covariance decomposition that applies for zero-one loss functions?.* If so, its formulation of the "covariance" term will be a major stepping stone in our understanding of the role of classification error diversity. The optimal classification error diversity will then be understood in terms of this trade-off for zero-one loss functions. This issue will be further discussed in the Conclusions chapter.

Taking all this into account, there is simply no clear analogue of the bias-variance-covariance decomposition when we have a zero-one loss function. We instead have a number of highly restricted theoretical results, each with their own assumptions that are probably too strong to hold in practice. We first describe the very well-known work by Tumer and Ghosh, on combining posterior probability estimates (ordinal values), and then turn to considering the harder question of non-ordinal outputs.

Ordinal Outputs

Tumer and Ghosh [139, 140] provided a theoretical framework for analysing the simple averaging combination rule when our predictor outputs are estimates of the posterior probabilities of each class. We already briefly summarised this in section 2.2.1 when we described how to compute optimal convex combination weights for such predictors. We will now describe this framework in more detail.

Regard figure 3.3. For a one dimensional feature vector x, the solid curves show the true posterior probabilities of classes a and b, these are $P(a)$ and $P(b)$, respectively. The dotted curves show *estimates* of the posterior probabilities, from one of our predictors, these are $\hat{P}(a)$ and $\hat{P}(b)$. The solid vertical line at x_* indicates the *optimal decision boundary*, that is the boundary that will minimise error given that our posterior probabilities overlap. This overlap, the dark shaded area, is termed the *Bayes error*, and is an irreducible quantity. The dotted vertical line at $\hat{x}_*$ indicates the boundary placed by our predictor, which is a certain distance from the optimal. The light shaded area indicates the *added error* that our predictor makes in addition to the Bayes error.

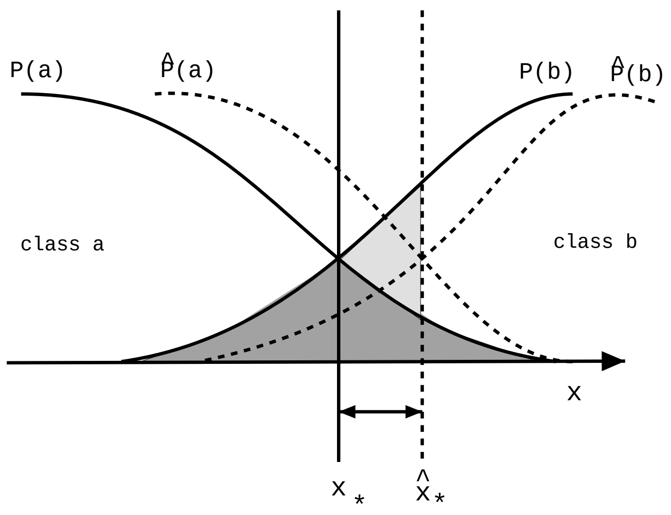

Figure 3.3: Tumer & Ghosh's framework for analysing classifier error

The individual predictor i approximates the posterior probability of class a as:

$$\hat{P}_i(a|x) = P(a|x) + \eta_i(a|x) \tag{3.11}$$

$P(a|x)$ is the true posterior probability of class a and $\eta_i(a|x)$ is the estimation error. Let us assume the estimation errors on different classes a and b are independent and identically distributed random variables [139, p5] with zero mean and variance $\sigma^2_{\eta_i}$. Consider the predictor's *expected added error* in distinguishing classes a and b, i.e. the *expected* size of the light shaded area given the variance $\sigma^2_{\eta_i}$. This can be stated as:

$$E_{add,i} = \frac{2\sigma^2_{\eta_i}}{P'(a|x) - P'(b|x)} \tag{3.12}$$

where $P'(a|x)$ is the derivative of the true posterior probability of class a. If the decision boundary was placed by an ensemble of predictors, Tumer and Ghosh show the expected added error of the ensemble estimator is:

$$E^{ens}_{add} = E_{add}\Big(\frac{1 + \delta(M-1)}{M}\Big) \tag{3.13}$$

where M is the number of classifiers. E_{add} is the expected added error of the individual classifiers: they are assumed to have the same error. δ is a correlation coefficient (see [139] for details). If δ is zero, i.e. the classifiers in the ensemble are statistically independent, we have $E^{ens}_{add} = \frac{1}{M}E_{add}$, i.e. the error of the ensemble will be M times smaller than the error of the individuals. If δ is 1, i.e. perfect correlation, then the error of the ensemble will just be equal to the errors of the individuals.

A further assumption made in this work is that the estimation errors of the different classifiers have the same variance $\sigma^2_{\eta_i}$. Another, possibly less critical assumption is that the posterior probabilities are monotonic around the decision boundary. This work has recently been extended by Roli and Fumera [114, 44], allowing for some of the assumptions to be lifted, the most important of which is that it allows for the estimators to exhibit unequal error variances (i.e. unequal performances on testing data). This demonstrates that the understanding of this particular diversity formulation (when outputting posterior probabilities) *is progressing.* What seems to be a sticking point for ensemble research is the non-ordinal output case, as we will now illustrate.

Non-Ordinal Outputs

In ensemble research, Hansen and Salamon [49] is seen by many as the seminal work on diversity in neural network classification ensembles. They stated that a necessary and sufficient condition for a majority voting[3] ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are *accurate* and *diverse.* An accurate classifier is one that has an error rate of better than random guessing, on new input. Two classifiers are diverse if they make different errors on new data points. They used the binomial theorem to explain this; we assume that all networks arrive at the correct classification with probability $(1-p)$, and that they make statistically independent errors. The chances of seeing exactly k errors among the M networks is:

$$\binom{M}{k} p^k (1-p)^{(M-k)} \tag{3.14}$$

which gives the following probability of the majority voted ensemble being in error:

$$\sum_{k>(M/2)}^{M} \binom{M}{k} p^k (1-p)^{(M-k)} \tag{3.15}$$

For example, with an ensemble of 21 predictors, combined with a majority vote, there would have to be at least 11 members wrong in order for the whole ensemble to be wrong. The area under the binomial distribution for 11 or more, and therefore the probability of the ensemble being incorrect, is 0.026. It should be stressed though, this *only* applies when the predictors are *statistically independent*, an assumption that is too strong to ever hold in practice, but it does represent the theoretical (if unachievable) ideal.

From this point onwards, when referring to "classification error diversity", it can be assumed that we are referring to this difficult task of quantifying non-ordinal output diversity. From the description of the problem as we have clearly stated, it seems deriving an expression for classification diversity is not as easy as it is for regression. The ideal situation would be a parallel to the regression case, where the squared error of the ensemble can be re-expressed in terms of the squared errors of the individuals and a term that quantifies their correlation. We would like to have an expression that similarly decomposes the classification error rate into the error rates of the individuals and a term that quantifies their 'diversity'. At present, this is beyond the state of the art, however a number of empirical investigations have gone into deriving heuristic expressions that may approximate this unknown diversity term.

A Heuristic Metric for Classification Error Diversity?

A number of authors have tried to qualitatively define classification error 'diversity'. Sharkey [123] suggested a scheme by which an ensemble's pattern of errors could be described in

[3]They also present results for plurality voting, which turns out to be far more complex.

terms of a *level of diversity.* Four levels were proposed, each one describing the incidence and effect of coincident errors amongst the members of the ensemble and the degree to which the target function is covered by the ensemble. A *coincident error* occurs when, for a given input, more than 1 ensemble member gives an incorrect answer. Function *coverage* is an indication of whether or not a test input yields a correct answer on ANY of the neural networks in the ensemble.

Level 1 No coincident errors, the target function is covered. Majority vote always produces the correct answer.

Level 2 Some coincident errors, but the majority is always correct and the function is completely covered. The ensemble size must be greater than 4 for this to hold.

Level 3 A majority vote will not always yield the right answer, but the members of the ensemble cover the function such that at least one always has the correct answer for a given input.

Level 4 The function is not always covered by the members of the ensemble.

Sharkey acknowledges that ensembles exhibiting level 2 or 3 diversity could be "upwardly mobile" as it is possible that an ensemble labelled as having level 2 diversity could contain a subset of neural networks displaying level 1 diversity using the test set and a level 3 ensemble could contain subsets of neural networks displaying level 1 and/or level 2 diversity on the test set, thus removal of certain networks could result in a change of diversity level for the better.

Carney and Cunningham [25] suggested an entropy-based measure, though this does not allow calculation of an individual's contribution to overall diversity. Zenobi and Cunningham [157] proposed a measure of *classification ambiguity.* The ambiguity of the *ith* classifier, averaged over N patterns, is

$$A_i = \frac{1}{N}\sum_{n=1}^{N} a_i(\mathbf{x}_n) \tag{3.16}$$

where $a_i(\mathbf{x}_n)$ is 1 if the output of individual i disagrees with the ensemble output, and 0 otherwise. The overall ensemble ambiguity is:

$$\bar{A} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{M}\sum_{i=1}^{M} a_i(\mathbf{x}_n) \tag{3.17}$$

The vast majority of empirical evidence examining classifier diversity is due to Kuncheva [72, 73, 74, 75, 76, 79, 129, 130]. These studies have explored several measures of diversity from the numerical taxonomy literature.

They emphasize two styles of measuring diversity, pairwise and non-pairwise. Pairwise measures calculate the average of a particular distance metric between all possible pairings of classifiers in the ensemble. Which distance metric is used therefore determines the characteristics of the diversity measure. The non-pairwise measures either use the idea of entropy or calculate a correlation of each ensemble member with the averaged output. Among the myriad of metrics studied, was the *Q-statistic*, which we will now consider. Take two classifiers, f_i and f_j. Over a large set of testing patterns, they will exhibit certain coincident errors, and therefore a probability of error coincidence can be calculated. Table 3.1 illustrates this, assigning a label a,b,c, or d to each type of coincidence.

	f_i correct	f_j wrong
f_i correct	a	b
f_i wrong	c	d

Table 3.1: Probabilities of coincident errors between classifier f_i and f_j. It should be noted, by definition, $a + b + c + d = 1$.

The Q statistic between classifier i and classifier j is:

$$Q_{i,j} = \frac{ad - bc}{ad + bc} \tag{3.18}$$

The Q statistic overall for an ensemble is calculated by taking the average Q-value from all possible pairings of the ensemble members. In addition to the Q statistic, several other metrics were examined. Extensive experimental work was conducted to find a measure of diversity that would correlate well with majority vote accuracy. In a summary paper of her own work, Kuncheva states:

"although a rough tendency was confirmed ... no prominent links appeared between the diversity of the ensemble and its accuracy. Diversity alone is a poor predictor of the ensemble accuracy." [74]

It can be argued that we should not in fact be looking for a metric which will "correlate well with ensemble accuracy", as many authors seem to be searching for. In a regression context, the portion of the ensemble error corresponding to diversity is the covariance term. It can be seen that this does not "correlate well" with the mean squared error, since it is intrinsically *part* of the ensemble error. Liu [85] conducted studies of the bias, variance and covariance of an ensemble, illustrating the nature of this three-way trade-off. This work showed that diversity (the covariance term) can be pushed too far, causing the other two components to rise. We emphasize that most probably the same principle will one day be shown for the ensemble majority vote accuracy. It can easily be shown in a hypothetical scenario, that maximum classification diversity can in be harmful if individual accuracy is not maintained. Take the hypothetical situation in table 3.2.

	A	B	C
Pattern 1	1	0	0
Pattern 2	0	1	0
Pattern 3	0	0	1
Pattern 4	0	0	0

Table 3.2: Error coincidences for three classifiers on a small hypothetical dataset. The presence of a one (1) in the table indicates that the classifier got the pattern correct, while a zero (0) indicates an error was made.

The situation in table 3.2 follows Kuncheva's method of analysing classifier diversity, in modelling the *error coincidences*. The presence of a one (1) in the table indicates that the classifier got that pattern correct, while a zero (0) indicates an error was made. For example, pattern 1 was correctly classified by B and C, but not A, and pattern 4 was misclassified by all three classifiers. The Q statistic for this arrangement is -1, maximum diversity, yet the ensemble majority vote accuracy is zero. This can be explained by the fact that the individual accuracies have dropped to 0.25. The ensemble has underperformed the individual classifiers due to this harmful diversity.

In spite of these seemingly intuitive definitions for diversity, none has yet *been proved* to have a definite link to overall ensemble error. It seems the amorphous concept of "diversity" is elusive indeed.

We have now reviewed the state of the field with regard to explanations of what the term 'diversity of errors' means, and why it can lead to improved ensemble performance. In conclusion, the community puts great stead in the concept of classification error "diversity", though it is still an ill-defined concept. The lack of a definition for diversity has not stopped researchers attempting to achieve it. So how do you make an effective, well-performing ensemble?

3.2 Towards A Taxonomy of Methods for Creating Diversity

We have determined in the previous section that in both regression and classification contexts, the correlation between the individual predictor outputs has a definite effect on the overall ensemble error, though for classification it is not yet formalised in the literature. In this section we attempt to move towards a possible way to understand the many different methods which researchers use to create an ensemble exhibiting error diversity; we show that these schemes can be categorised along three "diversity-creation" axes.

In constructing the ensemble, we may choose to take information about diversity into account, or we may not; i.e. we may or may not explicitly try to optimise some metric of diversity during building the ensemble. We make a distinction between these two types, *explicit* and *implicit* diversity methods. A technique such as Bagging is an *implicit method*, it randomly samples the training patterns to produce different sets for each network; at no point is a measurement taken to ensure diversity will emerge. Boosting is an *explicit method*, it directly manipulates the training data distributions to ensure some form of diversity in the base set of classifiers (although it is obviously not guaranteed to be the 'right' form of diversity).

During learning, a function approximator follows a trajectory in hypothesis space. We would like the networks in our ensemble to occupy different points in that hypothesis space. While implicit methods rely on randomness to generate diverse trajectories in the hypothesis space, explicit methods deterministically choose different paths in the space. In addition to this high level dichotomy, there are several other possible dimensions for ensuring diversity in the ensemble.

Sharkey [124] proposed that a neural network ensemble could be made to exhibit diversity by influencing one of four things: the initial weights, the training data used, the architecture of the networks, and the training algorithm used. This means providing each ensemble member with a *different* training set, or a *different* architecture, and so on. Though at first this seems a sensible way to group the literature, we found it difficult to group all ensemble techniques under these umbrellas. Specifically we could not see where a regularisation technique [117, 85] would fit. If we regularise the error function, we are changing none of Sharkey's four factors. Instead we came to the following categories upon which we believe the majority of neural network ensemble techniques can be placed.

Starting point in Hypothesis Space Methods under this branch vary the starting points within the search space, thereby influencing where in hypothesis space we converge to.

Set of Accessible Hypotheses These methods vary the set of hypotheses that are accessible by the ensemble. Given that certain hypotheses may be made accessible or inaccessible with a particular training subset and network architecture, these techniques either vary training data used, or the architecture employed, for different ensemble members.

Traversal of Hypothesis Space These alter the way we traverse the search space, thereby leading different networks to converge to different hypotheses.

3.2.1 Starting Point in Hypothesis Space

Starting each network with differing random initial weights will increase the probability of continuing in a different trajectory to other networks. This is perhaps the most common way of generating an ensemble, but is now generally accepted as the least effective method of achieving good diversity; many authors use this as a default benchmark for their own methods [101]. We will first discuss *implicit* instances of this axis, where weights are generated randomly, and then discuss *explicit* diversity for this, where networks are directly *placed* in different parts of the hypothesis space.

Sharkey, Neary and Sharkey [128] investigated the relationship between initialisation of the output weight vectors and final backpropagation solution types. They systematically varied the initial output weight vectors of neural networks throughout a circle of radius 10 and then trained them using the fuzzy XOR task with a fixed set of training data. The resulting networks differed in the number of cycles in which they took to converge upon a solution, and in whether they converged at all. However, the trained neural networks were not found to be statistically independent in their generalisation performance, i.e. they displayed very similar patterns of generalisation despite having been derived from different initial weight vectors. Thus, varying the initial weights of neural networks, although important when using a deterministic training method such as backpropagation, seems not to be an effective stand-alone method for generating error diversity in an ensemble of neural networks.

These observations are supported by a number of other studies. Partridge [107, 155] conducted several experiments on large ($> 150,000$ patterns) synthetic data sets, and concludes that after network type, training set structure, and number of hidden units, the random initialization of weights is the least effective method for generating diversity. Parmanto, Munro and Doyle [12] used one synthetic dataset and two medical diagnosis datasets to compare 10-fold cross-validation, Bagging, and random weight initializations; again the random weights method comes in last place.

We have now discussed implicit diversity methods for manipulating the starting point in hypothesis space. We will next discuss an explicit method for this, where randomisation of weights does not occur.

Maclin and Shavlik [89] present an approach to initializing neural network weights that uses competitive learning to create networks that are initialised far from the origin of weight space, thereby potentially increasing the set of reachable local minima; they show significantly improved performance over the standard method of initialization on two real world datasets.

A technique relevant to this discussion, Fast Committee Learning [132] trains a single neural network, taking M snapshots of the state of its weights at a number of instances during the training. The M snapshots are then used as M different ensemble members. Although the performance was not as good as when using separately trained nets, this offers the advantage of reduced training time as it is only necessary to train one network.

3.2.2 Set of Accessible Hypotheses

It can be argued that there are two ways to manipulate the set of hypotheses accessible to a network: firstly to alter the training data it receives, and secondly to alter the architecture of the network itself. We will now discuss these and how they have been used to create error diversity.

Manipulation of Training Data

Several methods attempt to produce diverse or complementary networks by supplying each network with a slightly different training set. This is probably the most widely investigated method of ensemble training. Regard figure 3.4; the frontmost bold square represents the training set for our ensemble. Different ensemble members can be given different parts of this set, so they will hopefully learn different things about the same task. Some methods will divide it by training pattern, supplying each member with all the K features, but a different subset of the rows (patterns). Other methods will divide it by feature, supplying each member with all the N patterns in the set, but each consists of a different subset of the columns (features). Both of these are termed *resampling methods*, and could provide overlapping or non-overlapping subsets of the rows or columns (or both) to different learners. Another alternative would be to pre-process the features in some way to get a different representation, for example using a log-scaling of the features. This can be viewed in our diagram as using a different plane, moving in the space of all possible features. The data techniques which transform features are termed *distortion methods* [127].

Duin and Tax [35] find that combining the results of one type of classifier on different feature sets is far more effective than combining the results of different classifiers on one feature set. They conclude that the combination of independent information from the different feature sets is more useful than the different approaches of the classifiers on the same data.

The most well-known resampling method is probably *k-fold cross-validation*. By dividing the dataset randomly into k disjoint pattern subsets, new overlapping training sets can be created for each ensemble member, by leaving out one of these k subsets and training on the remainder. The *Bagging* algorithm, is another example, randomly selecting N patterns *with replacement* from the original set of N patterns.

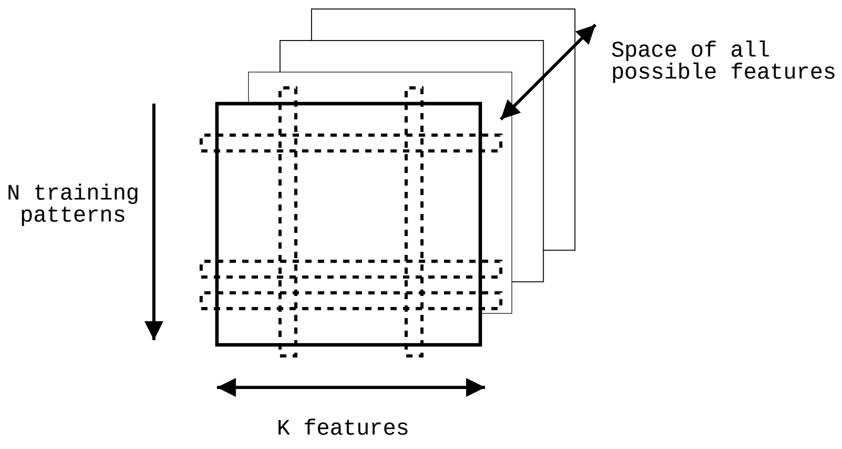

Figure 3.4: Space of possible training sets for an ensemble

Sharkey [125, 126] uses a distortion method to re-represent some features in an engine-fault diagnosis task. Two classifiers are provided with the original data to learn on, while a third is provided with the data after it has been passed through an *untrained* neural network. This essentially applies a random transformation to the features, yet Sharkey shows an ensemble using this technique can outperform an ensemble of classifiers using only the non-transformed data. Intrator and Raviv [113] report that simply adding Gaussian noise to the input data can help. They create a bootstrap resample, like Bagging, but then add a small amount of noise to the input vector. Several ensembles are then trained, using weight regularisation, to get a good estimate of the generalisation error. The process is then repeated with a different noise variance, to determine the optimal level. On test data, they show significant improvements on synthetic and medical datasets.

So far we have discussed how the input patterns could be resampled or distorted; Breiman [15] proposed adding noise to the *outputs* in the training data. This technique showed significant improvements over Bagging on 15 natural and artificial datasets; however when comparing to AdaBoost, no improvements were found.

We have now covered a number of papers which use randomisation of the training data to create diversity, therefore these are all so far *implicit diversity methods*. We will now turn to considering *explicit methods*, which deterministically change the data supplied to each network.

The very popular AdaBoost algorithm [40] explicitly alters the distribution of training data fed to each member. The distribution is recalculated in each round, taking into account the errors of the immediately previous network. Oza [104] presents a variant of AdaBoost that calculates the distribution with respect to *all networks* trained so far. In this way the data received by each successive network is explicitly 'designed' so that their errors should be diverse and compensate for one another.

Zenobi and Cunningham [157] use their own metric of classification diversity, as defined in equation (3.17) to select subsets of features for each learner. They build an ensemble by adding predictors successively, and use their metric to estimate how much diversity is in the ensemble so far. The feature subset used to train a predictor is determined by a hill-climbing strategy, based on the individual error and estimated ensemble diversity. A predictor is rejected if it causes a reduction in diversity according to a pre-defined threshold, or an increase in overall ensemble error. In this case a new feature subset is generated and another predictor trained. The DECORATE algorithm, by Melville and Mooney [97] utilises the same metric to decide whether to accept or reject predictors to be added to the ensemble. Predictors here are generated by training on the original data, plus a 'diversity set' of artificially generated new examples. The input vectors of this set are first passed through the current ensemble to see what its decision would be. Each pattern in the diversity set has its output re-labelled as the *opposite* of whatever the ensemble predicted. A new predictor trained on this set will therefore have a high disagreement with the ensemble, increasing diversity and hopefully decreasing ensemble error. If ensemble error is not reduced, a new diversity set is produced and a new predictor trained.

Oza and Tumer [105] present *Input Decimation Ensembles*, which seeks to reduce the correlations among individual estimators by using different subsets of the input features. Feature selection is achieved by calculating the correlation of each feature individually with each class, then training predictors to be specialists to particular classes or groups of classes. This showed significant benefits on several real [103] and artificial [105] datasets.

Liao and Moody [84] demonstrate an information-theoretic technique for feature selection, where all input variables are first grouped based on their mutual information [51, p492]. Statistically similar variables are assigned to the same group, and each member's input set is then formed by input variables extracted from different groups. Experiments on a noisy and nonstationary economic forecasting problem show it outperforms Bagging and random selection of features.

Several authors use domain knowledge to divide features between predictors. For example, Sharkey and Chandroth [126] use pressure and temperature information to indicate properties of an engine; and Wan [145] combines information from the fossil record with sunspot time series data to predict future sunspot fluctuations.

Most of the methods we have discussed manipulate *input* data. Dietterich and Bakiri [31] manipulate the output targets with *Error-Correcting Output Coding*. Each output class in the problem is represented by a binary string, chosen such that it is orthogonal (or as close as possible) from the representation of the other classes. For a given input pattern, a predictor is trained to reproduce the appropriate binary string. During testing, if the string produced by the predictor does not exactly match the string representing one of the classes, the Hamming distance is measured to each class, and the closest is chosen. Kong and Dietterich [68] investigate why this technique works. They find that, like Bagging, ECOC reduces the variance of the ensemble, but in addition can correct the bias component. An important point to note for this result is that the 0-1 loss bias-variance decomposition utilised assumes a Bayes rate of zero, i.e. zero noise.

Manipulation of Network Architectures

The number of investigations into using different types of neural network in ensembles is disappointingly small. If we want diverse errors in our ensemble, it makes sense that using different types of function approximator may produce this. Partridge [107, 106] concludes that variation in numbers of hidden nodes is, after initial weights, the least useful method of creating diversity in neural network ensembles, due to the methodological similarities in the supervised learning algorithms. However, the number of hidden nodes was only varied between 8 and 12, and on a single dataset; such a limited experiment indicates there is still some work to be done here. Partridge also used MLPs and radial basis functions in an ensemble to investigate the effect of network type on diversity, finding this was a more productive route than varying hidden nodes [155].

The vast majority of ensemble training methods are used to change the network weights. The structure of the ensemble, e.g., the number of NNs in the ensemble, and the structure of individual NNs, e.g., the number of hidden nodes, are all designed manually and fixed during the training process. While manual design of NNs and ensembles might be appropriate for problems where rich prior knowledge and an experienced NN expert exist, it often involves a tedious trial-and-error process for real-world problems because rich prior knowledge and experienced human experts are hard to get in practice. Opitz and Shavlik's Addemup algorithm [102], used an evolutionary algorithm to optimise the network topologies composing the ensemble. Addemup trains with standard backpropagation, then selects groups of networks with a good error diversity according to the measurement of diversity. Another recently proposed algorithm, CNNE [53], constructively builds an ensemble, monitoring diversity during the process. CNNE simultaneously designs the ensemble architecture along with training of individual NNs, whilst directly encouraging error diversity. It determines automatically not only the number of NNs in an ensemble, but also the number of hidden nodes in individual NNs. It uses incremental training based on Negative Correlation learning [85] in training individual NNs. It is entirely possible for an ensemble consisting of networks with very different architectures to emerge in such incrementally constructed ensembles.

Although this thesis is focusing on combining neural networks, for completeness of this section we feel it is necessary to mention that a few experiments have been done with *hybrid ensembles*. Wang et al [146] combined decision trees with neural networks. They found that when the neural networks outnumbered the decision trees, but there was at least 1 decision tree, the system performed better than any other ratio. Langdon [82] combines decision trees with neural networks in an ensemble, and uses Genetic Programming to evolve a suitable combination rule. Woods et al [150] combines neural networks, k-nearest neighbour classifiers, decision trees, and Quadratic Bayes classifiers in a single ensemble, then uses estimates of local accuracy in the feature space to choose one classifier to respond to a new input pattern.

Conclusions from the studies on hybrid ensembles seem to indicate that they will produce estimators with differing specialities and accuracies in different regions of the space—it seems sensible that two systems which represent a problem and search the space in radically different ways will show different strengths, and therefore different patterns of generalisation. This specialisation implies that with hybrid ensembles, *selection* of a single estimator rather than *fusion* of the outputs of all estimators may be more effective. The dynamic selection method by Woods et al [150] could easily be applied to an ensemble containing networks with differing numbers of hidden nodes, having first used an algorithm like CNNE [53] to ensure a the network architectures are established appropriately to ensure a diversity of errors.

3.2.3 Hypothesis Space Traversal

Given a particular search space, defined by the architecture of the network and training data provided, we could occupy any point in that space to give us a particular hypothesis. How we choose to traverse the space of possible hypotheses determines what type of ensemble we will end up with.

Regularisation Methods

Some authors have found benefit from using a *regularisation term* in the error function for the network. The justification from this stems from Tikhonov's work on *ill-posed problems* [133]. This showed that including another term in the error function, in addition to the objective function, could effectively control the bias-variance trade-off[4]. Using this approach, the error of network i becomes:

$$e_i = \frac{1}{2}(f_i - d)^2 + \lambda R \tag{3.19}$$

where λ is a weighting parameter on the regularisation term R. The λ parameter controls a trade-off between the two terms; with $\lambda = 0$ we would have an ensemble with each network training with plain backpropagation, and as λ increases more and more emphasis would be placed on minimising whatever the regularisation term is chosen to be.

The usual method of Tikhonov Regularisation uses derivatives of the objective function to make up the regularisation term. However we can employ the same principle to control a trade-off between *two* objectives: individual accuracy and ensemble diversity. If we want to enforce diversity, the regularisation term can quantify the amount of correlation in some way, so that it can be minimised explicitly during the training—as such, this is a type of *explicit* diversity method. Rosen [117] used a regularisation term:

$$R = \sum_{j=1}^{i-1} c(j,i) p_i \tag{3.20}$$

where $c(j,i)$ is an *indicator function* specifying decorrelation between networks j and i, and p_i is a penalty function:

$$p_i = (f_i - d)(f_j - d) \tag{3.21}$$

the product of the ith and jth network biases. The indicator function $c(j,i)$ specifies which networks are to be decorrelated. To penalise a network for being correlated with the previous trained network, the indicator function is:

$$c(j,i) = \begin{cases} 1 & \text{if} \quad i = j-1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.22}$$

Negative Correlation (NC) Learning [85] extended Rosen's work by training the networks in parallel. NC has a regularisation term:

$$R = p_i = (f_i - \bar{f}) \sum_{j \neq i} (f_j - \bar{f}) \tag{3.23}$$

where $\bar{f}$ is the average output of the whole ensemble of M networks at the previous timestep. NC has seen a number of empirical successes [85, 86, 87], consistently *outperforming* a simple ensemble system. Figure 3.5 shows the algorithm assuming a gradient descent method is used to learn the predictors. An important point to note here is that Liu calculates the derivative using the assumption that $\bar{f}$ is a constant with respect to f_i; the exact details and implications of this assumption will be considered in the next chapter.

[4]Bishop [9] showed that training a neural network with noise added to the input patterns was equivalent to Tikhonov Regularization

```
1. Let $M$ be the final number of predictors required.
2. Take a training set $t = \{(\mathbf{x}_1, d_1), ..., (\mathbf{x}_N, d_N)\}$.
3. For each training pattern from $n = 1$ to $N$ do :
   (a) Calculate $\bar{f} = \frac{1}{M}\sum_i f_i(\mathbf{x}_n)$
   (b) For each network from $i = 1$ to $M$ do:
       • Perform a single update for each weight in network $i$ using:
           $e_i = \frac{1}{2}(f_i(\mathbf{x}_n) - d_n)^2 + \lambda(f_i(\mathbf{x}_n) - \bar{f})\sum_{j \neq i}(f_j(\mathbf{x}_n) - \bar{f})$
           $\frac{\partial e_i}{\partial f_i} = (f_i(\mathbf{x}_n) - d_n) - \lambda(f_i(\mathbf{x}_n) - \bar{f})$

For any new testing pattern $\mathbf{x}$, the ensemble output is given by:
   $\bar{f} = \frac{1}{M}\sum_i f_i(\mathbf{x})$
```

Figure 3.5: The Negative Correlation Learning Algorithm

McKay and Abbass [95, 94] recently presented *Root Quartic Negative Correlation Learning* (RTQRT), based on an alternative penalty term. The technique was applied to a genetic programming system, and shown to outperform standard NC on larger ensembles. The penalty term used was:

$$p_i = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(f_i - d)^4}, \tag{3.24}$$

the derivative of this is:

$$\frac{\partial p_i}{\partial f_i} = \frac{1}{\sqrt{M}} \times \frac{2\sum_{i=1}^{M}(f_i - d)^3}{\sqrt{\sum_{i=1}^{M}(f_i - d)^4}} \tag{3.25}$$

Standard backpropagation presents a certain error landscape to an ensemble system. NC and RTQRT add penalty terms to this, providing each network within the ensemble with a different landscape to work in. It has been shown that the minimisation of error in each of these landscapes can reduce error further than using the normal unregularised landscapes.

Evolutionary Methods

The term "diversity" is also often used in the evolutionary computation literature, in the context of maintaining diversity in the population of individuals you are evolving [1, 29, 64]. This has a very different meaning to the concept of diversity as discussed in this thesis. In evolutionary algorithms, we wish to maintain diversity in order to ensure we have *explored a large area of the search space* and not focussed our search onto an unprofitable area. When we decide our search has continued for long enough, we will typically take the best performing individual found so far, *ignoring the other individuals in the population.* The optimal diversity in this context will be that which optimises the explore-exploit trade off, such that the best points in the search space are found quickly and efficiently. This is in contrast to ensemble diversity methods, which create diversity with the intention that the 'population' of ensemble members will *be combined.* As such, evolutionary diversity methods do not concern themselves with creating a population that is *complementary* in any way, but instead with just ensuring the maximum amount of the hypothesis space is being explored in order to find the best single individual.

In spite of these differences, some researchers have found interesting parallels. Yao [153] evolves a population of neural networks, using *fitness sharing* techniques to encourage diversity, then combines the *entire* population as an ensemble instead of just picking the best individual.

3.2.4 The Ambiguity Family

Krogh and Vedelsby [70] provided the Ambiguity decomposition, showing that the ensemble error could be broken down into two terms, one of which is dependent on the correlations between networks. In this section we define the *Ambiguity Family* of ensemble methods, all of which exploit Krogh and Vedelsby's result. Since they all measure diversity explicitly, they are a subset of the previously defined *explicit diversity methods.*

In the last few years, the ambiguity decomposition has quietly been utilised in almost every aspect of ensemble construction. Krogh and Vedelsby themselves developed an active learning scheme [70], based on the method of query by committee, selecting patterns to train on that had a large ambiguity; this showed significant improvements over passive learning in approximating a square wave function. In the same paper an estimate of ambiguity is used to optimise the ensemble combination weights; this showed in some cases it is optimal to set a network weight to zero, essentially removing it from the ensemble.

Opitz [100] selected feature subsets for the ensemble members to train on, using a genetic algorithm with an ambiguity-based fitness function; this showed gains over Bagging and Adaboost on several classification datasets from the UCI repository. A precursor to this work was Opitz and Shavlik's Addemup algorithm [102], which used the same fitness function to optimise the network topologies composing the ensemble. Addemup trains with standard backpropagation, then selects groups of networks with a good error diversity according to the measurement of ambiguity.

We can see that Ambiguity has been utilised in many ways: pattern selection [70], feature selection [100], optimising the topologies [102] of networks in the ensemble, and optimising the combination function [70].

3.3 Chapter Summary

In this chapter we reviewed the existing theoretical and heuristic explanations of what error "diversity" is, and how it affects the error of the overall ensemble. We described the two most prominent theoretical results for regression ensembles: the Ambiguity Decomposition and the bias-variance-covariance decomposition. We demonstrated what we believe to be the first formal link between these two, showing how they relate to each other. This link turns out to be crucial to further results we present in chapter 5.

We described the current state of the art in formalising the concept of diversity for classification ensembles, illustrating the clearly that the problem is actually one of quantifying some form of correlation between non-ordinal predictor outputs. In the process of this we show a paradox in that though diversity is widely sought after, it is still ill-defined. We then suggested that an analogue of the bias-variance-covariance decomposition that applies for 0-1 loss functions may be a useful research direction in this area; this will be further discussed in the Conclusions chapter.

Following on from this review of theoretical results supporting diversity, we reviewed practical techniques that have been applied to create well-performing ensembles exhibiting diversity. We suggested a possible way to structure the field, based on how techniques choose to create error diversity—we noted that this improves upon a previously suggested taxonomy [124] of neural network ensembles. As a parallel to this taxonomy, we introduced the *Ambiguity* family of methods, all of which exploit the Ambiguity decomposition of the ensemble error.

In the Introduction chapter we described our motivation for this work: *to better understand "diversity" and how to create it.* One particular learning technique we mentioned in section 3.2.3 claims to *enforce* diversity (both regression and classification) with a regularisation term in the error function. This is Negative Correlation Learning [85], and will be our focal point for the remainder of the thesis.

Chapter 4

Parameter Selection in Negative Correlation Learning

In the previous chapter we introduced Negative Correlation (NC) learning [85]. This technique utilises a regularisation term added to the original backpropagation error function to penalise correlated errors between networks in the ensemble, which serves to encourage lower overall ensemble error. The amount of regularisation is controlled by a strength parameter, λ; optimising this parameter with respect to the mean squared error can bring statistically significant improvements in performance [18, 85, 86, 94, 152]. Previous work by Liu [85] and other authors has shown that the optimum λ value is task-dependent, and can also vary with the configuration of the ensemble. Liu [85] reports experiments with the λ values $0.0, 0.25, 0.5, 0.75$ and 1.0, but no values outside this range or any intermediate values. Liu does present a hypothesis that the bounds of λ should be $[0, 1]$, but this makes an assumption that can be questioned, as will be detailed in chapter 5. In fact we have no definite evidence about the shape or range of the λ search landscape. It may well be the case that the landscape is highly rugged, meaning the optimum may be difficult to locate, or there may be multiple optima with equivalent error values. It may also be the case optimum values can exist beyond $[0, 1]$. In general, we would like to understand the behaviour of λ under different circumstances.

What options do we have for understanding the λ landscape? We could perform an exhaustive search; however to do this within a reasonable amount of time we would need to make assumptions about the range to search, for example between -1 and $+1$, and also on the resolution at which we will sample the space. We may end up overlooking an optimum, or missing it entirely if it is outside our assumed range. We could perform a gradient descent or hill climbing strategy; however it is well known that these approaches are prone to being caught in local minima. Alternatively, we could use an *evolutionary search algorithm* [4]. This type of algorithm conducts a parallelized stochastic search over a space, and can often escape local minima; in addition, an appropriately chosen evolutionary technique could efficiently explore values outside the $[0, 1]$ range and possibly provide solutions we had not considered. In the following section we employ an evolutionary algorithm to set λ for various different ensemble configurations and datasets.

4.1 An Evolutionary Approach to Setting λ

4.1.1 Method

Evolutionary algorithms can search a space of solutions to a given problem by distributing a *population* of candidate solutions (called 'individuals') over the space and using principles of natural evolution to 'breed' this population. The performance of an individual is assessed by a pre-defined *fitness function*. A special case of evolutionary algorithms is *evolutionary programming*—this is especially suited to optimisation of real-valued numbers. We use the Improved Fast Evolutionary Programming (IFEP) algorithm [151] to optimise the λ parameter. We need to explore the search space of possible λ values thoroughly in order to be sure we have located the optimal setting; the IFEP algorithm is known for good exploration without getting stuck in local minima [151] and so is suited to our task. The IFEP algorithm is described in figure 4.1.

We use a population size of $P = 10$, and the algorithm was allowed to run for a total of 100 generations. The initial values of the λ_i variables were chosen from a Gaussian distribution with mean zero and variance one. The initial values of the η_i variables were all set at 3.0; this corresponds to the standard deviation of the Gaussian mutations, so 3.0 means it is possible for a mutation to produce a new λ value up to 3.0 standard deviations away from a parent.

The fitness function to be maximised is the reciprocal of the mean squared error on the testing data:

$$F(\lambda) = \frac{1}{\frac{1}{N}\sum_{n=1}^{N}(\bar{f}_\lambda(\mathbf{x}_n) - d_n)^2} \tag{4.4}$$

where N is the number of testing patterns, $\bar{f}_\lambda(\mathbf{x}_n)$ is the output of the ensemble on pattern n after training using NC with the relevant λ value, and d_n is the target for that pattern.

All networks are multilayer perceptrons, using learning rate $\alpha = 0.1$, with a single hidden layer; all hidden nodes use sigmoid activation functions. For dataset 1, we use a linear output activation function, while for dataset 2 we use a sigmoid output activation function. The ensemble is combined by a uniformly weighted linear combination.

Dataset 1: Boston Housing

This regression dataset concerns housing values in suburbs of Boston, the problem is to predict the median house price given a number of demographic features. There are 506 examples, each containing 13 input variables (12 continuous, 1 binary), and 1 continuous output variable in the range 0 to 50. All input variables were linearly rescaled, independently of each other, to be in the range $[0, 1]$, and the output variable was linearly rescaled to $[-1, 1]$. Networks were trained for 1000 iterations; we used 5-fold cross-validation, with each fold being evaluated from 40 trials of random weights, giving a total of 200 trials for each experiment. Whenever mentioned, the mean squared error on this dataset refers to an average over these 200 trials.

Dataset 2: Phoneme

This classification dataset is a French and Spanish phoneme recognition problem from the Elena project [47]; the problem is to distinguish between nasal (AN, IN, ON) and oral (A, I, O, E) vowels. The data has 5404 examples, each with 5 continuous valued inputs (the normalized amplitudes of the five first harmonics), and 1 binary output. The data is available in two forms, the original unscaled version, and a version where all input variables were rescaled to have zero mean and variance 1; we use the latter, rescaled form. Networks were trained for 500 iterations. We divided the dataset into 5 equal sized portions; we trained on one fifth of the data (1080 examples), then tested on the remaining four fifths (4324 examples). This was repeated 5 times so each portion was used in turn as training data. Each time, the ensemble was evaluated from 40 trials of random weights, giving a total of 200 trials for each experiment. Whenever mentioned, the mean squared error on this dataset refers to an average over these 200 trials.

4.1.2 Results

All results are reported as averages over 10 runs of the IFEP algorithm; on close examination of the evolutionary run, it was confirmed that the algorithm had explored and discarded several λ values well above 1.0 and well below 0.0. The near optimal values found for λ were confirmed in each case as providing a statistically significant gain (two-tailed t-test, $\alpha = 0.05$) over $\lambda = 0.0$.

Optimal λ as we add more networks to the ensemble

Figures 4.2 and 4.3 show results as we increase the size of the ensemble; the two datasets seem to exhibit opposite trends for optimal λ. On the Phoneme dataset, using simple networks (2 hidden nodes), the value seems relatively constant at ~ 0.93, but changes and becomes more variable with more complex networks. On the Boston dataset, the converse seems to be true; λ_* stabilizes to ~ 0.9 when we use more complex networks. The pattern that seems to be common between the two datasets shows up using the largest ensemble, 12 networks; in this case, $\lambda_* \approx 0.9$, relatively invariant to the complexity of the networks.

Optimal λ as we add more hidden nodes to the networks

Figures 4.4 and 4.5 show results as we increase the complexity of the ensemble members. On the Boston dataset, when we have just two networks, then adding more hidden nodes per network seems to push optimal λ up; however on the Phoneme data, we see the opposite, pushing the optimal value *down.* For both datasets, when we have a large ensemble, then adding hidden nodes to networks caused a slight downward trend in λ_*.

1. Generate the initial population of P individuals randomly, and set $g = 0$. Each individual is taken as a pair of real valued numbers (λ_i, η_i), $\forall i \in \{1, \cdots, P\}$, where λ_i's are the objective variables to be optimised and η_i's are strategy parameters that determine the search step size of mutation.

2. Evaluate the fitness score for each individual in this $g = 0$ initial random population (λ_i, η_i), $\forall i \in \{1, \cdots, P\}$, of the population based on the fitness function, $F(\lambda_i)$.

3. Each parent (λ_i, η_i), $i = 1, \cdots, P$, creates two offspring $({\lambda_i}'^N, {\eta_i}')$ and $({\lambda_i}'^C, {\eta_i}')$ by:

$$\begin{aligned} {\eta_i}' &= \eta_i \exp(\tau' N(0,1) + \tau N(0,1)), && (4.1)\\ {\lambda_i}'^N &= \lambda_i + {\eta_i}' N(0,1), && (4.2)\\ {\lambda_i}'^C &= \lambda_i + {\eta_i}' C(0,1), && (4.3) \end{aligned}$$

   $N(0,1)$ denotes a normally distributed one-dimensional random number with mean zero and standard deviation one. $C(0,1)$ means a Cauchy distributed random number with mean zero and scaling parameter $t = 1$. The factors τ and τ' are set to $\left(\sqrt{2\sqrt{l}}\right)^{-1}$ and $\left(\sqrt{2l}\right)^{-1}$, where $l = 1$, the dimension of the objective variable we are evolving, as in Bäck and Schwefel [4].

4. Calculate the fitness of each offspring pair $({\lambda_i}'^N, {\eta_i}')$ and $({\lambda_i}'^C, {\eta_i}')$, $\forall i \in \{1, \cdots, P\}$. For each pair of Cauchy and Gaussian offspring produced from the same parent, discard the one with the lowest fitness.

5. Select the best P individuals from the set of all remaining offspring and parents to become the next generation.

6. Stop if the halting criterion is satisfied; otherwise, $g = g + 1$ and go to Step 3.

Figure 4.1: Improved Fast Evolutionary Programming Algorithm, with Truncation Selection

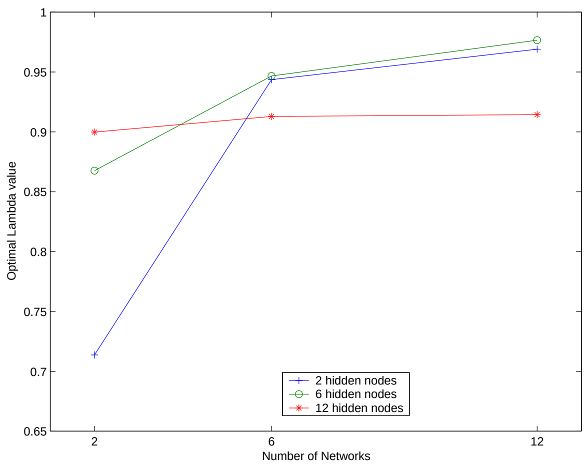

Figure 4.2: Optimal λ values for Boston dataset, varying number of networks in the ensemble

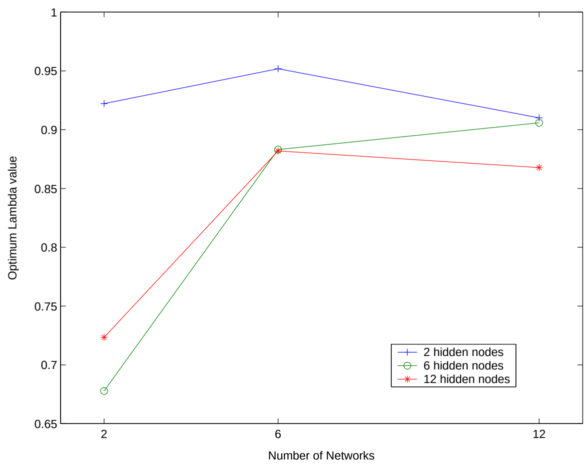

Figure 4.3: Optimal λ values for Phoneme dataset, varying number of networks in the ensemble

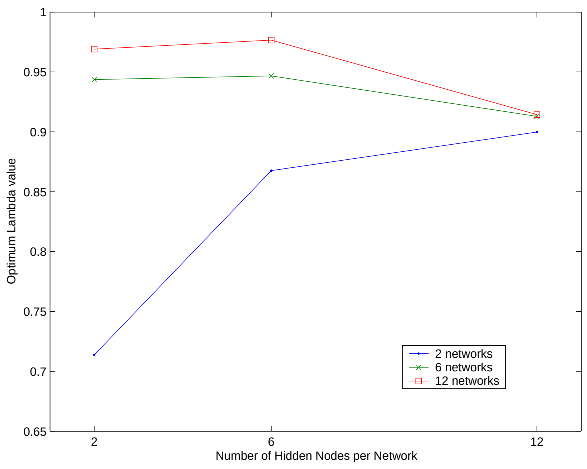

Figure 4.4: Optimal λ values for Boston dataset, varying complexity of networks in the ensemble

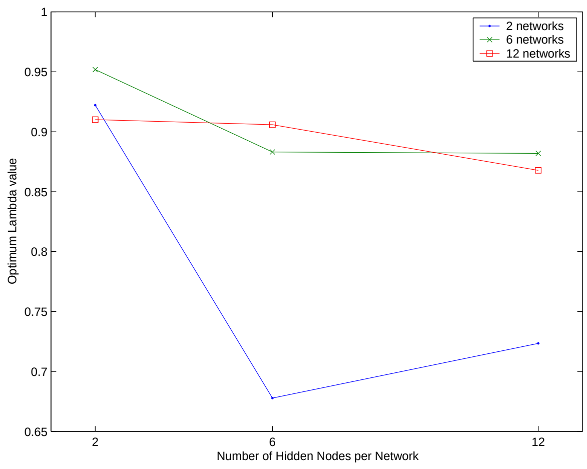

Figure 4.5: Optimal λ values for Phoneme dataset, varying complexity of networks in the ensemble

Using a different Fitness Function

The results reported so far have been obtained using the mean squared error in the fitness function. The Phoneme dataset is a classification problem; as such, we are not so concerned with the MSE, but more in the *classification error rate* (percentage of misclassified examples). If this is the case, we should use a different fitness function that will optimise the error rate instead of the MSE. An experiment was performed to evolve λ using the testing classification error rate in the fitness function; our new function is:

$$F(\lambda) = \frac{1}{\frac{1}{N}\sum_{n=1}^{N} L(\bar{f}_\lambda(\mathbf{x}_n); d_n)} \tag{4.5}$$

where $L(\bar{f}_\lambda(\mathbf{x}_n); d_n)$ returns 1 if $\bar{f}_\lambda(\mathbf{x}_n) \neq d_n$, and 0 otherwise. Figures 4.6 and 4.7 show results for this. As we increase the number of networks, there is a uniform upward trend towards 0.9, regardless of network complexity. As we increase network complexity, there seems to be a significant variability when using a small ensemble, but stabilizing again to ~ 0.9 when we use a larger ensemble.

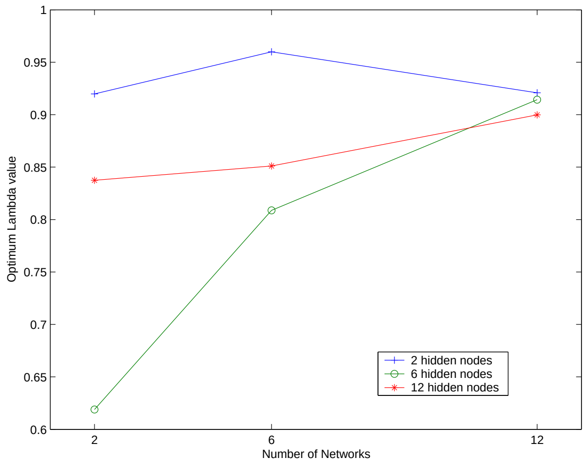

Figure 4.6: Optimal λ values for Phoneme dataset, varying number of networks in the ensemble, evaluted with respect to classification error rate

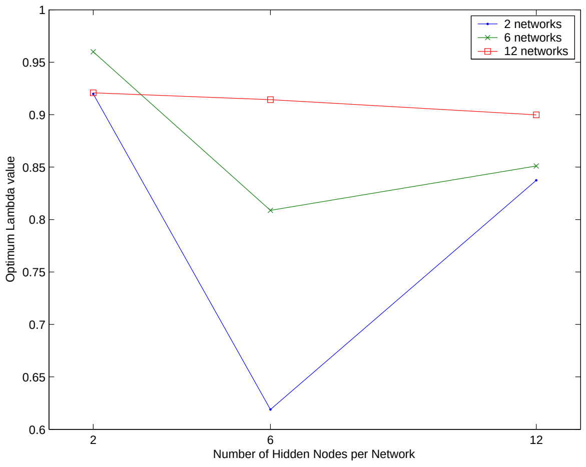

Figure 4.7: Optimal λ values for Phoneme dataset, varying complexity of networks in the ensemble, evaluated with respect to classification error rate

4.2 Chapter Summary

In this chapter we applied an evolutionary algorithm to locate the optimal value of the regularisation strength parameter λ in NC learning. We investigated this with two datasets and under several different ensemble architectures. The aim was to characterise the optimal value, λ_*, such that it could be predicted reliably in future situations.

What conclusions can we draw about the characteristics of λ_*? On both datasets, the optimal value had more variability on smaller ensembles, and was more stable with larger ensembles, regardless of which error criterion[1] was used. There was no definitive pattern observed with respect to changes in complexity of the individual networks (i.e. varying the number of hidden nodes). After averaging over several trials, near optimal values were located: the lowest was 0.62, and the largest 0.98. This chapter's investigation has therefore served to support Liu's hypothesis that λ should be between 0 and 1; however we cannot make definite claims from just two datasets; a larger investigation of the empirical behaviour of NC learning is conducted in chapter 6.

A more desirable goal would be to find a mathematical framework in which to characterise NC learning and the behaviour of the strength parameter, and discover *why* optimal λ is affected as it is by the ensemble configuration; in the following chapter we address this.

[1]MSE or classification error rate

Chapter 5

Theoretical Analysis of Negative Correlation Learning

In this chapter we continue our study of Negative Correlation (NC) Learning, a successful neural network ensemble learning technique developed in the evolutionary computation literature. NC has shown a number of empirical successes and varied applications, including regression problems [152], classification problems [94], and time-series prediction [85]. It has consistently demonstrated significant performance improvements over a simple ensemble system, showing very competitive results with other techniques like Mixtures of Experts and RBF networks [86]. NC so far has had very little formal analysis to explain its successes; this provides the motivation for our work.

Our theoretical analysis begins with a summary of the assumptions and properties of NC as published in the original work. We show that by removing an assumption, NC can be seen to be exploiting the well-known *Ambiguity* decomposition of the ensemble error, grounding it in a statistics framework around the bias-variance decomposition. We use this grounding to find bounds for the parameters, and provide insights into the behaviour of the optimal parameter values. When taking into account our new understanding of the algorithm, significant reductions in error rates were observed in empirical tests.

5.1 NC Learning: Why does it work?

It is well known that the MSE of a convex-combined ensemble can be decomposed into bias, variance and covariance terms [141]. Liu showed empirically [85] that the strength parameter λ in NC provides a way of controlling the trade-off between these three components. Liu showed that a value greater than zero can encourage decreases in covariance, however also observes that too high a value can cause a rapid increase in the variance component, causing overall error to be higher. No theoretical explanation was given for this behavior, and as such we do not yet have a clear picture of the exact dynamics of the algorithm.

When $\lambda = 1$, we have a special situation; the gradient of the individual network error is directly proportional to the gradient of the ensemble error. This was described by Liu to show a theoretical justification for NC-Learning. In order to show this, Liu calculated the first partial derivative of the network error function with respect to the output of the network. It should be noted that, in the calculation of the derivative, Liu has: *"... made* *use of the assumption that the output of the ensemble $\bar{f}$ has constant value with respect to f_i"* [85, p.29]. When $\lambda = 1$, we have:

$$
\begin{aligned}
e_i &= \frac{1}{2}(f_i - d)^2 + \lambda(f_i - \bar{f})\sum_{j \neq i}(f_j - \bar{f}) && (5.1)\\
\frac{\partial e_i}{\partial f_i} &= f_i - d + \sum_{j \neq i}(f_j - \bar{f})\\
&= f_i - d - (f_i - \bar{f})\\
&= \bar{f} - d
\end{aligned}
$$

However, although the assumption of constant $\bar{f}$ is used, so too is the property that:

$$\sum_{j \neq i}(f_j - \bar{f}) = -(f_i - \bar{f}) \qquad (5.2)$$

This is derived from the fact that the sum of deviations around a mean is equal to zero; obviously the sum of deviations around a constant (since Liu assumed $\bar{f}$ is as such) does not have this property, therefore (5.2) should not be used. Using this assumption (apparently inconsistently), and with the overall ensemble error function defined as,

$$e_{ens} = \frac{1}{2}(\bar{f} - d)^2 \qquad (5.3)$$

it was stated that:

$$\frac{\partial e_{ens}}{\partial f_i} = \frac{1}{M}\left[\frac{\partial e_i}{\partial f_i}\right] \qquad (5.4)$$

showing that the gradient of the individual network error is directly proportional to the gradient of the ensemble error. This means all the optima are in the same locations as in the ensemble error function, but the landscape is scaled by a factor of $\frac{1}{M}$. Though this is obviously a useful property, the justification for the assumption is unclear. The remainder of this work will illustrate the benefits that can be gained from removing this assumption that $\bar{f}$ is a constant term. Before we embark on this, it would be useful to first understand a framework into which NC can fit. Can we find a more solid theoretical grounding to NC?

5.2 Formalising NC Learning

In this section we show how NC can be related to the ambiguity decomposition [70] which showed that the mean-square error of the ensemble estimator is guaranteed to be less than or equal to the average mean-square error of the component estimators. They showed the ensemble error could be broken down into two terms, one of which is dependent on the correlations between network outputs; the exact nature of this result will be given in the next section.

5.2.1 NC uses the Ambiguity Decomposition

Note that the penalty function is actually a sum of pairwise correlations between the ith network output and the other network outputs:

$$\begin{aligned} p_i &= (f_i - \bar{f})(f_j - \bar{f}) \\ &\quad +(f_i - \bar{f})(f_k - \bar{f}) \\ &\vdots \\ &\quad +(f_i - \bar{f})(f_M - \bar{f}) \end{aligned} \tag{5.5}$$

If we remember that the MSE of an ensemble decomposes into bias plus variance plus covariance [141], then including some measure of correlation to be minimised seems like an intuitive thing to do, first noted by Rosen [117]. However this intuition is not enough to fully understand NC. We note that the penalty function can be rearranged to:

$$p_i = -(f_i - \bar{f})^2 \tag{5.6}$$

which is again due to the property that the sum of deviations around a mean is equal to zero. This rearrangement is only possible if we remove Liu's assumption of constant $\bar{f}$. As can be seen, each network minimises its penalty function by moving its output away from the ensemble output, the mean response of all the other networks.

So, why should increasing distance from the mean, or optimising equation (5.1), necessarily lead to a decrease in ensemble error? An examination of the proof by Krogh and Vedelsby can answer this question, and also raise some new questions on the setting for the λ parameter. Their work showed that the following statement about ensemble error was true:

$$(f_{ens} - d)^2 = \sum_i w_i (f_i - d)^2 - \sum_i w_i (f_i - f_{ens})^2 \tag{5.7}$$

where w_i is the weighting on the i^{th} network, and d is the target. This says that the *squared error of the ensemble estimator is equal to the weighted average squared error of the individuals, minus a term which measures average correlation.* This allows for non-uniform weights (with the constraint $\sum_i w_i = 1$) so the general form of the ensemble output is $f_{ens} = \sum_i w_i f_i$. This result in equation (5.7) stems from a number of definitions, one of which is the *ambiguity* of a single member of the ensemble:

$$v_i = (f_i - f_{ens})^2 \tag{5.8}$$

If the ensemble is uniformly weighted, we simply have:

$$v_i = (f_i - \bar{f})^2 \tag{5.9}$$

Remembering that the individual networks in NC-learning minimise the penalty function, and looking at equations (5.6) and (5.9) we can see $p_i = -v_i$ and so the networks are in fact maximising this ambiguity term, equation (5.9). This in turn of course affects the total ensemble error.

To understand this further we take equation (5.7), multiply through by $\frac{1}{2}$ and rearrange slightly assuming our ensemble is uniformly weighted, we then have:

$$\frac{1}{2}(\bar{f} - d)^2 = \frac{1}{M} \sum_i \left[\frac{1}{2}(f_i - d)^2 - \frac{1}{2}(f_i - \bar{f})^2 \right] \tag{5.10}$$

We see that the MSE of an ensemble can be decomposed, with the Ambiguity Decomposition, into a weighted summation, where the i^{th} term is the backpropagation error function plus the NC-Learning penalty function with the strength parameter set at 0.5.

Where does NC belong among other algorithms?

In section 3.2.4 we reviewed how various researchers have exploited the Ambiguity decomposition to guide their machine learning techniques; we saw that Ambiguity has been utilised in many ways: pattern selection [70], feature selection [100], optimising the topologies [102] of networks in the ensemble, optimising the combination function [70], and also optimising training time [21]. We can now see that NC fits neatly into the gap as the first technique to directly use Ambiguity for network weight updates.

5.2.2 How should we understand the Strength Parameter?

The Relation to Liu's λ

Since we have removed the constraint of assuming constant $\bar{f}$ to allow a link to the ambiguity decomposition, it seems more rigorous to differentiate the network error again without this assumption. What happens in this case? We introduce a strength parameter γ, in place of λ, to indicate we perform the derivative *correctly*; we calculate the partial derivative with respect to f_i as follows:

$$\begin{aligned} e_i &= \frac{1}{2}(f_i - d)^2 + \gamma(f_i - \bar{f})\sum_{j \neq i}(f_j - \bar{f}) \\ \frac{\partial e_i}{\partial f_i} &= f_i - d - \gamma\Big[2(1 - \frac{1}{M})(f_i - \bar{f})\Big] \end{aligned} \tag{5.11}$$

where M is the number of networks in the ensemble. Keeping the assumption of constant $\bar{f}$ causes this term $2(1 - \frac{1}{M})$ to disappear. However, it does seem sensible to retain this, as it takes account of the number of networks. In all of the previous work on NC [85, 86, 94], the strength parameter was thought to be problem dependent. Now we understand that it has a deterministic component, this $2(1 - \frac{1}{M})$. To avoid confusion, from this point on, we shall refer to the λ parameter in the following context, where γ is still a problem-dependent scaling parameter:

$$\lambda = \gamma\left[2(1 - \frac{1}{M})\right] = \gamma\left[2\frac{M-1}{M}\right] \tag{5.12}$$

Defining Parameter Bounds

As with any problem-dependent parameter, we would like to know bounds on it, to allow us to set it sensibly. Liu stated that the bounds of λ should be $[0, 1]$, based on the following calculation:

$$\begin{aligned} \frac{\partial e_i}{\partial f_i} &= f_i - d + \lambda\sum_{j \neq i}(f_j - \bar{f}) \\ &= f_i - d - \lambda(f_i - \bar{f}) \\ &= (1 - \lambda)(f_i - d) + \lambda(\bar{f} - d) \end{aligned}$$

He states: *"the value of parameter λ lies inside the range $0 \leq \lambda \leq 1$ so that both $(1-\lambda)$ and λ have non-negative values"* [85, p.29]. However this justification is questionable, and again here we see the assumption of constant $\bar{f}$ is violated.

We therefore have to ask, why would it be a problem if either $(1-\lambda)$ or λ took negative values? Maybe the bounds of λ should not be $[0,1]$. How can we determine a more justifiable bound? The NC penalty term 'warps' the error landscape of the network, making the global minimum hopefully easier to locate. If the landscape is warped too much, it could eliminate any useful gradient information that we can use. This state is indicated by the positive-definiteness of the Hessian matrix (the matrix of second partial derivatives of the network error with respect to the weights). If the Hessian matrix, evaluated at a given point, is non-positive definite, then the error gradient consists of either a local maximum or a point of inflexion, and we have lost any useful gradient information from our original objective function.

It is the case that, if the Hessian matrix is positive definite, then all elements on the leading diagonal are also positive [80]; therefore if any element on that diagonal is zero or less, the entire matrix *cannot* be positive definite. Assuming we have a network with a linear output function, then for an arbitrary diagonal element corresponding to a weight in the output layer, we can show (derivation in appendix C) that:

$$\frac{\partial^2 e_i}{\partial w_i^2} = \left[1 - \lambda(1 - \frac{1}{M})\right] h_j^2 \tag{5.13}$$

where h_j^2 is the squared output of the corresponding hidden node. If this element becomes non-positive, the entire matrix is guaranteed to be non-positive definite; therefore we would like the following inequality to hold:

$$\begin{aligned}
0 &< \left[1 - \lambda(1 - \frac{1}{M})\right] h_j^2 \\
0 &< h_j^2 - \lambda h_j^2 (\frac{M-1}{M}) \\
\lambda h_j^2 (\frac{M-1}{M}) &< h_j^2 \\
\lambda &< \frac{h_j^2}{h_j^2(\frac{M-1}{M})} \\
\lambda &< \frac{M}{M-1} \qquad (5.14)
\end{aligned}$$

Since the effect of the hidden node cancels out, we find that this inequality is *independent* of all other network parameters, so it is *a constant for any ensemble architecture using linear output nodes and a simple average combination function.* This defines an upper bound for λ and, since we know the relationship between the two strength parameters, we can also show a bound for γ:

$$\lambda_{upper} = \frac{M}{M-1} \qquad \gamma_{upper} = \frac{M^2}{2(M-1)^2} \tag{5.15}$$

When λ or γ is varied beyond these upper bounds, the Hessian matrix is guaranteed to be non-positive definite. Figure 5.1 plots λ_{upper} and the equivalent γ_{upper} for different numbers of networks. We see that as the number of networks increases, λ_{upper} asymptotes to 1, and γ_{upper} to 0.5. It should be noted that for less than 6 networks or so, λ_{upper} is considerably greater than 1.0, contrary to Liu's original bound.

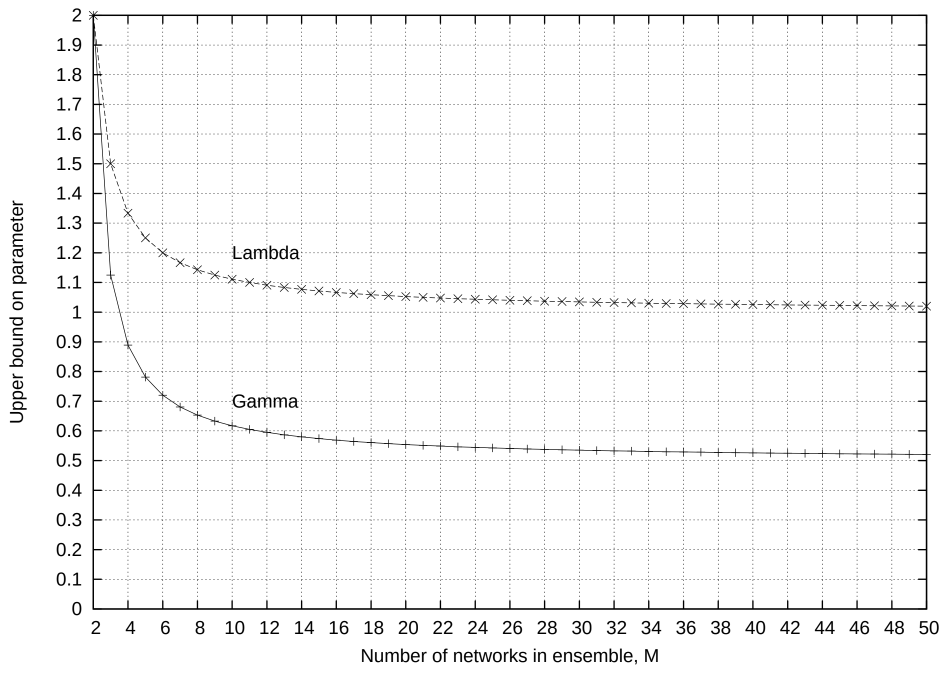

Figure 5.1: The Upper bound on γ and λ

However, it could easily be the case that the leading diagonal is positive, yet the entire matrix is still non-positive definite. This implies that the matrix could become non-positive definite well before our upper bound is reached; the exact situation at which this occurs is as yet unknown. Our bound is therefore a conservative one, and it may be possible to define a tighter bound. The question of whether a tighter bound can be defined can be phrased as *"Are there any general conditions for the off-diagonal elements of the Hessian, that will force non-positive definiteness, in spite of all leading diagonal elements being positive?"*. This question is currently outside the scope of the thesis, but will be discussed as possible future work in the Conclusions chapter.

Strengthening Liu's Result

Liu showed that at $\lambda = 1$, we have the property described in equation (5.4) that the gradient of the individual network error is directly proportional to the ensemble error. However, this property was established under the assumption of constant $\bar{f}$. We can strengthen this result however, by showing that it can still hold if we remove the assumption, but under a different strength parameter value. Examining equations (5.11) and (5.1), we can see that the same proportional relationship will hold when:

$$\gamma = \frac{M}{2(M-1)} \tag{5.16}$$

The implications of this will be further discussed in section 5.4.1.

5.3 A Statistical Interpretation of NC

How can we interpret NC in a statistics framework? In chapter 3 we illustrated the how Ambiguity term is related to the bias-variance-covariance decomposition of the ensemble error. The expected value of the average individual squared error is provides the ensemble bias component, while the expected value of the ambiguity term provides the ensemble variance and covariance components. When training a simple ensemble, we only minimise the errors of the individual networks, and therefore only explicitly influence the bias component in the ensemble. However when training with NC, we use the individual error plus the Ambiguity term as a regularisation factor. The expected value of the Ambiguity term provides the missing second half of the ensemble error that is not included in simple ensemble learning. It therefore can be seen that the reason for the success of NC is that it trains the individual networks with error functions which more closely approximate the individual network's contribution to ensemble error, than that used by simple ensemble learning.

5.4 The Ensemble as a Single Estimator

We have seen that the error of ensemble system can be interpreted in two ways: firstly with the bias-variance decomposition, and secondly with the bias-variance-covariance decomposition. If we regard the ensemble as a single estimator in its own right, we have the bias-variance interpretation; if we regard the ensemble as a combination of other estimators, we have the bias-variance-covariance interpretation. In section 5.4.1 we briefly show how NC-Learning works on a search landscape that, using the λ parameter, can be smoothly scaled between that of a fully parallel ensemble system, and a single large neural network. In section 5.4.2 we show that if we regard the ensemble as a single large neural network, then the error gradient of that network can be broken into four individually understandable components. We then demonstrate that the error gradient of a simple ensemble, and ensemble using NC, can be understood as different combinations of these four components.

5.4.1 Understanding the link to a Single Network

Regard the architecture in figure 5.2. This is an ensemble of three networks, each with

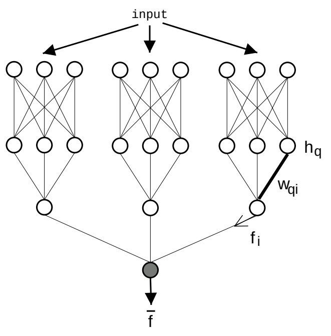

Figure 5.2: A typical ensemble architecture

three hidden nodes, using a simple average combination rule for the ensemble output. We desire to update the weight, w_{qi}, marked in bold—this is one of the output layer weights for the ith network (connected to the qth hidden node). If we now consider the ensemble architecture as one large network (with fixed output layer weights), then our output node is marked in dark gray, and has a linear activation function:

$$\bar{f} = a_i \tag{5.17}$$

and its activation is defined by:

$$a_i = \sum_i \frac{1}{M} f_i \tag{5.18}$$

The error of this large network on a single pattern is:

$$e_{ens} = \frac{1}{2}(\bar{f} - d)^2 \tag{5.19}$$

Now, as before, we find the derivative of e_{ens} with respect to the weight w_{qi}:

$$\frac{\partial e_{ens}}{\partial w_{qi}} = \frac{\partial e_{ens}}{\partial \bar{f}} \frac{\partial \bar{f}}{\partial a_i} \frac{\partial a_i}{\partial f_i} \frac{\partial f_i}{\partial a_i} \frac{\partial a_i}{\partial w_{qi}} \tag{5.20}$$

$$\frac{\partial e_{ens}}{\partial w_{qi}} = \Big[(\bar{f} - d)\Big] \cdot \Big[1\Big] \cdot \Big[\frac{1}{M}\Big] \cdot \Big[f_i(1 - f_i)\Big] \cdot \Big[h_q\Big] \tag{5.21}$$

If we use NC-Learning, we need the derivative of the error e_i with respect to w_{qi}. This is:

$$\frac{\partial e_i}{\partial w_{qi}} = \Big[(f_i - d) - \lambda(\bar{f} - d)\Big] \cdot \Big[f_i(1 - f_i)\Big] \cdot \Big[h_q\Big] \tag{5.22}$$

If $\lambda = 0$, we have:

$$\frac{\partial e_i}{\partial w_{qi}} = \Big[(f_i - d)\Big] \cdot \Big[f_i(1 - f_i)\Big] \cdot \Big[h_q\Big] \tag{5.23}$$

And if $\lambda = 1$, we have:

$$\frac{\partial e_i}{\partial w_{qi}} = \Big[(\bar{f} - d)\Big] \cdot \Big[f_i(1 - f_i)\Big] \cdot \Big[h_q\Big] \tag{5.24}$$

In this latter case, the only difference between (5.21) and (5.24) is the $\frac{1}{M}$. All the minima are in the same locations, but the landscape is M times shallower—the effect of which could be duplicated with a smaller learning rate in the update rule. When we scale λ, we scale smoothly between the gradient of a single network with a linear output function, and that of a parallel ensemble system.

5.4.2 Understanding Components of the Ensemble Error Gradient

We have seen that NC incorporates the ambiguity term into its error function. The actual error function we wish to decrease is the ensemble error:

$$e_{ens} = \frac{1}{2}(\bar{f} - d)^2 = \frac{1}{M} \sum_i [e_i] \tag{5.25}$$

which we choose to achieve by minimising e_i:

$$e_i = \frac{1}{2}(f_i - d)^2 - \frac{1}{2}(f_i - \bar{f})^2 \tag{5.26}$$

However it can be seen that the e_i terms are not independent of one another when varying f_i, simply because they all contain $\bar{f}$. So why does it work? And what is the relationship to a single network method? To understand this we will examine the gradient of the ensemble error function.

We can obtain the gradient by taking the first partial derivative of the functions with respect to f_i. It should be noted that we are assuming an ensemble of networks with linear output functions. If this is the case, the error gradient we would like to minimise is: $\frac{\partial e_{ens}}{\partial w_i} = \frac{\partial e_{ens}}{\partial f_i}\frac{\partial f_i}{\partial w_i}$. The second term $\frac{\partial f_i}{\partial w_i}$ evaluates to simply the output of the relevant hidden node (which we assume to be sigmoid activation). The error gradient is therefore proportional to $\frac{\partial e_{ens}}{\partial f_i}$, and we can simplify our calculations below by omitting the reference to the hidden node since it just acts as a scaling component.

We can differentiate the ensemble error in two ways. Firstly from the composite:

$$\begin{aligned}
e_{ens} &= \frac{1}{2}(\bar{f} - d)^2 \\
\frac{\partial e_{ens}}{\partial f_i} &= \frac{1}{M}(\bar{f} - d)
\end{aligned}$$

in one simple step using the chain rule. Or we can do it from the decomposed form:

$$e_{ens} = \frac{1}{M}\sum_i \left[\frac{1}{2}(f_i - d)^2 - \frac{1}{2}(f_i - \bar{f})^2\right]$$

Before we begin the calculation of the derivative, we first break this into two components, where the first term concerns network i, and the second concerns all the other networks, $j \neq i$:

$$e_{ens} = \frac{1}{M}\left[\frac{1}{2}(f_i - d)^2 - \frac{1}{2}(f_i - \bar{f})^2\right] + \frac{1}{M}\sum_{j\neq i}\left[\frac{1}{2}(f_j - d)^2 - \frac{1}{2}(f_j - \bar{f})^2\right]$$

If this is the case, we can see:

$$\begin{aligned}
e_{ens} = &\frac{1}{M}\left[\frac{1}{2}(f_i - d)^2 - \frac{1}{2}(f_i - \bar{f})^2\right] \\
&+\frac{1}{M}\left[\frac{1}{2}(f_j - d)^2 - \frac{1}{2}(f_j - \bar{f})^2\right] \\
&+\frac{1}{M}\left[\frac{1}{2}(f_k - d)^2 - \frac{1}{2}(f_k - \bar{f})^2\right]
\end{aligned}$$

the partial derivative of this with respect to f_i is:

$$\begin{aligned}
\frac{\partial e_{ens}}{\partial f_i} &= \frac{1}{M}\left[(f_i - d) - (1 - \frac{1}{M})(f_i - \bar{f})\right] \\
&+ \frac{1}{M}\left[\quad 0 \quad - (0 - \frac{1}{M})(f_j - \bar{f})\right] \\
&+ \frac{1}{M}\left[\quad 0 \quad - (0 - \frac{1}{M})(f_k - \bar{f})\right]
\end{aligned}$$

which is equal to:

$$\begin{aligned}\frac{\partial e_{ens}}{\partial f_i} &= \frac{1}{M}\left[(f_i - d) - (1 - \frac{1}{M})(f_i - \bar{f})\right] \\ &+ \frac{1}{M}\left[\frac{1}{M}(f_j - \bar{f})\right] \\ &+ \frac{1}{M}\left[\frac{1}{M}(f_k - \bar{f})\right]\end{aligned}$$

which is in turn equal to:

$$\frac{\partial e_{ens}}{\partial f_i} = \frac{1}{M}\left[(f_i - d) - (1 - \frac{1}{M})(f_i - \bar{f})\right] + \frac{1}{M}\sum_{j\neq i}\left[\frac{1}{M}(f_j - \bar{f})\right] \tag{5.27}$$

if we multiply out the brackets in the first term:

$$\frac{\partial e_{ens}}{\partial f_i} = \frac{1}{M}\left[(f_i - d) - (f_i - \bar{f}) + \frac{1}{M}(f_i - \bar{f})\right] + \frac{1}{M}\sum_{j\neq i}\left[\frac{1}{M}(f_j - \bar{f})\right]$$

Finally, multiplying through with the $\frac{1}{M}$, we can see the error of the ensemble has four components, shown and described in table 5.1.

Component	*Interpretation*
$\frac{1}{M}(f_i - d)$	This is the component of the error gradient due to the difference between the ith network output and the desired output (due to the fact that f_i is changing.)
$-\frac{1}{M}(f_i - \bar{f})$	This is the component of the error gradient due to the difference between f_i and $\bar{f}$ (due to the fact that f_i is changing.)
$\frac{1}{M^2}(f_i - \bar{f})$	This is the component of the error gradient due to the difference between f_i and $\bar{f}$ (due to the fact that $\bar{f}$ changes, because f_i is changing.)
$\frac{1}{M^2}\sum_{j\neq i}(f_j - \bar{f})$	This is the component of the error gradient due to the differences between the f_js and $\bar{f}$ (due to the fact that $\bar{f}$ changes, because f_i is changing.)

Table 5.1: Ensemble gradient components

Each of these components contributes to the error of the ensemble estimator $\bar{f}$, as defined in equation 5.25. If we take the $\frac{1}{M}$ on the outside and label the components, we can make an interesting observation.

$$\frac{\partial e_{ens}}{\partial f_i} = \frac{1}{M}\left[\underbrace{(f_i - d)}_{A} - \underbrace{(f_i - \bar{f})}_{B} + \underbrace{\frac{1}{M}(f_i - \bar{f})}_{C} + \underbrace{\frac{1}{M}\sum_{j\neq i}(f_j - \bar{f})}_{D}\right]$$

When using a simple ensemble, the error gradient for a single network f_i is:

$$\frac{\partial}{\partial f_i}\left[\frac{1}{2}(f_i - d)^2\right] = (f_i - d) \tag{5.28}$$

Looking at equation 5.4.2, we can see this is component A. The error gradient for f_i in an ensemble using NC is:

$$\begin{aligned}\frac{\partial}{\partial f_i}\left[\frac{1}{2}(f_i - d)^2 - \gamma(f_i - d)^2\right] &= (f_i - d) - 2\gamma(1 - \frac{1}{M})(f_i - \bar{f}) \\ &= (f_i - d) - 2\gamma\left[(f_i - d) - \frac{1}{M}(f_i - \bar{f})\right] \\ &= A - 2\gamma(B - C)\end{aligned} \tag{5.29}$$

alternatively, because we know $\lambda = 2\gamma(1 - \frac{1}{M})$, this can also be expressed as $A - \lambda B$.

From all this we can understand the relationships between minimising the simple ensemble error, the NC ensemble error, and a single network, described in table 5.2.

Algorithm	*Components in error gradient for network i*
Simple Ensemble	A
Ensemble with NC	$A - 2\gamma(B - C)$ **or** $(A - \lambda B)$
Ensemble with NC, $\lambda = 1$ or equivalently $\gamma = \frac{M}{2(M-1)}$	$A - B$
Single large network (with fixed output layer weights)	$\frac{1}{M}(A - B)$

Table 5.2: Components

As an illustration of the way these components interact when we use NC, consider the scenario in figure 5.3.

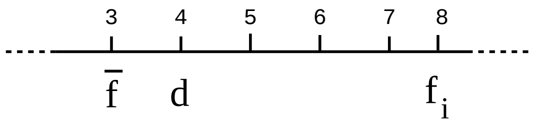

Figure 5.3: Number line

On a single datapoint, the network f_i is estimating too high at 8.0, which is right of the target $d = 4$. We have an ensemble of $M = 5$ networks, but for clarity the outputs of the other ensemble members are not shown; the resulting ensemble output is $\bar{f} = 3$, too low, left of the target. When updating the value of f_i, a simple ensemble will use the gradient measurement $(f_i - d) = 4$, resulting in f_i being shifted left, towards the target. However, this will cause the ensemble output $\bar{f}$ to shift *left*, moving *away from the target.* An ensemble using NC will include three gradient components:

$$\begin{aligned} A - 2\gamma(B - C) &= (f_i - d) - 2\gamma\Big[(f_i - \bar{f}) - \frac{1}{M}(f_i - \bar{f})\Big] \qquad (5.30) \\ &= 4 - 2\gamma(5 - \frac{1}{5}5) \\ &= 4 - \gamma 8 \end{aligned}$$

If we choose $\gamma = 0.3$, this sum evaluates to 1.6, still a positive gradient for f_i, meaning the ensemble output will still be moved away from the target. If however we choose $\gamma = 0.7$, it evaluates to -1.6, giving a pressure for the network f_i to move *away* from the target, causing the ensemble output to move *closer* to the target. The setting of the γ value provides a way of finding a trade-off between these gradient components that will cause the ensemble output $\bar{f}$ to move toward the target value d.

5.4.3 Understanding a 2-Network Search Landscape

Figure 5.6 shows the error functions for a 2-network ensemble as we vary the strength parameter. This shows that larger strength parameters force a wider basin of attraction; with $\gamma = 0$, there is only one minimum, when both networks are at the target; when $\gamma = 0.5$, the basin is infinitely wide and the networks can settle anywhere along the ravine created and still fit the datapoint. Although we cannot visualise higher dimensions than this, it implies that NC functions by widening the basins of attraction, allowing flexibility for other datapoints.

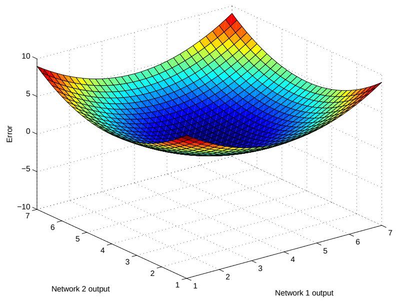

Figure 5.4: The joint error function for a two-network ensemble, $\lambda = 0$

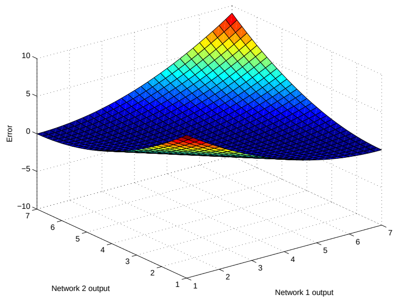

Figure 5.5: The joint error function for a two-network ensemble, $\lambda = 0.5$

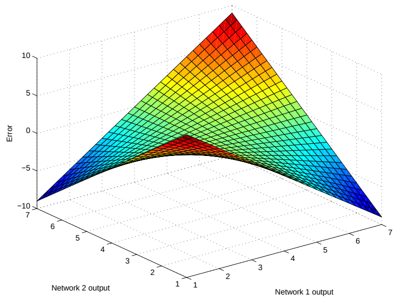

Figure 5.6: The joint error function for a two-network ensemble, $\lambda = 1$

5.5 An Empirical Study

5.5.1 The Bounds of the λ and γ Parameters

With our new understanding of the grounding behind NC, we now perform an empirical evaluation, and show that it is critical to consider values for the strength parameter *outside* the originally specified range.

Table 5.3 shows the classification error rates of two empirical tests, on the Wisconsin breast cancer dataset from the UCI repository (699 patterns), and the Heart disease dataset from Statlog (270 patterns). An ensemble consisting of two networks, each with five hidden nodes, was trained using NC. We use 5-fold cross-validation, and 40 trials from uniform random weights in $[-0.5, 0.5]$ for each setup; in total 200 trials were conducted for each experimental configuration. It should be noted that with 2 networks, $\gamma = \lambda$. The λ values tested are those considered in the original work on NC: 0.0, 0.5 and 1.0. When λ was set appropriately, results on the heart data showed NC significantly better than a simple ensemble (equivalent to $\lambda = 0$) at $\alpha = 0.05$ on a two-tailed t-test. On the breast cancer data, although the mean was lower, it was not statistically significant.

Table 5.3: Mean classification error rates (200 trials) using NC on two UCI datasets

	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 1$
Breast cancer	0.0408	0.0410	0.0383
Heart disease	0.2022	0.1995	**0.1802**

Figures 5.7 and 5.8 show the results of repeating our experiments, but illustrating the full range of the strength parameter. Mean error rate over the 200 trials is plotted, and 95% confidence intervals shown. We see that performance on the breast cancer data *can be improved significantly* by considering the upper bounds beyond those previously specified; on the heart disease data, stable performance is observed beyond $\lambda = 1$.

As a further measure of comparison, we calculated the percentage reduction in the mean error rate, in relation to when $\lambda = 0$. On the breast cancer data, using $\lambda = 1$ gave a 1% reduction, but using the optimum value at $\lambda = 1.7$ gave a 3% reduction.

We have shown a significant performance improvement by reconsidering the bounds of the strength parameters. It should be noted that, even though the theoretical upper bound is known, in practise it seems error can rise rapidly long before this bound is reached. On the breast cancer data, error became uncontrollable beyond $\lambda = 1.8$, and on the heart disease data at $\lambda = 1.45$; it remains to be seen if it is possible to empirically characterise when this rapid increase will occur.

5.5.2 Single Estimator or Ensemble?

We have seen how NC Learning can be compared to a single estimator. When $\lambda = 1$ the only difference between an ensemble and a single large neural network is that the ensemble combination weights are not updated—they are fixed at $\frac{1}{M}$. For completeness of this investigation, in this section we train an ensemble with NC ($\lambda = 1$) and allow the combination

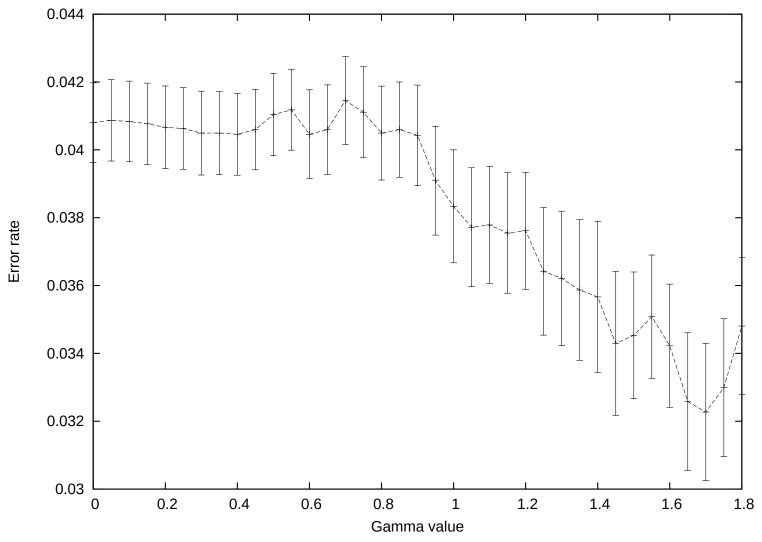

Figure 5.7: Breast cancer dataset, 2 networks, 5 hidden nodes each

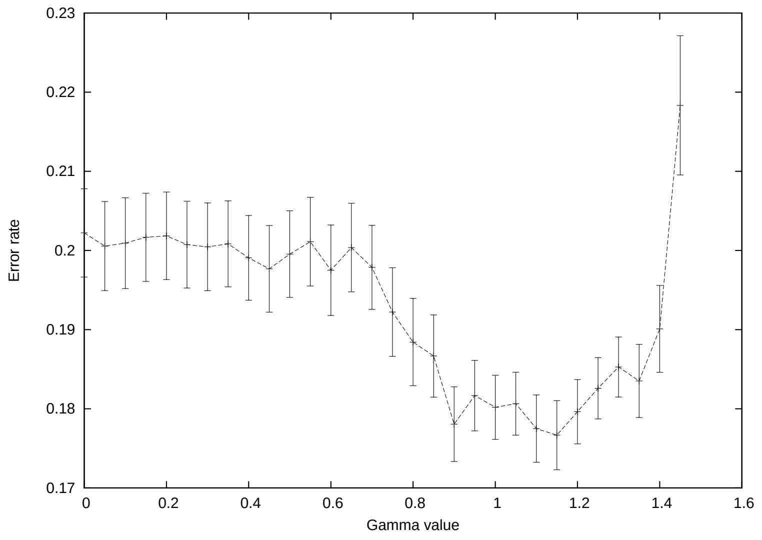

Figure 5.8: Heart disease dataset, 2 networks, 5 hidden nodes each

weights to be updated also—effectively learning with a single non-fully connected neural network, with a linear output node. The update equation for the combination weight w_i is:

$$\Delta w_i = -\alpha(\sum_j (w_j f_j) - d) f_i \tag{5.31}$$

Figures 5.9 to 5.11 show error rates on the Breast cancer dataset. We used 5-fold cross validation, and 20 trials of random weights for each fold, training for 2000 iterations. The ensembles were trained using NC, with $\lambda = 1$. For the results labelled "Monolithic", the weights in the output layer of the ensemble were allowed to be updated as well, making the estimator one large neural network. The results labelled "Non-monolithic" are standard

NC learning. Figure 5.9 shows results for ensembles using networks with 2 hidden nodes each, figure 5.10 shows results with 5 hidden nodes per network, and figure 5.11 shows the same for 10 hidden nodes.

All results indicate that the modification to train the ensemble as a single estimator makes no significant difference to error rates; as we know from experiments in the previous section, a λ value other than 1.0 can make significant improvements to error rates. We noted at the end of section 5.4.1 that NC allows a smooth transition of the error gradients between that of a fully parallel ensemble system and a single estimator. Based on this and our empirical observations in this section, we conclude that NC succeeds because it allows us to choose the optimum blend between training an ensemble and training a single estimator.

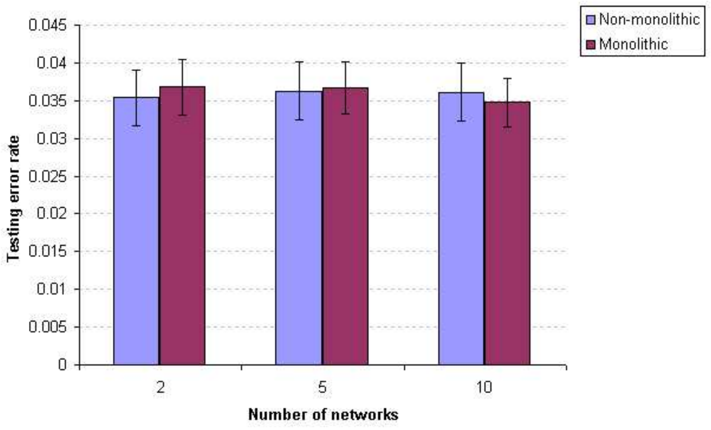

Figure 5.9: Error rates for ensembles of networks with 2 hidden nodes, using NC ($\lambda = 1$)

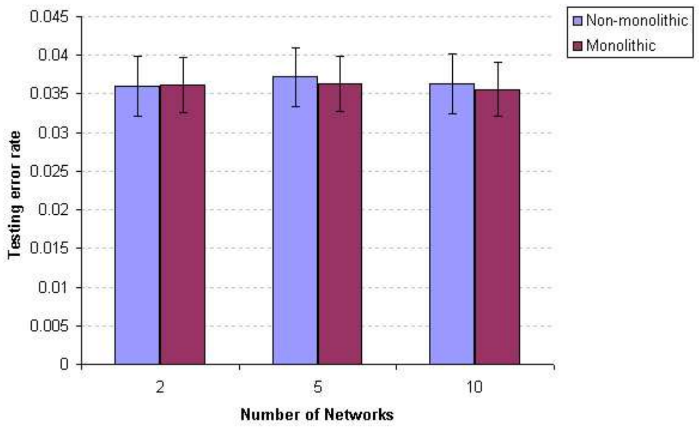

Figure 5.10: Error rates for ensembles of networks with 5 hidden nodes, using NC ($\lambda = 1$)

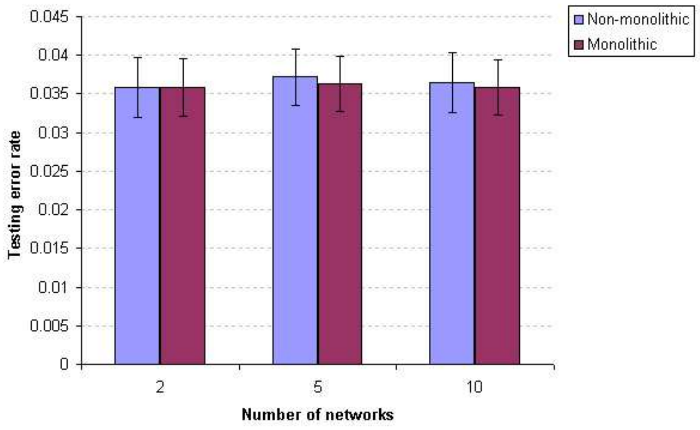

Figure 5.11: Error rates for ensembles of networks with 10 hidden nodes, using NC ($\lambda = 1$)

5.6 Chapter Summary

In this chapter we provided a thorough critique of Negative Correlation (NC) Learning [85], a technique that extended from [117], and developed in the evolutionary computation literature. We showed a link to the Ambiguity decomposition [70], and explained the success of NC in terms of the bias-variance-covariance decomposition [141]. This formalisation allowed us to define bounds on the parameters and understand how it relates to other algorithms. When taking into account our new understanding of the parameters, significant reductions in error were observed in empirical tests.

We then engaged in a detailed study of the error gradient and how it changes when using NC learning. We showed that the error of an NC learning ensemble can be broken down into four components, each with its own interpretation with respect to the current state of the ensemble. Further to this we noted at the end of section 5.4.1 that NC allows a smooth transition of the error gradients between that of a fully parallel ensemble system and a single estimator. Based on this and our empirical observations in this section, we conclude that NC succeeds because it allows us to choose the optimum blend between training an ensemble and training a single estimator.

This theoretical chapter has served to illustrate how NC is not merely a heuristic technique, but *fundamentally* tied to the dynamics of training an ensemble system with the mean squared error function. The observations we made are in fact all *properties of the mean squared error function.* NC is therefore best viewed as a *framework* rather than an algorithm itself. In summary the observations we have made in this chapter are not specific to any type of predictor, but general to any predictor that can minimise a quadratic loss function.

We know from figure 5.1 that the upper bound reduces as we add more networks; from this it is reasonable to infer that the *optimal* value may follow a similar trend. In the following chapter we gather empirical evidence to support some of the theoretical claims made in this chapter, including an analysis of how the optimal strength parameter γ_* changes as we alter the ensemble architecture and dataset.

Chapter 6

Empirical Analysis of Negative Correlation Learning

In this chapter we perform a number of experiments, to determine how the NC algorithm behaves in different circumstances. Our question to be answered is "can the optimal value of γ be characterised in any way?". To answer this we observe the behaviour as we vary (1)the dataset used, (2)the number of networks in the ensemble, (3)the number of hidden nodes per network,and (4)the amount of noise in the training data. This chapter also serves to support the theoretical findings in chapter 5 regarding the bounds of the strength parameter.

6.1 Datasets

In order to gain a fair appreciation of algorithm performance, we employ a number of different datasets to test on. We use three regression problems and three classification problems, including the Boston and Phoneme datasets as described in section 4.1. We use sigmoid activation functions for the hidden units of each network. For the regression datasets, we use networks with a single output node with a linear activation function; for the classification datasets, the single output node is sigmoid activation. In all cases, we use the simple average combination rule for the ensemble, which for the classification problems is thresholded at 0.5 to provide a binary value.

6.1.1 Classification Problems

Skin

This is a dataset produced by the author and colleagues; the problem is to predict whether a particular pixel in a real-world image is human skin or not. The data was generated by asking volunteers to select pixels corresponding to skin and not skin, from a variety of real-world images, and recording the image information at those pixels. The data has 4500 examples (each corresponding to one pixel in an image), with 6 continuous valued inputs and 1 binary output. For a given pixel, the first three input variables are the red, green and blue values at that point, rescaled to $[0, 1]$. The last three input variables were generated by calculating the sample variance of a 3x3, 5x5 and 7x7 window around the pixel. Networks were trained for 500 iterations. We divided the dataset into 5 equal sized portions; we trained on one fifth of the data (900 examples), then tested on the remaining four fifths (3600 examples). This was repeated 5 times so each portion was used in turn as training data. Each time, the ensemble was evaluated from 40 trials of random weights, giving a total of 200 trials for each experiment.

Breast cancer

This is the Wisconsin Breast Cancer data from the UCI Machine Learning Depository [11]; the problem is to distinguish malignant (cancerous) from benign (non-cancerous) examples. The data has 699 examples, 458 benign and 241 malignant. Each example has 9 integer inputs and 1 binary output. All input values were linearly rescaled to the range $[0, 1]$. Networks were trained for 2000 iterations, following Yao and Liu [154]. We used 5-fold cross-validation, with each fold being evaluated from 40 trials of random weights, giving a total of 200 trials for each experiment.

6.1.2 Regression Problems

Jacobs' Dataset

This data, used in [56] and [85], was generated by the function:

$$h(\mathbf{x}) = \frac{1}{13}\left[10sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5\right] - 1 \tag{6.1}$$

where $\mathbf{x} = [x_1, .., x_5]$ is an input vector whose components are drawn uniformly at random from $[0, 1]$. The output of the function, $h(x)$, is in the range $[-1, 1]$. We used a training set of size 500, and a testing set of size 1000. For the training data only, Gaussian random noise with zero mean and $\sigma^2 = 0.2$ was added to the outputs. This dataset is therefore referred to as Jacobs(0.2). We generated 5 separate training sets, but only 1 testing set. Networks were trained for 500 iterations on a training set and evaluated on the testing set, each time from 40 trials of random weights, giving a total of 200 trials for each experiment.

Friedman#1

This data, first used in [59] was generated by the function:

$$h(\mathbf{x}) = 10sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5 \tag{6.2}$$

where $\mathbf{x} = [x_1, .., x_{10}]$ is an input vector whose components are drawn uniformly at random from $[0, 1]$. Totally there are 10 continuous valued attributes, but only the first 5 are used in the function, leaving the last 5 as irrelevant characteristics that the algorithm has to learn to ignore. It can be seen that Jacobs' Dataset, described above, is a shifted and rescaled version of this function. However, since Friedman has a larger range, a very high degree of noise and several irrelevant attributes, it is a notably different problem. We used a training set of size 200, and a testing set of size 1000. For the training data only, Gaussian random noise with zero mean and $\sigma^2 = 1.0$ was added to the outputs. We generated 5 separate training sets, but only 1 testing set. Networks were trained for 1000 iterations on a training set and evaluated on the testing set, each time from 40 trials of random weights, giving a total of 200 trials for each experiment.

6.2 How is γ_* affected by the Number of Networks?

We identified in chapter 3 that the number of networks, M, in the ensemble plays a key role in determining the performance of NC; this M term is an intrinsic part of the theoretical upper bound (see section 5.2.2). In this section we would like to understand how M affects performance, with the goal of characterising where the optimal value of γ occurs in different circumstances.

Figures 6.1 to 6.6 show how error changes with γ and M. For clarity of presentation, in all cases, the γ parameter is plotted for the range 0 to 1, but it should be remembered as demonstrated in the previous chapter, the useful range can sometimes be greater than 1. Experiments are performed according to the methodology described in section 6.1. We used networks with 6 hidden nodes throughout, and plotted the testing error for ensembles of 1, 2, 4, 8, 12 and 16 networks. It should be noted that for the regression problems we plotted the mean-squared error, whereas for the classification problems we have a different statistic of interest—the classification error rate (percentage of misclassified examples). This point is important as it has consequences for the theoretical upper bound we introduced in the last chapter.

Through figures 6.1 to 6.6, we can see an easily identifiable trend, present in five out of the six datasets used: increasing γ causes a virtually monotone decrease in error, up to a particular "threshold" point, beyond which a rapid rise in error is observed; on closer examination it was revealed that at this point the network weights had diverged to excessively large values. Figure 6.2 stands out as the only dataset not to exhibit this neat trend. We repeated this experiment on the Jacobs dataset with simpler networks, each with 2 hidden nodes; figure 6.7 shows this result, and demonstrates the trend we expected, of a steady decrease followed by a rapid increase. It appears that when the networks are relatively complex for the task[1], the behaviour of the γ parameter becomes unpredictable, and no significant gain can be seen for using NC. However, as can be seen by comparing figure 6.2 with figure 6.7, using the simpler networks in conjunction with NC allows us to equal the performance of the more complex networks; at γ_* there is no statistically significant difference (two tailed t-test, $\alpha = 0.05$) between using 2 and 6 hidden nodes, for an ensemble size of 4 or more. This behaviour is further examined in section 6.4 when we see how γ_* changes with noisy data.

It is interesting to note that γ_* occurs for all datasets just *before* this rapid increase. Figure 6.8 plots γ_* for ensembles of networks with 6 hidden nodes as we increase M, the number of networks. We have also superimposed the theoretical upper bound calculated in the previous chapter as a bold dashed line. An obvious discrepancy can be seen in this graph, concerning the Breast Cancer dataset.

[1]It appears 6 hidden nodes is more than enough to solve the Jacobs dataset.

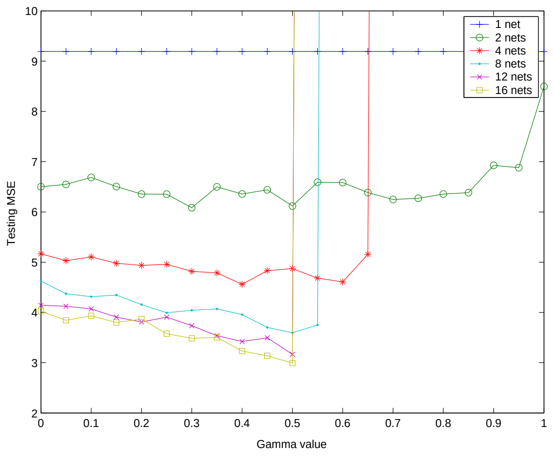

Figure 6.1: Friedman, 6 hidden nodes

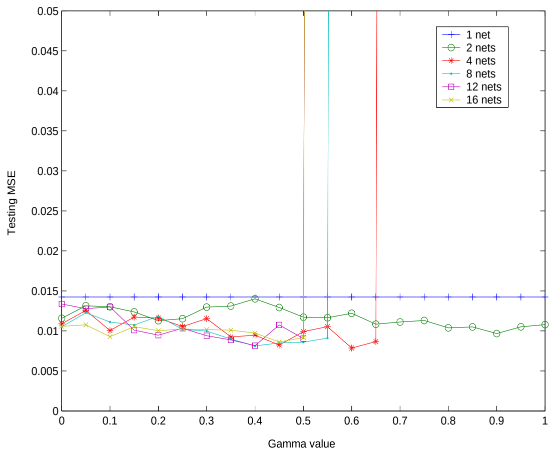

Figure 6.2: Jacobs(0.2), 6 hidden nodes

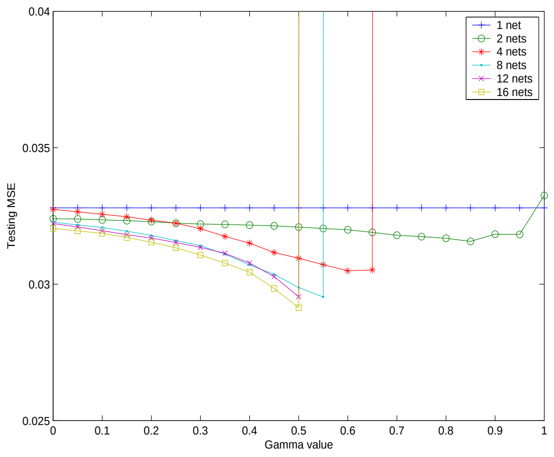

Figure 6.3: Boston, 6 hidden nodes

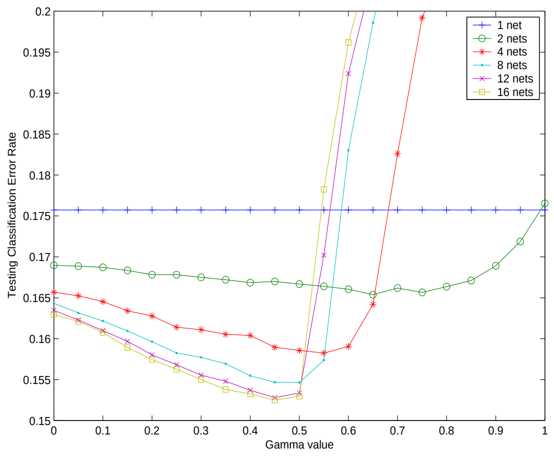

Figure 6.4: Phoneme, 6 hidden nodes

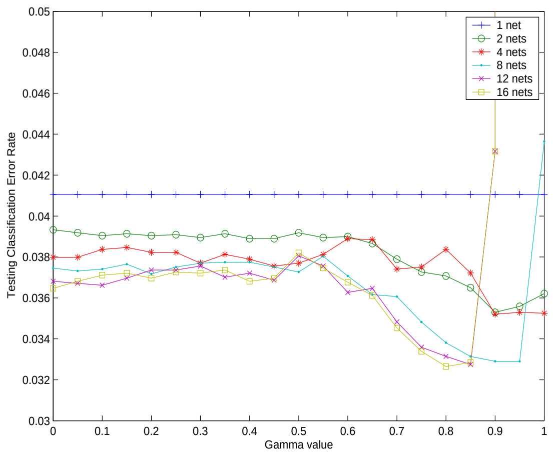

Figure 6.5: Breast Cancer, 6 hidden nodes

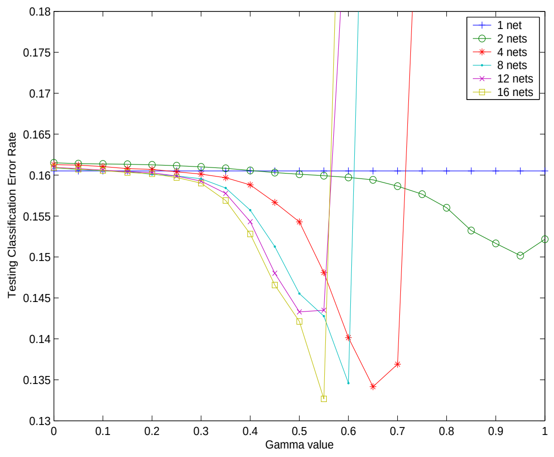

Figure 6.6: Skin, 6 hidden nodes

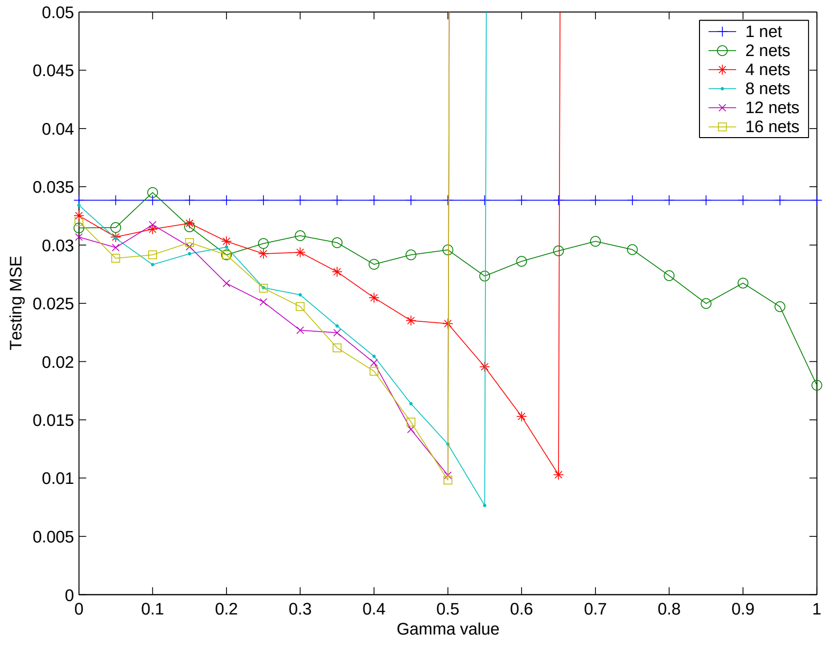

Figure 6.7: Jacobs(0.2) Dataset: MSE vs γ, ensembles of networks with 2 hidden units

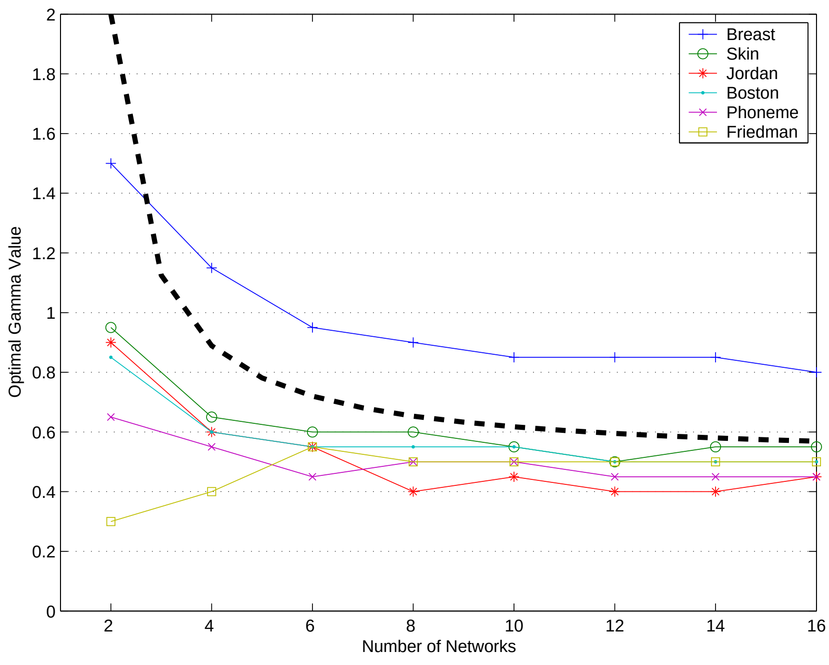

Figure 6.8: γ_* for ensembles of networks with 6 hidden nodes (also shows theoretical γ_{upper})

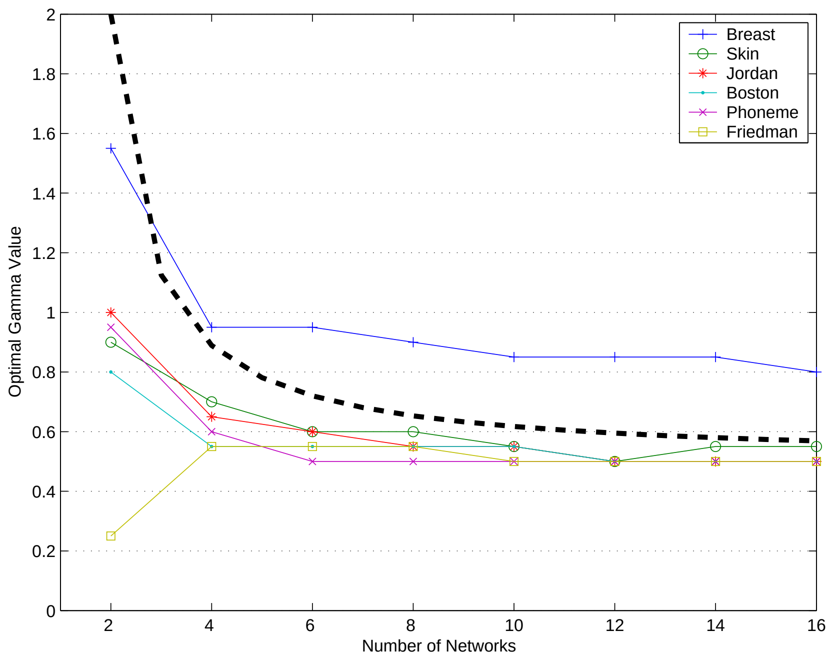

Figure 6.9: γ_* for ensembles of networks with 2 hidden nodes (also shows theoretical γ_{upper})

It seems from figure 6.8, that the optimal values for $M \geq 4$ on the Breast Cancer dataset occur *above* the theoretical upper bound. On closer examination, it was found that the error at these supposedly optimal γ values was not statistically significantly lower than at $\gamma = 0.0$. In spite of these significance results, this serves to highlight a limitation of the upper bound we previously defined. The bound was defined by calculating second derivatives of the MSE error function, and seeing when the Hessian matrix was forced to non-positive definiteness. The loss function of interest in classification problems is *zero-one loss*, and as such it is non-differentiable and our calculations with the second derivative *do not apply*. If however we plot the *MSE* for this dataset, shown in figure 6.10, we see that it still behaves according to our upper bound. In figure 6.11, we show γ_* approaching 0.5 (and equivalently λ_* goes to 1.0) with increasing M.

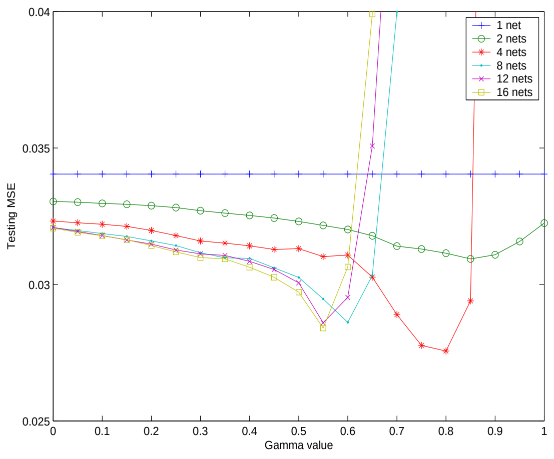

Figure 6.10: Breast Cancer: MSE vs γ, ensembles of networks with 6 hidden units.

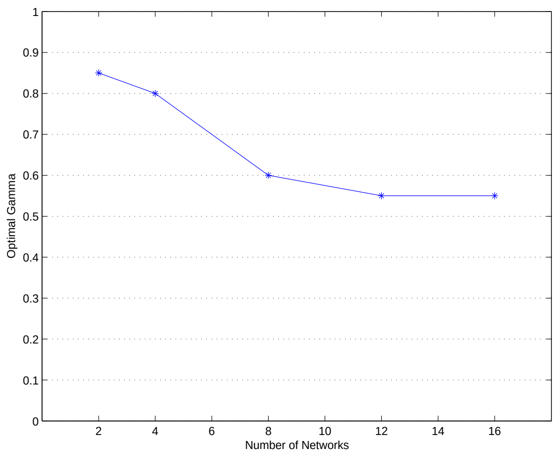

Figure 6.11: Optimal γ with respect to MSE; illustrating that although error rate does not follow the theoretical upper bound, the MSE does.

Our theory work (section 5.2.2) shows that the upper bound will go to 0.5, and as we discussed in the summary of chapter 5, it is reasonable to hypothesize that the optimum value, γ_*, may well also follow the same trend. The trend in figure 6.8 seems mostly to support this hypothesis, except that in some situations the Phoneme, Jacobs and Friedman datasets show $\gamma_* < 0.5$. On closer examination it was found that on these datasets, when γ_* was found to be less than 0.5, the error was not statistically significantly lower (two-tailed t-test, $\alpha = 0.05$) than at $\gamma = 0.5$. Figure 6.9 plots the same information but for networks each with 2 hidden nodes, which shows no optimal values below 0.5. We conclude therefore that significant improvements are likely to occur with relatively simple networks at $\gamma = 0.5$; but, with more complex networks, the individuals are powerful enough to solve the problem configured as a simple ensemble, and NC can show no viable benefits when used. Therefore, this graph agrees with the hypothesis we proposed at the end of chapter 5, that since the upper bound decreases toward $\gamma = 0.5$, so should the optimal value.

But *why* does γ_* approach 0.5 as we add networks? To answer this question we refer back to the decomposition of the ensemble error into a weighted summation of components:

$$\frac{1}{2}(\bar{f}-d)^2 = \frac{1}{M}\sum_i \left[\frac{1}{2}(f_i-d)^2 - \frac{1}{2}(f_i-\bar{f})^2\right] \tag{6.3}$$

The ith component, corresponding to the error contribution of the ith network, is:

$$\frac{1}{2}(f_i-d)^2 - \frac{1}{2}(f_i-\bar{f})^2 \tag{6.4}$$

If we use a simple ensemble, we minimize $\frac{1}{2}(f_i - d)^2$ for each network i. Doing so however makes the assumption that the error contribution of each network is independent of the others; this is clearly false. Since changing f_i will cause $\bar{f}$ to also change, and $\bar{f}$ is present in the error contribution of all networks, then changing the output of any network i will have consequences for the other network errors. If we have a larger ensemble though, the influence of f_i on $\bar{f}$ becomes less and less, effectively making $\bar{f}$ independent. At this point we can minimize each error contribution as though they are independent of one another—so the optimum error function is simply the contribution itself, as in equation (6.4), which is equal to the NC error function with $\gamma = 0.5$.

How much gain can NC with γ_* provide?

Figures 6.12 to 6.17 illustrate the gain that can be achieved at γ_* compared to simple ensemble learning ($\gamma = 0.0$). The Friedman, Boston and Phoneme datasets (figures 6.12, 6.14 and 6.15) all show a general trend that NC shows more benefits when used on larger ensembles. The Jacobs dataset (figure 6.13) shows a trend of increasing gain with larger M, up to a point, followed by a decrease in gain, indicating that the larger ensembles using NC (from 8 networks onwards) are overfitting the data. The Skin dataset (figure 6.17) shows larger gains for $M \geq 4$ than at $M = 2$; the largest gain is shown at $M = 16$. It seems in general, the benefits of NC are shown best with a large number of simple networks, except where the ensembles begin to overfit the data; this hypothesis is also supported by similar results in our previous experiments [18]. A notable exception to this is the Breast Cancer dataset, to which we will now devote special attention.

Results for the Breast Cancer dataset, in figure 6.16, shows a very large drop in error for the $M = 2$ network ensemble, followed by an increase for $M \geq 4$, and then no further significant gains with larger ensembles. This error rate for $M = 2$ is not equalled by any ensemble of *any* structure, including larger networks and larger ensembles. How can this be? To answer this question we used ROC (*Receiver Operator Characteristics*) Analysis [158]. This type of analysis is widely used in medical diagnosis, and is based on the fact that certain information is hidden by assessing our estimator with a single measure (the error rate). It provides us with two measures: *sensitivity* and *specificity*. Sensitivity can be understood as the likelihood of correctly identifying a cancerous case when presented with one. Specificity is the likelihood of correctly identifying a non-cancerous case when presented with one. For a single pattern, we have a target value and a corresponding response from the predictor; in our case the predictor is *the ensemble*. Our target value is either class 'positive' or class 'negative'. Dependent on these values, the predictor's response to each pattern is labelled as one of four possible outcomes:

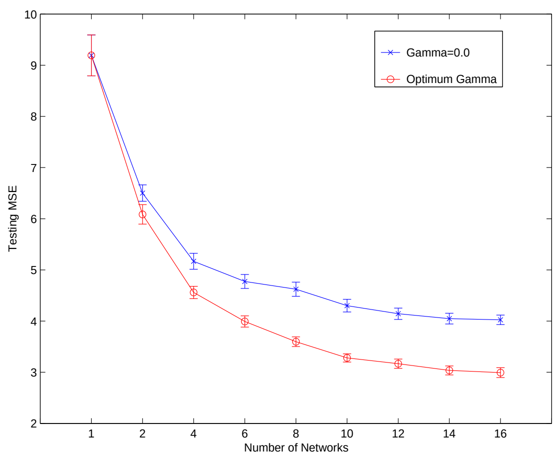

Figure 6.12: Friedman, 6 hidden nodes

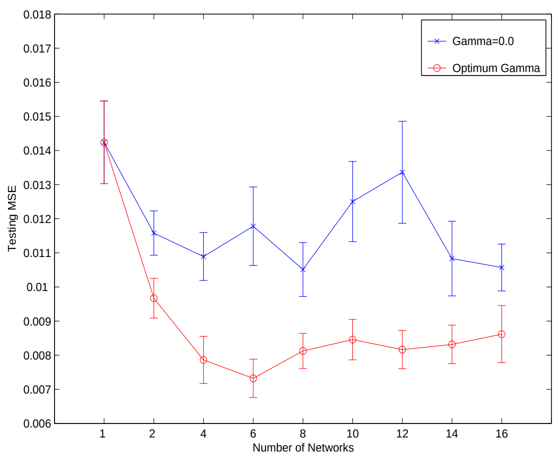

Figure 6.13: Jacobs(0.2), 6 hidden nodes

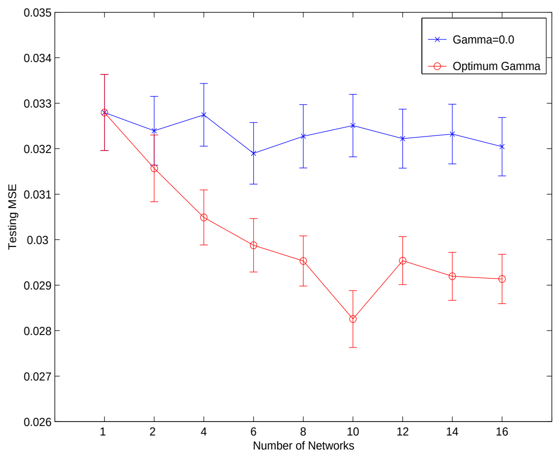

Figure 6.14: Boston, 6 hidden nodes

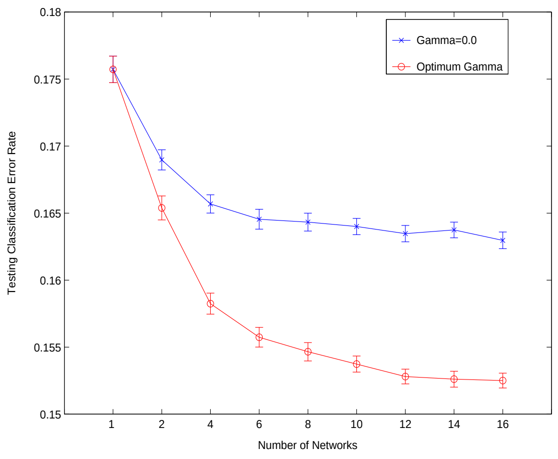

Figure 6.15: Phoneme, 6 hidden nodes

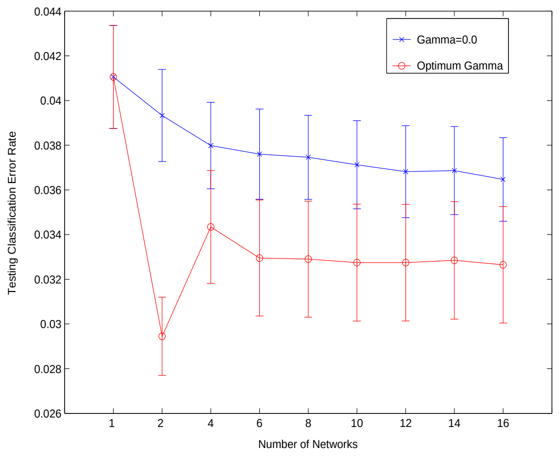

Figure 6.16: Breast Cancer, 6 hidden nodes

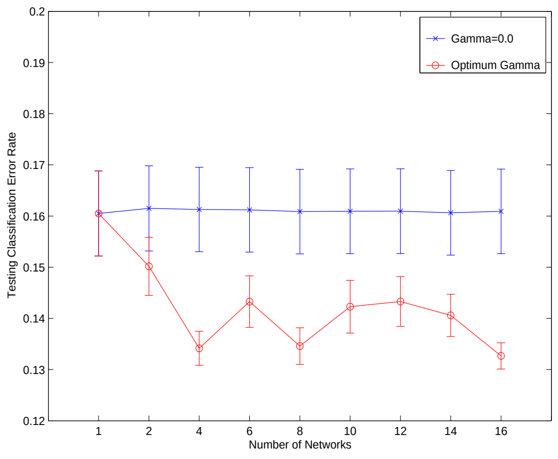

Figure 6.17: Skin, 6 hidden nodes

True Positive The predictor said *'positive'*, and was *correct.*

True Negative The predictor said *'negative'*, and was *correct.*

False Positive The predictor said *'positive'*, and was *incorrect.*

False Negative The predictor said *'negative'*, and was *incorrect.*

Now sum over the testing set the number of occurrences of each of these. The number of true positives is TP, the number of false negatives is FP, and the others are similarly defined. The formulae defining our statistics for ROC analysis are:

$$Sensitivity = \frac{TP}{TP + FN} \qquad Specificity = \frac{TN}{TN + FP}$$

Figures 6.18 and 6.19 plot sensitivity and specificity for ensembles on the Breast Cancer dataset, as we change the γ strength parameter.

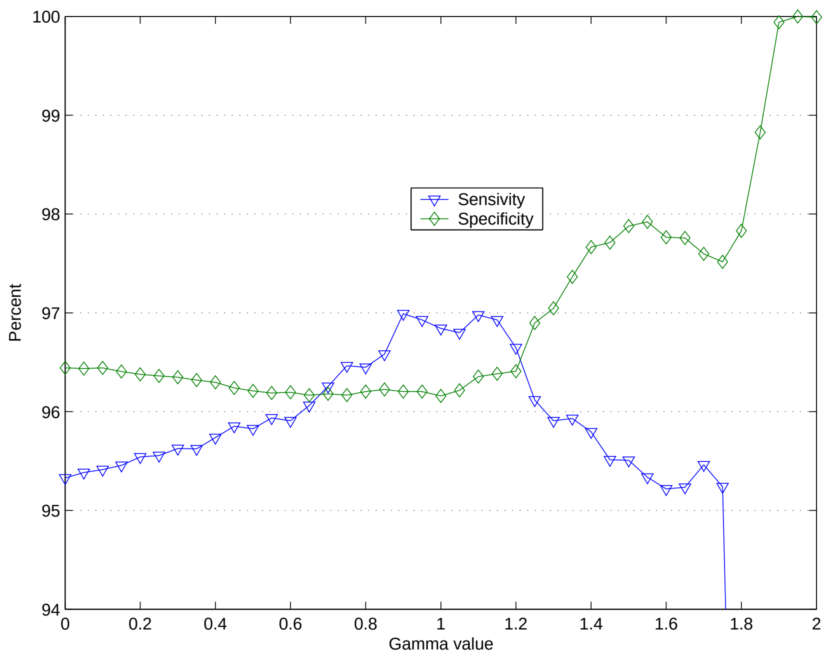

Figure 6.18: ROC analysis on Breast cancer dataset: ensemble of 2 networks, optimum γ minimising the error rate is at $\gamma = 1.5$

The optimum γ value (in the sense of reducing error rate) for 2 networks was $\gamma_* = 1.5$, and for 6 networks was $\gamma_* = 0.95$. For 2 networks, the error rate was very low at 0.029, but at $\gamma_* = 1.5$ in figure 6.18 we see a *large drop in sensitivity* in comparison to other γ values—a sensitivity of 95.5% and specificity of 97.9%. For 6 networks, the error rate is higher at 0.033, however we see in figure 6.19 that sensitivity is kept relatively high at 97.2%, and specificity at 96.5%. The high specificity of the two-network ensemble counted for more in reducing the error rate, since there were almost twice as many non-cancerous patients (458) as there were cancerous patients (241).

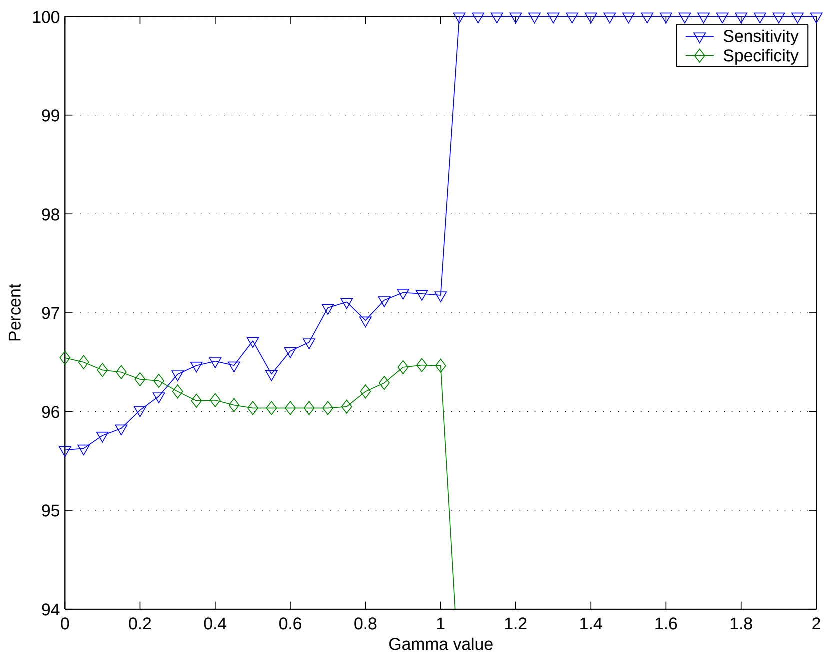

Figure 6.19: ROC analysis on Breast cancer dataset: ensemble of 6 networks, optimum γ minimising the error rate is at $\gamma = 0.9$

In conclusion to this question, although the two-network ensemble achieved significantly higher accuracy, it did so by sacrificing sensitivity for specificity. In effect the ensemble started ignoring the cancerous patients because it could gain accuracy more easily by identifying the more numerous non-cancerous patients.

6.3 How is γ_* affected by Individual Network Complexity?

How does the optimal value of γ change with the individual complexity of the networks in the ensemble? Using the same datasets and methodology as in the previous section, we chose to use an ensemble of 6 networks, and varied the number of hidden nodes per network from 2 to 12. Figures 6.20 to 6.25 show this for different datasets, again plotting either MSE or error rate depending on the problem.

As a summary of these graphs, the optimal value of γ in each situation is plotted in figure 6.26. The upper bound for $M = 6$ networks is $\gamma_{upper} = \frac{M^2}{2(M-1)^2} = 0.72$. We note that the Breast Cancer dataset has $\gamma_* \approx 0.9$, above the upper bound, but again these improvements are in terms of error rates, and again were not a significant improvement over $\gamma = 0.0$. It is interesting to note that for any given dataset, the optimal value is almost invariant with increasing complexity of the individual networks.

How much gain can NC with γ_* provide?

Figures 6.27 to 6.32 illustrate the gain that can be achieved at γ_* compared to $\gamma = 0.0$ (simple ensemble learning). On the first four datasets (Friedman, Jacobs, Boston, Phoneme) it can be seen that the gap between simple ensemble learning and NC with γ_* becomes

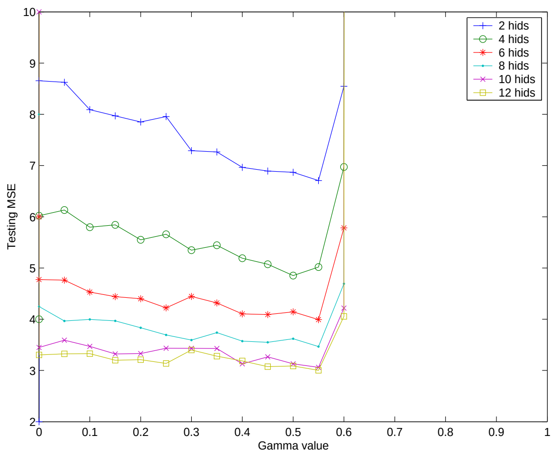

Figure 6.20: Friedman, 6 networks

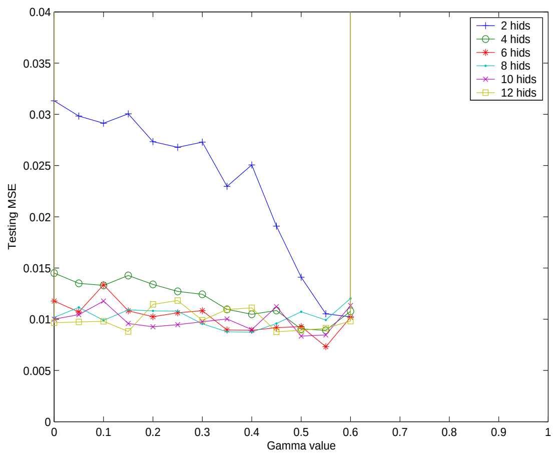

Figure 6.21: Jacobs(0.2), 6 networks

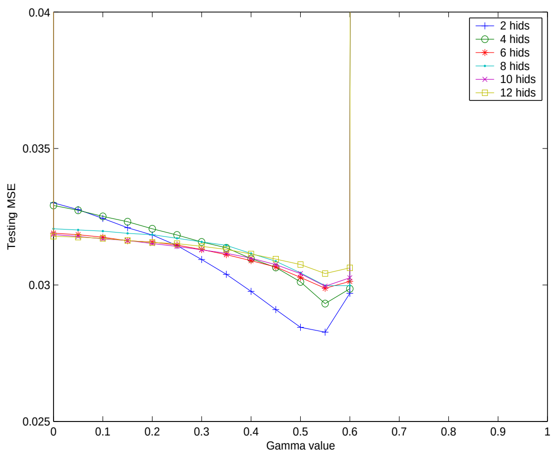

Figure 6.22: Boston, 6 networks

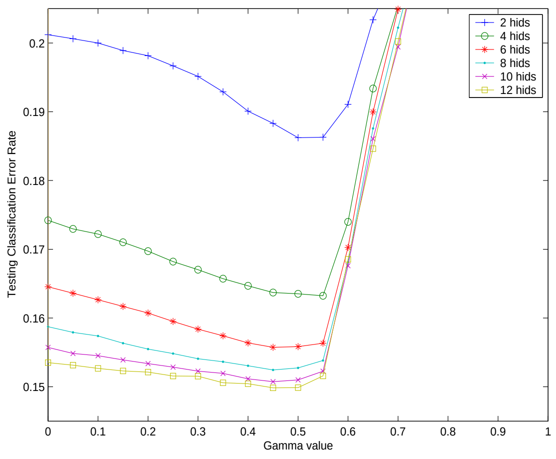

Figure 6.23: Phoneme, 6 networks

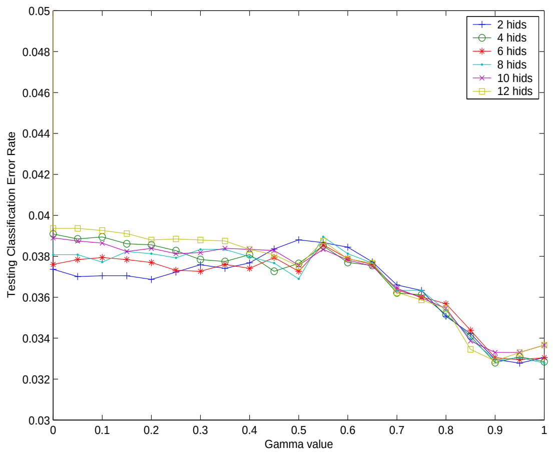

Figure 6.24: Breast Cancer, 6 networks

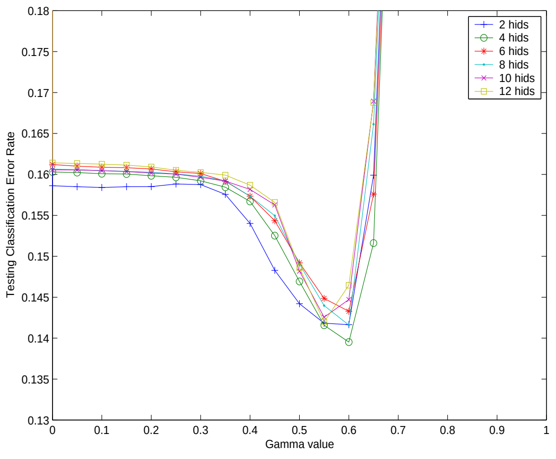

Figure 6.25: Skin, 6 networks

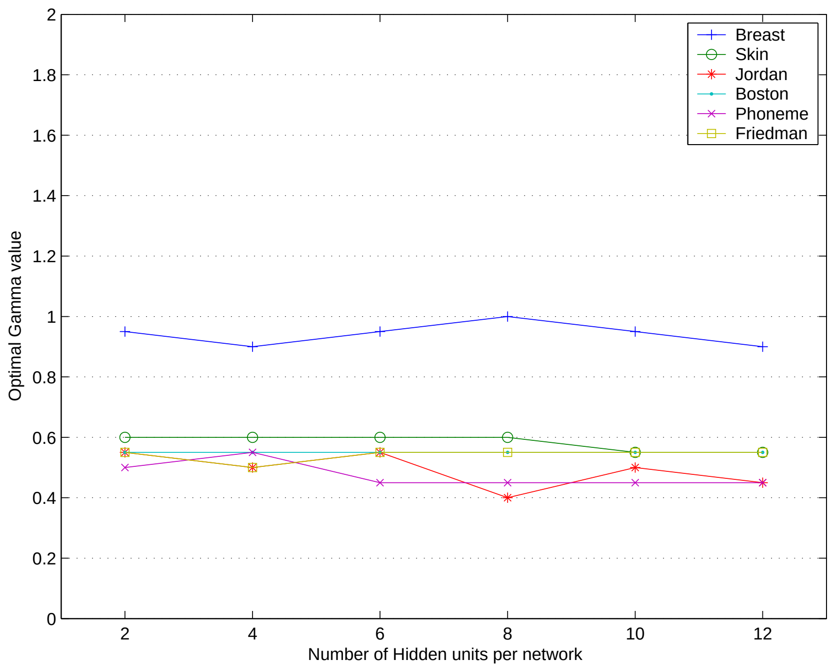

Figure 6.26: γ_* for ensembles of 6 networks

smaller and smaller as we add more hidden nodes. On the last two datasets (Breast cancer, Skin) we see no significant change in the gain achieved as we increase network complexity.

We previously found that adding more networks made NC *more effective*; the results in this section indicate that NC becomes *less effective* or at least no more effective as the individual network complexity increases. It seems that when the individual networks are relatively powerful themselves, NC can provide little further gain. This hypothesis agrees with our earlier observations in section 6.2 on the optimal γ parameter for the Jacobs dataset, and its effect with respect to complex and simple networks. From this we infer that NC seems to require *weak learners*; this point is discussed further in section 6.6.

6.4 How is γ_* affected by Noisy Data?

The original Jacobs dataset (as described in section 6.1) had Gaussian noise with variance 0.2 added to the training data. Here we created four new datasets, with noise variance 0.0, 0.1, 0.2, and 0.3 respectively; we refer to the variance 0.0 dataset as “no noise”, and the variance 0.3 set as “high noise”. Figure 6.33 shows the MSE of an ensemble of 8 networks each with 6 hidden nodes, as we vary the γ parameter; we show the four datasets as four different lines on this graph. A statistically significant improvement (two tailed t-test, $\alpha = 0.05$) was observed at γ_* for the no noise and two medium noise datasets, but for the high noise the effect of the γ value became unpredictable and no improvement over a simple ensemble could be found. The AdaBoost algorithm is well known to respond badly to noise—it therefore would be interesting to conduct a comparison of NC and AdaBoost, with various noisy datasets.

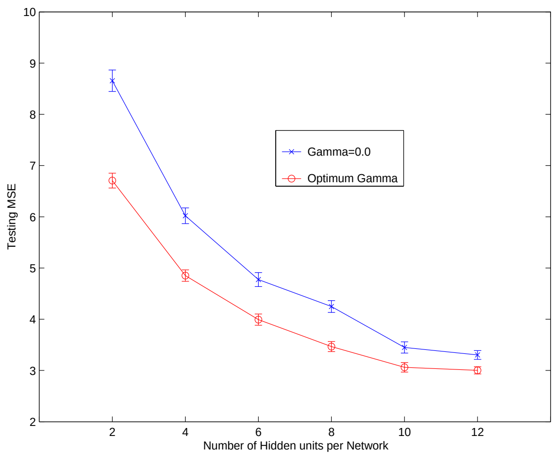

Figure 6.27: Friedman, 6 networks

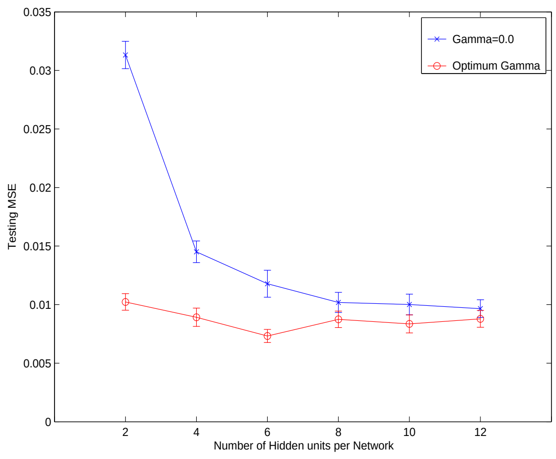

Figure 6.28: Jacobs(0.2), 6 networks

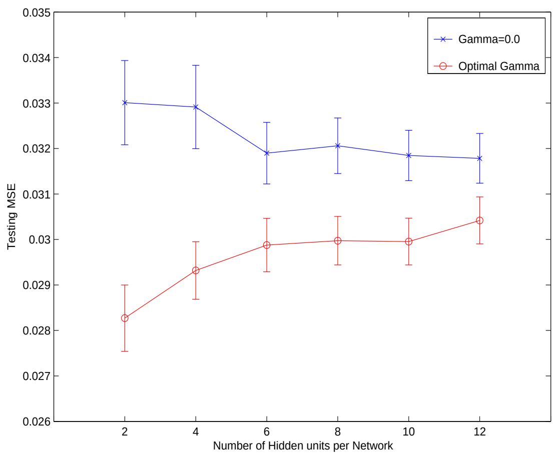

Figure 6.29: Boston, 6 networks

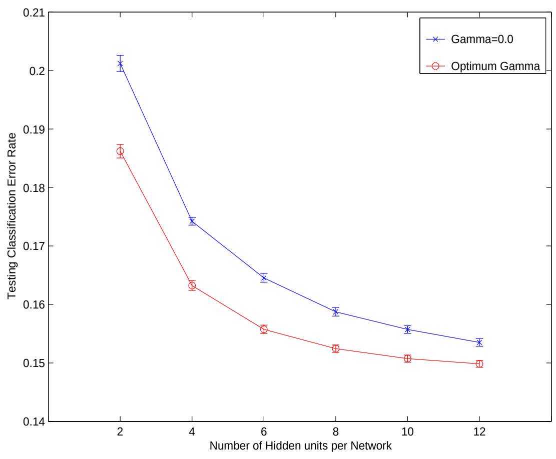

Figure 6.30: Phoneme, 6 networks

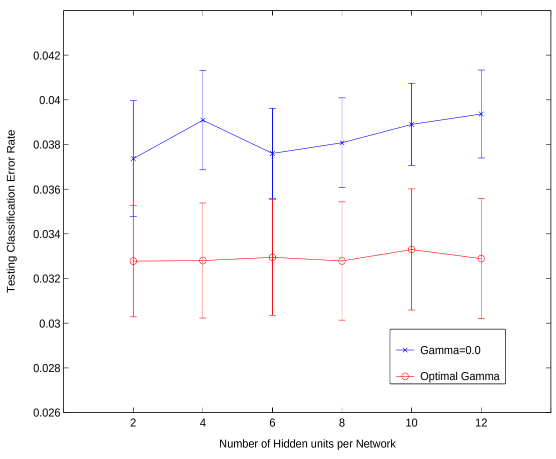

Figure 6.31: Breast Cancer, 6 networks

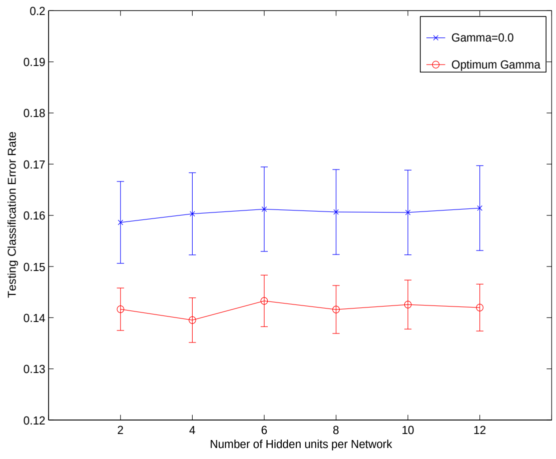

Figure 6.32: Skin, 6 networks

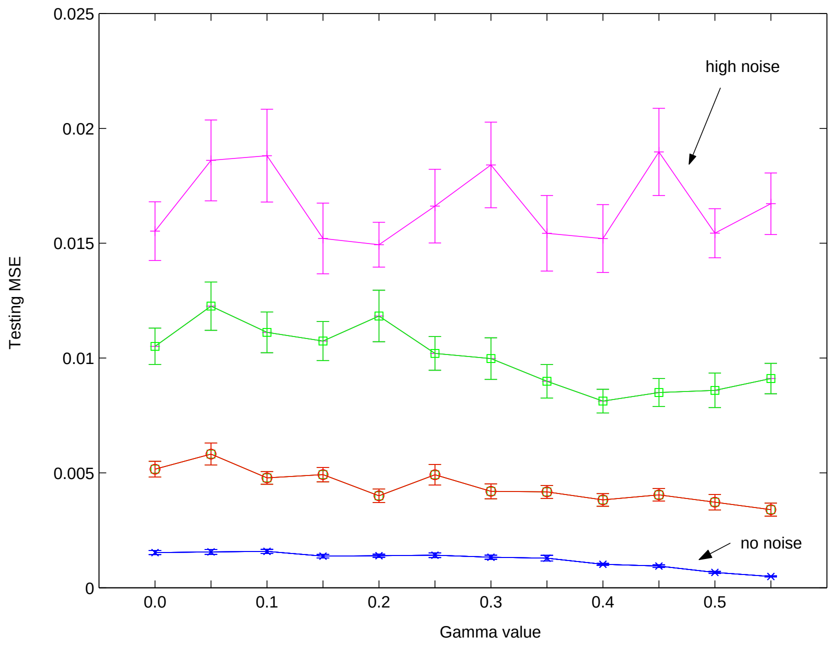

Figure 6.33: MSE vs γ as we increase training data noise on the Jacobs dataset

6.5 How does NC compare to other Algorithms?

In the literature to date, NC has only been compared to Mixture of Experts and simple ensemble learning [85]. It would be desirable to have a large scale empirical comparison of NC against several other ensemble techniques and on several different domains; however this is outside the scope of this thesis. In this section we provide the first comparison of NC against two of the most popular ensemble techniques: AdaBoost, and Bagging. This simply demonstrates that NC can provide competitive performance to other popular algorithms and is worthy of further empirical study.

We use ensembles of networks each with 6 hidden nodes, and evaluate performance of Bagging (as in figure 2.3), AdaBoost (as in figure 2.4), and NC learning on the three classification datasets we used earlier in this chapter: Breast Cancer, Skin, and Phoneme. All experimental methodology follows that described in section 6.1; the Bagged and AdaBoosted networks are trained for the same number of iterations as NC.

Results are shown in figures 6.34 to 6.36. NC significantly outperforms both Bagging and AdaBoost on Phoneme, and outperforms Adaboost on the Breast Cancer, though is not significantly better than Bagging on Breast Cancer. On the Skin dataset, the story is very different, with NC coming in last place, regardless of how many networks are present, while AdaBoost beats both NC and Bagging.

It is well known that the AdaBoost algorithm requires 'weak' learners to show an improvement over a single learning machine [148]. We noted that NC seems to display the same trend, only showing benefits when using relatively small networks. We repeated the algorithm comparisons, but using simpler networks, each with 2 hidden nodes. We expect AdaBoost to show improved performance, and hypothesize NC may do the same. Results are shown in figures 6.37 to 6.39. No noticable changes in relative performances are observed for the Skin or Breast Cancer sets. For the Phoneme set however, AdaBoost shows a very large improvement, eventually equalling NC's performance at 16 networks.

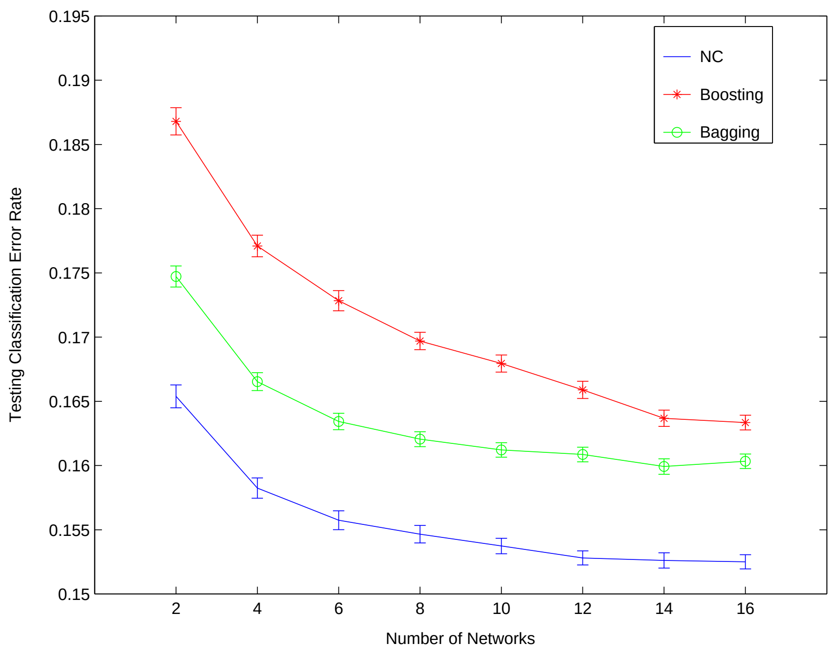

Figure 6.34: Phoneme dataset, 6 hidden nodes per network

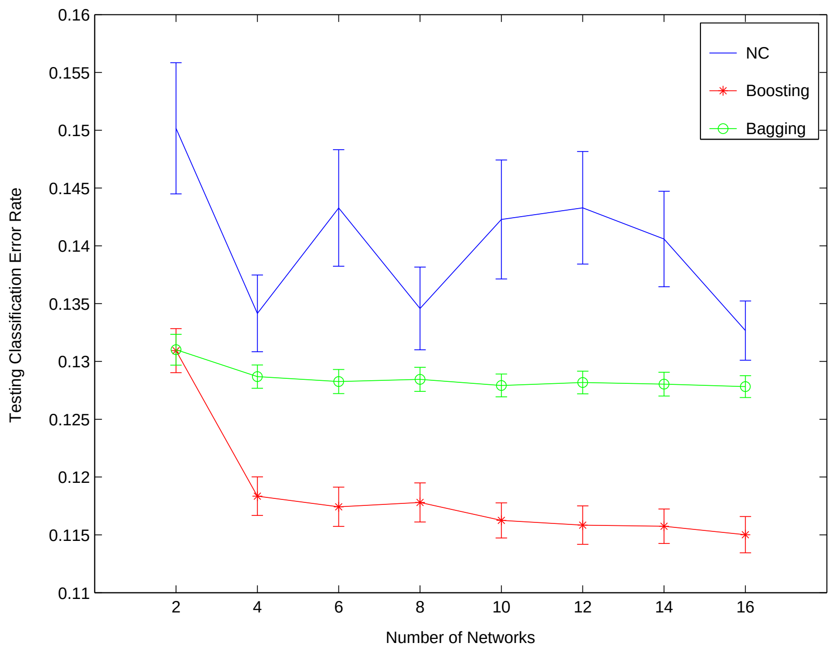

Figure 6.35: Skin dataset, 6 hidden nodes per network

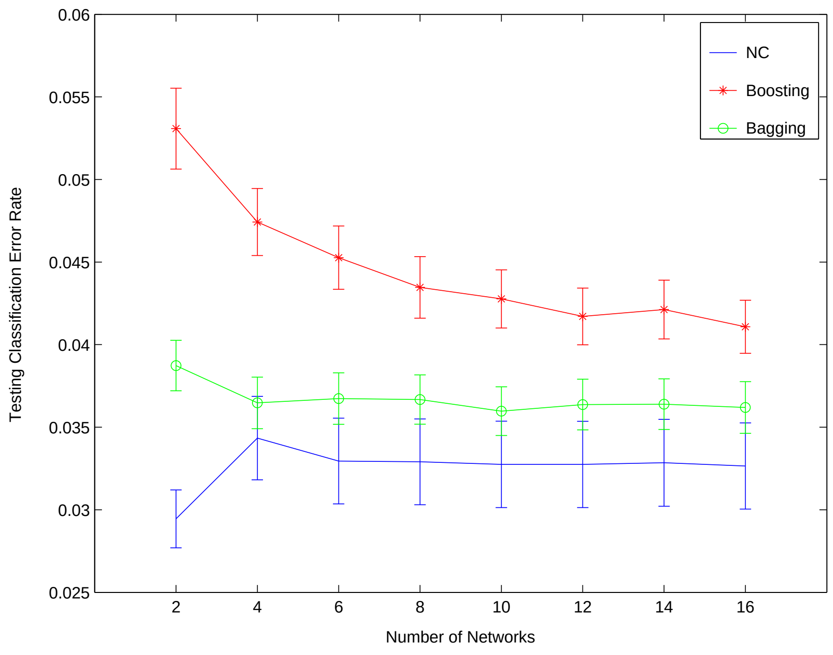

Figure 6.36: Breast Cancer dataset, 6 hidden nodes per network

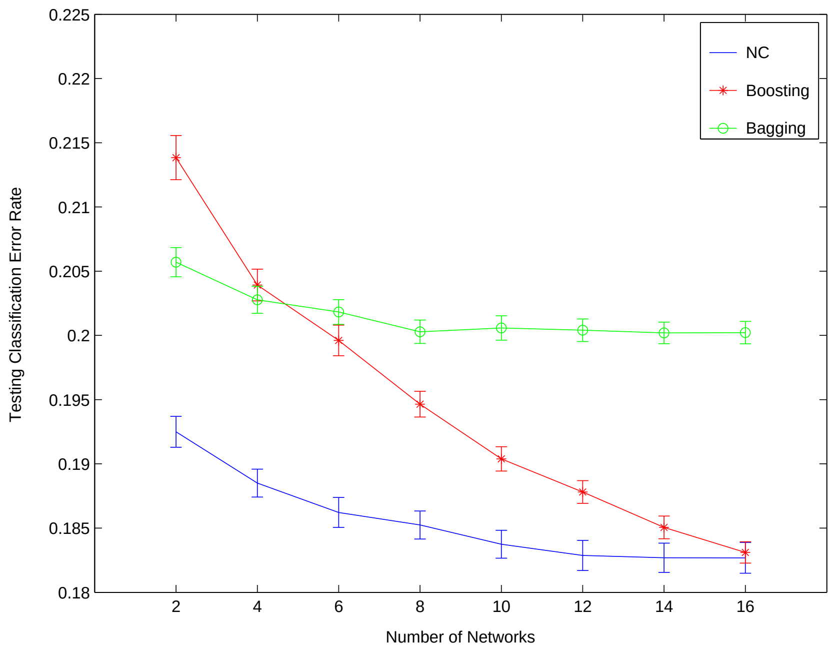

Figure 6.37: Phoneme dataset, 2 hidden nodes per network

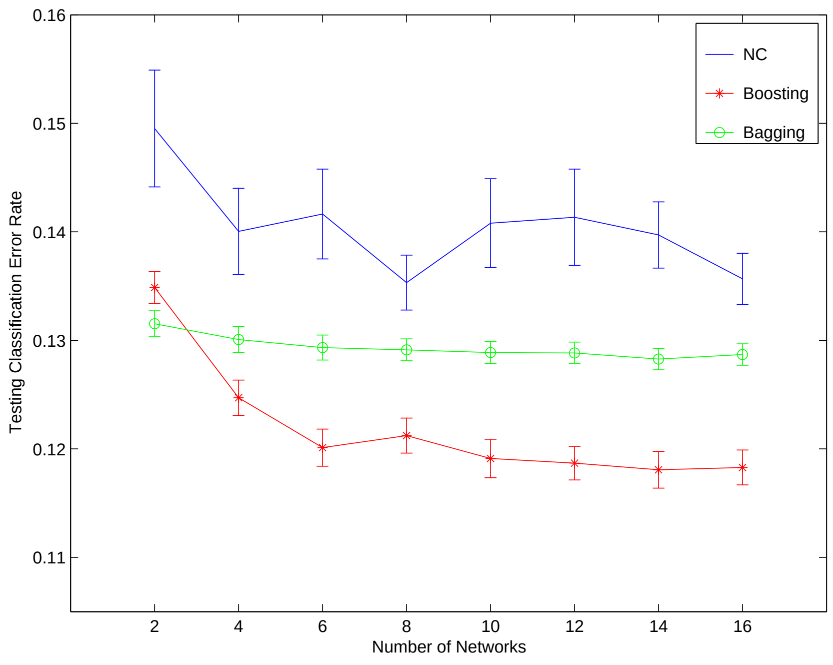

Figure 6.38: Skin dataset, 2 hidden nodes per network

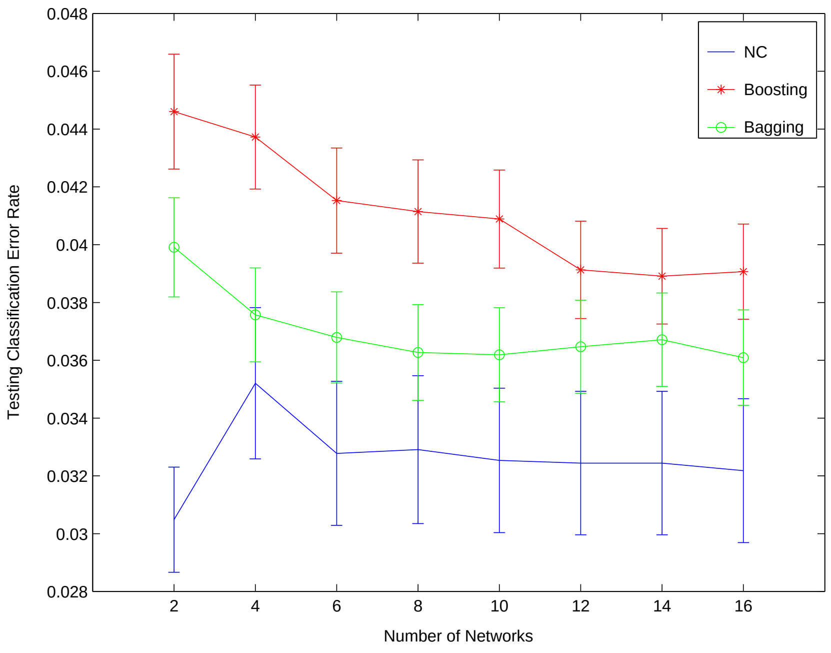

Figure 6.39: Breast Cancer dataset, 2 hidden nodes per network

6.6 Chapter Summary

In this chapter we attempted to answer two questions: (1) can we provide solid evidence to support the theoretical upper bound on γ?, and (2) how can the optimal value of γ be characterised and made predictable? We approached these by systematically varying both ensemble size and individual network complexity over a number of problems.

The upper bound on γ was determined analytically in the previous chapter; to validate this we exhaustively searched the range of possible γ values to locate the optimum, at a resolution of 0.05, for several problems and ensemble architectures. Results from all experiments supported the predicted bound, with error becoming wildly out of control when the bound was exceeded. In addition, it seems that γ_*, never occurs below 0.5, or at least no lower γ value is significantly better than 0.5. Therefore from the empirical work here we suggest that a *lower bound* on the γ parameter should be 0.5. As we add networks to the ensemble, the range necessary to explore to find γ_* becomes smaller and smaller, and eventually at around 10-15 networks (on average) it seems best to simply set $\gamma = 0.5$ immediately. In other situations, with smaller ensembles, we suggest that a good strategy for locating γ_* is to start with $\gamma = 0.5$ and search upwards. The variability in the results we have seen indicate a step-size of 0.05 should be sufficient to locate the optimum.

It seems that the optimal value of γ is determined almost entirely by how many networks are in the ensemble, and to a far lesser extent by how complex the individual networks are. It was however observed in several situations that very complex networks capable of solving the task themselves could not gain a lot from using NC, and the optimum γ was quite unpredictable. The same situation, where the behaviour of γ became unpredictable, occurred when large amounts of noise were added to the data. In summary, better relative performance gains (in comparison to a simple ensemble) were observed when using larger ensembles made up of simpler networks.

In addition to these questions, we presented the first known experiments comparing NC learning to the two most popular ensemble methods, Boosting and Bagging. Experiments show that NC is a competitive technique, significantly outperforming Boosting and Bagging on 2 out of the 3 datasets we used.

This marks the end of our investigations to date concerning Negative Correlation Learning. In the following chapter we will conclude the thesis and discuss the findings and their implications.

Chapter 7

Conclusions and Future Work

The primary question which this thesis has answered is *"Why and how does Negative Correlation Learning succeed as an ensemble method?"*. This is an important question to answer for two reasons. The first is from the viewpoint of NC as a learning technique in its own right—NC has seen a number of empirical successes (including those in chapter 6) and anything that helps us set its parameters or understand its behaviour is useful. The second is from the viewpoint of understanding how the findings in this thesis contribute toward understanding the concept of classification error "diversity" and how to enforce it. We will now discuss each of these in turn and answer the thesis questions as originally stated in the Introduction chapter.

7.1 What have we learnt...

7.1.1 About Negative Correlation Learning?

Our first objective in studying NC learning was to characterise the optimal value of the regularisation strength parameter, λ_*, such that it could be predicted reliably in future situations. In chapter 4 we applied an evolutionary algorithm to locate the optimal value in different situations. We investigated this with two datasets and under several different ensemble architectures. We noted that in general the optimal value had more variability on small ensembles, and was more stable with larger ensembles; we could not identify any definite pattern with respect to changes in the complexity of the individual networks.

Following from this, in chapter 5 we presented our main contribution: a thorough critique of Negative Correlation Learning. We showed a link to the Ambiguity decomposition [70], and explained the success of NC in terms of the bias-variance-covariance decomposition [141]. This formalisation allowed us to define an upper bound on the strength parameter (see section 5.2.2) and understand how NC relates to other algorithms. When taking into account our new understanding of the parameters, significant reductions in error were observed in empirical tests.

We then engaged in a detailed study of the error gradient and how it changes when using NC learning. We showed that the error of an NC learning ensemble can be broken down into four components, each with its own interpretation with respect to the current state of the ensemble. Further to this we noted at the end of section 5.4.1 that NC allows a smooth transition of the error gradients between that of a fully parallel ensemble system and a single estimator.

Thesis Question 1

The work in the thesis up to this point enabled us to answer our primary thesis question: *Why and how does NC learning succeed as an ensemble method?*. The reason for the success of NC is that it trains the individual networks with error functions which more closely approximate the individual network's contribution to overall ensemble error, than that used by simple ensemble learning. It achieves this by using the *Ambiguity* term [70] as a regularisation factor. The expected value of the Ambiguity term provides the missing second half of the ensemble error, that simple ensemble learning does not include (see chapter 5). As an additional observation, after our studies of the error gradient, we also concluded that NC succeeds because it allows us to choose the optimum blend between training an ensemble and training a single estimator.

The theoretical chapter served to illustrate how NC is not merely a heuristic technique as originally thought, but *fundamentally* tied to the dynamics of training an ensemble system with the mean squared error function. The observations we made are in fact all *properties of the mean squared error function.* NC is therefore best viewed as a *framework* rather than an algorithm itself. In summary, the observations we made in chapter 5 are not specific to any type of predictor, but general to any predictor that can minimise a quadratic loss function. As a consequence of this we could imagine any other type of machine learning device that could make use of the NC framework. McKay and Abbass [95], though they used NC in its original form with the λ parameter, recognised this also, and applied Genetic Programming as the machine learning method. It would be interesting to see how well the NC framework applies in ensembles of other estimator types, such as RBF networks or even just simple polynomial functions.

Thesis Question 2

In chapter 6 we attempted to answer two questions: (1) can we provide solid evidence to support the theoretical upper bound on γ?, and (2) how can the optimal value of γ be characterised and made predictable? We approached these by systematically varying both ensemble size and individual network complexity over a number of problems.

Here we addressed our second thesis question: *How can we provide guidance for setting λ in the NC learning algorithm?*. From our theoretical studies we realised that part of the λ parameter can be determined analytically (see section 5.2.2), and also an upper bound can be placed on it. Results from our empirical studies in chapter 6 continued to support the predicted upper bound for γ, with error becoming wildly out of control when the bound was exceeded. In addition, it seems that γ_*, never occurs below 0.5, or at least no lower γ value is significantly better than 0.5. Therefore from the empirical work here we suggested that a *lower bound* on the γ parameter should be 0.5. As we add networks to the ensemble, the range necessary to explore to find γ_* becomes smaller and smaller, and eventually at around 10-15 networks (on average) it seems best to simply set $\gamma = 0.5$ immediately. However, we also showed that equivalent performance can be obtained from smaller ensembles that use a strength parameter slightly larger than 0.5. In these situations, with smaller ensembles, we suggested that a good strategy for locating γ_* is to start with $\gamma = 0.5$ and search upwards.

The variability in the results we have seen indicate a step-size of 0.05 should be sufficient to locate the optimum.

It seems that the optimal value of γ is determined almost entirely by how many networks are in the ensemble, and to a far lesser extent by how complex the individual networks are. It was however observed in several situations that very complex networks capable of solving the task themselves could not gain a lot from using NC, and the optimum γ was quite unpredictable. The same situation, where the behaviour of γ became unpredictable, occurred when large amounts of noise were added to the data. In summary, better relative performance gains (in comparison to a simple ensemble) were observed when using larger ensembles made up of simpler networks.

At the end of chapter 6 we presented the first known experiments comparing NC learning to the two most popular ensemble methods, Boosting and Bagging. Experiments show that NC is a competitive technique, significantly outperforming Boosting and Bagging on 2 out of the 3 datasets we used. These experiments are intended to provide a proof-of-concept that NC can perform competitively; it is not intended that this be a comprehensive empirical comparison. There are a number of extensions which could be performed to make the comparison more meaningful. One possibility is to allow the Boosted and Bagged networks to train for more iterations; it is well known that Bagging benefits from overfitted individuals in the ensemble.

7.1.2 About Diversity in General?

In chapter 2 we reviewed the existing theoretical and heuristic explanations of what error "diversity" is, and how it affects the error of the overall ensemble. We described the two most prominent theoretical results for regression ensembles: the Ambiguity Decomposition and the bias-variance-covariance decomposition. We demonstrated what we believe to be the first formal link between these two, showing how they relate to each other. This link turned out to be crucial in our discoveries regarding Negative Correlation Learning.

As part of the literature review, we reviewed practical techniques that have been applied to create well-performing ensembles exhibiting diversity. We suggested a possible way to structure the field, based on how techniques choose to create error diversity—we noted that this improves upon a previously suggested taxonomy [124] of neural network ensembles. As a parallel to this taxonomy, we introduced the *Ambiguity* family of methods, all of which exploit the Ambiguity decomposition of the ensemble error. This was the first time this family of algorithms had been brought together, and as mentioned, our later work showed that NC learning was also a member of this family—the first method to directly utilise Ambiguity for weight updates in the networks.

Thesis Question 3

The third thesis question was *"Can we identify a good way to understand classifier ensemble diversity, and can we explicitly control it?"*. This is a difficult question, outstanding in the field for over a decade now. We have not been able to answer this question in full, but we believe the thesis observations have contributed toward an overall contiuing effort.

In chapter 2, we described the current state of the art in formalising the concept of diversity for classification ensembles, showing a paradox in that though it is widely sought after, it is still ill-defined. The type of diversity controlled by NC learning is very different from this one—strictly, NC currently works in a regression framework, function approximation of real valued numbers. It can solve classification tasks because we formulate a classification problem as a regression one, approximating the posterior probabilities. From this viewpoint, "diversity" can be formally defined in terms of covariance between continuous random variables; this has been studied extensively in the multiple regression literature as *multicollinearity* [88]. Therefore we understand that NC is a method for decreasing collinearity between the network outputs. The corresponding question that has plagued ensemble researchers for the last decade is "what is *classification* error diversity?". When we approach the question from this chain of thought, a possible new route to answering it is highlighted. Since diversity in the regression context is understood as the covariance portion of the bias-variance-covariance decomposition (for quadratic loss functions, as shown in chapter 3) then maybe we can understand classification diversity by forming a bias-variance-covariance decomposition for zero-one loss functions. The possibilities for this will be considered in next section.

7.2 Future work

Quantifying Classification Error Diversity

At the time of writing, James [58] is the most comprehensive work on bias-variance decompositions for zero-one loss[1]. This shows that two *types* of bias and variance exist. For squared loss as we have discussed in this thesis, they are identical. For zero-one loss, and general loss functions, the differences start to appear. James [58] defines general bias and variance terms, which have the typical properties we discussed in section 2.1.3. He notes that unlike squared loss, they cannot be derived from a single expression and be additive terms. He shows the *systematic effect* and *variance effect* as parallels to these, that can be additive and derived from one expression. These terms define the *effect* that bias and variance have on the predictor's error. A fundamental part of James' decomposition is to use the *mode* operator instead of the conventional statistical expectation operator, $E\{\cdot\}$. When combining classifiers we can use the same operator, the *plurality vote*, which means picking the most common class predicted by the ensemble members. On two class problems, this is equivalent to *majority vote.* It seems that an extension of these *effect* terms into something akin to the bias-variance-covariance decomposition may prove interesting for research in majority voting ensembles.

Further Analysis of Overfitting Behaviour

A very desirable addition to the work in chapter 6 is a detailed analysis of the overfitting behaviour of NC. We showed that NC transforms the ensemble into a single large predictor at particular values of the strength parameters. With such a large predictor, we may expect it to overfit to noisy data. We hypothesized that the choice of γ strength parameter allows us to choose the optimal point in between an ensemble and a large estimator, allowing us to control the optimal degree of fit for our data. Our investigations however were mostly conducted with a fixed number of iterations for training; we have no definite evidence that overfitting has not occurred. Since a larger γ value serves to bring the ensemble closer

[1]In fact this work applies to *any* symmetric loss function.

to being a single estimator, we might expect the ensemble has overfitted at these larger values of γ. If we control the overfitting with *early stopping* [111], we may be able to improve performance further. Early stopping is by no means a straightforward procedure; Prechelt [111] analyses 14 methods of stopping criterion, from 3 distinct classes. For a proper analysis of NC we would need to take this into account.

NC, Mixture of Experts, and Dyn-Co

In section 2.3.3 we described the *Dyn-Co* algorithm [48], which claimed to be able to blend smoothly between a Mixture of Experts system and a pure ensemble system. We noted that with a simple ensemble, we would expect the error gradient of the ith member to be $(f_i - d)$, whereas with Dyn-Co configured as a supposedly 'pure' ensemble method, the gradient is in fact $\frac{1}{M}(\bar{f} - d)$. However, seen in relation to NC and our findings about the gradient components in chapter 5, this opens up a new interesting point. If we are learning with NC, when $\lambda = 1$ or equivalently when $\gamma = \frac{M}{2(M-1)}$, the gradient for the ith ensemble member is $(\bar{f} - d)$. This is directly proportional to Dyn-Co's gradient, and we could easily add a scaling term to make them equivalent.

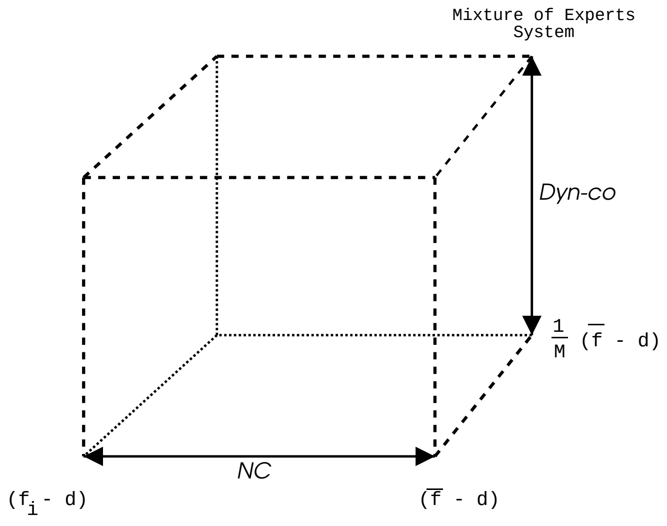

Figure 7.1: Relationships between error gradients in NC, Mixture of Experts, and Dyn-Co

Regard figure 7.1; we showed in chapter 5 that NC is able to blend smoothly from a simple ensemble, with gradient $(f_i - d)$, to a single large predictor with gradient $(\bar{f} - d)$. Dyn-Co can blend smoothly from a system with gradient $\frac{1}{M}(\bar{f} - d)$ to a Mixture of Experts system. Does this mean there exists a smooth three-dimensional space between these three algorithms, as we depict above? The similarities between the gradient components here cannot be coincidental; it would therefore be interesting to further investigate the links between all three methods.

Definition of a Tighter Upper Bound

We defined a conservative upper bound on the strength parameters by considering the state of the Hessian matrix. We showed that if the upper bound is exceeded, some of the leading diagonal elements will become negative, and therefore the entire Hessian matrix is non-positive definite. However, it is possible that the matrix could become non-positive definite *before* our upper bound is reached. An interesting extension to this work would be to investigate whether it is possible to define a tighter bound. We formulate this question as: *"Are there any general conditions for the off-diagonal elements of the Hessian, that will force non-positive definiteness, in spite of all leading diagonal elements being positive?"*. Some standard results from linear algebra [147] that may prove fruitful are as follows. A matrix H that is positive definite exhibits, among others, the following properties:

1. $H_{ii} + H_{jj} > 2H_{ij}$ for all $i \neq j$
2. The element with the largest modulus lies on the leading diagonal.

If we were to take these entries in the Hessian matrix, involving λ or γ, we may be able to define a similar inequality as that which allowed us to derive our current bound. However, these expressions are almost certainly input-dependent, so we may not be able to derive a bound that is constant for all ensemble architectures as we currently have.

Other Regularisation Methods for Enforcing Diversity

NC has shown it is possible to enforce diversity with a regularisation term in the error function. It would be desirable to have a large empirical investigation, over *several* datasets, of different regularization terms for enforcing diversity. As part of his investigations, Liu [85] empirically studied another possible penalty function for NC, in addition to the one that we have concentrated on in this thesis. This was:

$$p_i = (f_i - 0.5) \sum_{j \neq i} (f_j - 0.5) \tag{7.1}$$

which was found to be suitable for classification problems [154]. In addition, McKay and Abbas [94] have used:

$$p_i = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (f_i - d)^4}, \tag{7.2}$$

The theoretical findings in this thesis do not apply to such penalties, and they therefore remain an area in need of investigation.

So far we have only considered differentiable functions, that would be suitable for gradient descent techniques. What about non-differentiable penalty terms? One possible interesting algorithm might be to have an error function:

$$e_i = \frac{1}{M} \sum_i L(f_i, d) + \gamma p_i \tag{7.3}$$

where L is the zero-one loss function. The penalty p_i could be:

$$p_i = \frac{1}{M} \sum_i L(f_i, \bar{f}) \tag{7.4}$$

Since e_i and p_i are non-differentiable functions, the parameters of the estimator f_i could not be updated with gradient descent using this error function. Instead we could imagine using an evolutionary algorithm like the one in chapter 4 to update the parameters.

Links to Evolutionary Diversity

The term "diversity" is also often used in the evolutionary computation literature, in the context of maintaining diversity in the population of individuals you are evolving [1, 29, 64]. This has a very different meaning to the concept of diversity as discussed in this thesis. In evolutionary algorithms, we wish to maintain diversity in order to ensure we have *explored a large area of the search space* and not focussed our search onto an unprofitable area. When we decide our search has continued for long enough, we will typically take the best performing individual found so far, *ignoring the other individuals in the population.* The optimal diversity in this context will be that which optimises the explore-exploit trade off, such that the best points in the search space are found quickly and efficiently. This is in contrast to ensemble diversity methods, which create diversity with the intention that the 'population' of ensemble members will *be combined.* As such, evolutionary diversity methods do not concern themselves with creating a population that is *complementary* in any way, but instead with just ensuring the maximum amount of the hypothesis space is being explored in order to find the best individual.

In spite of these differences, some researchers have found interesting parallels [153]. A technique which we believe to be relevant to this thesis is *fitness sharing* [28, 93]. This forces differences among the individuals by adding penalty terms to their fitness functions; a penalty is higher if the individual is similar to others in the population. This penalty is somewhat of a heuristic, and its dynamics are still being understood [27]. We have studied NC learning under an assumption that our search method is gradient descent, however it is easy to imagine applying an evolutionary search algorithm like the one in chapter 4 to the weights of the networks in our ensemble, using the NC error terms in the fitness function. With the statistics framework we have provided for NC learning, further study may provide leverage into a more formal basis for evolutionary techniques like Fitness Sharing.

Appendix A

Proof of the Ambiguity Decomposition

In the original paper by Krogh & Vedelsby [70], the proof of the Ambiguity decomposition was omitted for space considerations. We have presented an alternative proof in the thesis body (see chapter 3). For completeness we now present the details of Krogh & Vedelsby's proof. Given that the ensemble output is a convex combination of the individual members:

$$f_{ens} = \sum_i w_i f_i \tag{A.1}$$

we show that:

$$(f_{ens} - d)^2 = \sum_i w_i (f_i - d)^2 - \sum_i w_i (f_i - f_{ens})^2 \tag{A.2}$$

We must first make some definitions. The *quadratic error* of network i and of the ensemble, respectively, are:

$$e_i = (f_i - d)^2 \tag{A.3}$$

$$e_{ens} = (f_{ens} - d)^2 \tag{A.4}$$

The *ambiguity* of a single member of the ensemble is defined as

$$v_i = (f_i - f_{ens})^2 \tag{A.5}$$

The *ensemble ambiguity* on input n is

$$v_{ens} = \sum_i w_i v_i = \sum_i w_i (f_i - f_{ens})^2 \tag{A.6}$$

It is simply the variance of the weighted ensemble around the weighted mean, and it measures the disagreement among the networks on input n. Now we will show that the ensemble quadratic error e_{ens} can be expressed in terms of the individual network quadratic errors e_i and their ambiguity v_{ens}. Taking equation (A.6),

$$\begin{aligned} v_{ens} &= \sum_i w_i \left[f_i - f_{ens}\right]^2 \\ &= \sum_i w_i \left[f_i - d + d - f_{ens}\right]^2 \end{aligned}$$

$$
\begin{aligned}
&= \sum_i w_i \left[(f_i - d) - (f_{ens} - d)\right]^2 \\
&= \sum_i w_i \left[(f_i - d)^2 - 2(f_i - d)(f_{ens} - d) + (f_{ens} - d)^2\right]
\end{aligned}
$$

then incorporating (A.3) and (A.4) we have

$$
\begin{aligned}
v_{ens} &= \sum_i w_i \left[e_i - 2(f_i - d)(f_{ens} - d) + e_{ens}\right] \\
&= \sum_i w_i e_i - \sum_i w_i 2(f_i - d)(f_{ens} - d) + \sum_i w_i e_{ens} \\
&= \sum_i w_i e_i - 2(f_{ens} - d)\sum_i w_i(f_i - d) + e_{ens} \\
&= \sum_i w_i e_i - 2(f_{ens} - d)^2 + e_{ens} \\
&= \sum_i w_i e_i - 2e_{ens} + e_{ens} \\
&= \sum_i w_i e_i - e_{ens}
\end{aligned}
$$

then rearranging this, we have shown:

$$
e_{ens} = \sum_i w_i e_i - v_{ens}
$$

Re-expanding this gives us our original equation:

$$
(f_{ens} - d)^2 = \sum_i w_i(f_i - d)^2 - \sum_i w_i(f_i - f_{ens})^2 \tag{A.7}
$$

and we have proved the Ambiguity Decomposition.

Appendix B

The Relationship between Ambiguity and Covariance

We now show the exact link between the Ambiguity decomposition and the bias-variance-covariance decomposition. The bias-variance-covariance decomposition gives us:

$$E\{(\bar{f}-d)^2\} = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1-\frac{1}{M}\right)\overline{covar} \tag{B.1}$$

Now using the Ambiguity decomposition, we have the result:

$$E\{\frac{1}{M}\sum_i (f_i-d)^2 - \frac{1}{M}\sum_i (f_i-\bar{f})^2\} = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1-\frac{1}{M}\right)\overline{covar} \tag{B.2}$$

It would be interesting to understand what portions of the bias-variance-covariance decomposition correspond to the ambiguity term. We place an α in front of the Ambiguity term, then derive the relationship between the left and right sides of equation (B.2). Wherever the α appears in the derivation will indicate how the Ambiguity term plays a role in the bias-variance-covariance decomposition. We have:

$$\begin{aligned} e_{ens} &= E\Big\{\frac{1}{M}\sum_i \left[(f_i-d)^2 - \alpha(f_i-\bar{f})^2\right]\Big\} \\ &= \frac{1}{M}\sum_i \Big[E\Big\{(f_i - E\{\bar{f}\} + E\{\bar{f}\} - d)^2 - \alpha(f_i - E\{\bar{f}\} + E\{\bar{f}\} - \bar{f})^2\Big\}\Big] \end{aligned}$$

now multiply out the brackets:

$$\begin{aligned} e_{ens} &= \frac{1}{M}\sum_i \Big[E\Big\{(f_i - E\{\bar{f}\})^2 + (E\{\bar{f}\}-d)^2 + 2(f_i - E\{\bar{f}\})(E\{\bar{f}\}-d) \\ &\quad - \alpha(f_i - E\{\bar{f}\})^2 - \alpha(E\{\bar{f}\}-\bar{f})^2 - 2\alpha(f_i - E\{\bar{f}\})(E\{\bar{f}\}-\bar{f})\Big\}\Big] \end{aligned}$$

and evaluate the expectation and summation:

$$\begin{aligned} e_{ens} &= \frac{1}{M}\sum_i E\Big\{(f_i - E\{\bar{f}\})^2\Big\} + (E\{\bar{f}\}-d)^2 \\ &\quad - \alpha\frac{1}{M}\sum_i E\Big\{(f_i - E\{\bar{f}\})^2\Big\} - \alpha E\Big\{(\bar{f} - E\{\bar{f}\})^2\Big\} - 2\alpha E\Big\{(\bar{f} - E\{\bar{f}\})(E\{\bar{f}\}-\bar{f})\Big\} \end{aligned}$$

and by rearranging the last term we obtain:

$$\begin{aligned} e_{ens} &= \frac{1}{M}\sum_i E\Big\{(f_i - E\{\bar{f}\})^2\Big\} + (E\{\bar{f}\} - d)^2 \\ &\quad - \alpha\frac{1}{M}\sum_i E\Big\{(f_i - E\{\bar{f}\})^2\Big\} + \alpha E\Big\{(\bar{f} - E\{\bar{f}\})^2\Big\} \end{aligned}$$

Obviously now if we remove the α term that we have been using, this would simplify to give us the bias of $\bar{f}$, plus the variance of $\bar{f}$: which we could then break down further using the bias-variance-covariance decomposition as we showed earlier. The interesting part here though, is the term that would cancel out. The expected value of the Ambiguity term is equal to whatever parts of this that contain the α term. Therefore:

$$\begin{aligned} E\{\frac{1}{M}\sum_i (f_i - \bar{f})^2\} &= \frac{1}{M}\sum_i E\{(f_i - E\{\bar{f}\})^2\} - E\{(\bar{f} - E\{\bar{f}\})^2\} \\ &= \text{``\textit{interaction term}''} - var(\bar{f}) \end{aligned}$$

And the other side of the Ambiguity decomposition, the expected value of the average individual error is whatever parts *do not* contain α :

$$\begin{aligned} E\{\frac{1}{M}\sum_i (f_i - d)^2\} &= \frac{1}{M}\sum_i E\Big\{(f_i - E\{\bar{f}\})^2\Big\} + (E\{\bar{f}\} - d)^2 \\ &= \text{``\textit{interaction term}''} + bias(\bar{f})^2 \end{aligned}$$

This "interaction term" is present in both sides, and cancels out to allow the normal bias-variance decomposition of ensemble error. But what does it *mean*? If we examine it a little further:

$$\begin{aligned} \frac{1}{M}\sum_i E\{(f_i - E\{\bar{f}\})^2\} &= \frac{1}{M}\sum_i E\{(f_i - E_i\{f_i\} + E_i\{f_i\} - E\{\bar{f}\})^2\} \\ &= \frac{1}{M}\sum_i E_i\{(f_i - E_i\{f_i\})^2\} + \frac{1}{M}\sum_i (E_i\{f_i\} - E\{\bar{f}\})^2 \end{aligned}$$

This is the average variance of the estimators, plus the average squared deviation of the expectations of the individuals from the expectation of the ensemble. Both the average individual error and the Ambiguity contain this interaction; when we subtract Ambiguity from average individual error, as in equation (B.2), the interaction cancels out, and we get the original bias-variance-covariance decomposition back. The fact that this term is common to both parts shows clearly why we cannot maximise ensemble Ambiguity without affecting the ensemble bias component as well.

Appendix C

Calculations Supporting the Strength Parameter Bounds

We now present additional calculations supporting the work on the theoretical upper bounds for the λ and γ parameters, as in chapter 5. All notational conventions and definitions follow those used in that chapter.

Assuming a network with a linear output function, we wish to derive one of the entries in the leading diagonal of the Hessian matrix. If one of these entries, $\frac{\partial^2 e_i}{\partial {w_i}^2}$, is zero or less, then the Hessian is guaranteed to be non-positive definite, an undesirable prospect. For an arbitrary weight w_i in the *output layer*, connecting a hidden node h_j to the output node f_i, we have:

$$\frac{\partial e_i}{\partial w_i} = \frac{\partial e_i}{\partial f_i}\frac{\partial f_i}{\partial w_i} \tag{C.1}$$

$$\frac{\partial^2 e_i}{\partial {w_i}^2} = \left[\frac{\partial}{\partial w_i}\frac{\partial e_i}{\partial f_i}\right]\frac{\partial f_i}{\partial w_i} + \left[\frac{\partial}{\partial w_i}\frac{\partial f_i}{\partial w_i}\right]\frac{\partial e_i}{\partial f_i} \tag{C.2}$$

$$\tag{C.3}$$

Taking the first term on the right hand side:

$$\begin{aligned}\frac{\partial e_i}{\partial f_i} &= (f_i - d) - \lambda(f_i - \bar{f}) \\ \frac{\partial}{\partial w_i}\frac{\partial e_i}{\partial f_i} &= h_j - \lambda(h_j - \frac{1}{M}h_j) \\ &= \Big(1 - \lambda(1 - \frac{1}{M})\Big)h_j \qquad \text{(C.4)}\end{aligned}$$

Now for the second term, remembering that we use linear output nodes, we have:

$$\frac{\partial f_i}{\partial w_i} = h_j \tag{C.5}$$

$$\frac{\partial}{\partial w_i}\frac{\partial f_i}{\partial w_i} = \frac{\partial^2 f_i}{\partial {w_i}^2} = 0 \tag{C.6}$$

Therefore we simply have:

$$\frac{\partial^2 e_i}{\partial {w_i}^2} = \Big[\Big(1 - \lambda(1 - \frac{1}{M})\Big) h_j\Big] h_j + \Big[0\Big] \frac{\partial e_i}{\partial f_i} \tag{C.7}$$

$$= \Big(1 - \lambda(1 - \frac{1}{M})\Big) {h_j}^2 \tag{C.8}$$

It is interesting to observe that since we have:

$$\frac{\partial e_i}{\partial f_i} = (f_i - d) - \lambda(f_i - \bar{f}) \tag{C.9}$$

$$\frac{\partial^2 e_i}{\partial {f_i}^2} = 1 - \lambda(1 - \frac{1}{M}) \tag{C.10}$$

then we can see:

$$\frac{\partial^2 e_i}{\partial {w_i}^2} = \frac{\partial^2 e_i}{\partial {f_i}^2} h_j^2 \tag{C.11}$$

Since h_j^2 is positive, this demonstrates that the sign of the leading diagonal entry $\frac{\partial^2 e_i}{\partial {w_i}^2}$ in the Hessian is decided by the sign of $\frac{\partial^2 e_i}{\partial {f_i}^2}$.

Appendix D

Evolving λ : Empirical Data

This appendix presents the exact values of λ that were discovered as near optimal, during experiments with the evolutionary algorithm described in chapter 4.

Ensemble configuration	2 hids	6 hids	12 hids
2 nets	0.7137	0.8676	0.8998
6 nets	0.9436	0.9467	0.9129
12 nets	0.9691	0.9766	0.9144

Figure D.1: Near Optimal λ values for Boston dataset

Ensemble configuration	2 hids	6 hids	12 hids
2 nets	0.9222	0.6778	0.7234
6 nets	0.9518	0.8831	0.8819
12 nets	0.9101	0.9059	0.8678

Figure D.2: Near Optimal λ values for Phoneme dataset

Ensemble configuration	2 hids	6 hids	12 hids
2 nets	0.9198	0.6189	0.8374
6 nets	0.9599	0.8088	0.8511
12 nets	0.9208	0.9143	0.8998

Figure D.3: Near Optimal λ values for Phoneme dataset (fitness function used classification error rate)

Bibliography

[1] Joon-Hyun Ahn and Sung-Bae Cho. Speciated neural networks evolved with fitness sharing technique. In *Proceedings of the Congress on Evolutionary Computation*, pages 390–396, Seoul, Korea, May 27-30 2001.

[2] K. Ali. A comparison of methods for learning and combining evidence from multiple models. Technical Report UCI TR #95-47, University of California, Irvine, Dept. of Information and Computer Sciences, 1995.

[3] R. Avnimelech and N. Intrator. Boosting regression estimators. *Neural Computation*, 11:491–513, 1999.

[4] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[5] Dennis Bahler and Laura Navarro. Methods for combining heterogeneous sets of classifiers. In *17th Natl. Conf. on Artificial Intelligence (AAAI), Workshop on New Research Problems for Machine Learning*, 2000.

[6] J. M. Bates and C. W. J. Granger. The combination of forecasts. *Operations Research Quarterly*, (20):451–468, 1969.

[7] Roberto Battiti and Anna Maria Colla. Democracy in neural nets: Voting schemes for classification. *Neural Networks*, 7(4):691–707, 1994.

[8] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1,2), 1999.

[9] Chris M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995.

[10] Christopher M. Bishop. *Neural Networks for Pattern Recogntion.* Oxford University Press, 1995.

[11] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[12] B.Parmanto, P.W.Munro, and H.R.Doyle. Improving committee diagnosis with resampling techniques. *Advances in Neural Information Processing Systems*, 8:882–888, 1996.

[13] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[14] Leo Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California, Berkeley, 1996.

[15] Leo Breiman. Randomizing outputs to increase prediction accuracy. Technical Report 518, Statistics Department, University of California, May 1998.

[16] Gavin Brown and Jeremy L. Wyatt. Negative correlation learning and the ambiguity family of ensemble methods. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2709)*, pages 266–275, Guildford, Surrey, June 2003. Springer.

[17] Gavin Brown and Jeremy L. Wyatt. The use of the ambiguity decomposition in neural network ensemble learning methods. In Tom Fawcett and Nina Mishra, editors, *20th International Conference on Machine Learning (ICML'03)*, Washington DC, USA, August 2003.

[18] Gavin Brown and Xin Yao. On the Effectiveness of Negative Correlation Learning. In *Proceedings of First UK Workshop on Computational Intelligence*, pages 57–62, 2001. Edinburgh, Scotland.

[19] Gavin Brown, Xin Yao, Jeremy Wyatt, Heiko Wersing, and Bernhard Sendhoff. Exploiting ensemble diversity for automatic feature extraction. In *Proc. of the 9th International Conference on Neural Information Processing (ICONIP'02)*, pages 1786–1790, November 2002.

[20] Peter Buhlmann and Bin Yu. Explaining bagging. Technical Report 92, ETH Zurich, Seminar Fur Statistik, May 2000.

[21] John Carney and Padraig Cunningham. Tuning diversity in bagged neural network ensembles. Technical Report TCD-CS-1999-44, Trinity College Dublin, 1999.

[22] Nitesh Chawla, Thomas Moore, Kevin Bowyer, Lawrence Hall, Clayton Springer, and Philip Kegelmeyer. Bagging is a small-data-set phenomenon. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[23] R. Clemen. Combining forecast: A review and annotated bibliography. *International Journal on Forecasting*, 5:559–583, 1989.

[24] M. Costa, E. Filippi, and E. Pasero. Artificial neural network ensembles: a bayesian standpoint. In M. Marinaro and R. Tagliaferri, editors, *Proceedings of the 7th Italian Workshop on Neural Nets*, pages 39–57. World Scientific, 1995.

[25] Padraig Cunningham and John Carney. Diversity versus quality in classification ensembles based on feature selection. In *LNCS - European Conference on Machine Learning*, volume 1810, pages 109–116. Springer, Berlin, 2000.

[26] J. A.and Madigan D. Hoeting and A.E.and Volinsky C.T. Raftery. Bayesian model averaging: A tutorial. *Statistical Science*, 44(4):382–417, 1999.

[27] Paul Darwen and Xin Yao. How good is fitness sharing with a scaling function. Technical Report CS 8/95, University of New South Wales, April 1995.

[28] Paul Darwen and Xin Yao. Every niching method has its niche: fitness sharing and implicit sharing compared. In *Proc. of Parallel Problem Solving from Nature (PPSN) IV - Lecture Notes in Computer Science 1141*. Springer-Verlag, 1996.

[29] Paul J. Darwen and Xin Yao. Speciation as automatic categorical modularization. *IEEE Trans. on Evolutionary Computation*, 1(2):100–108, 1997.

[30] P. S. de Laplace. *Deuxieme supplement a la theorie analytique des probabilites*. Paris, Gauthier-Villars, 1818. Reprinted (1847) in Oeuvres Completes de Laplace, vol. 7.

[31] T. G. Dietterich and G. Bakiri. Error-correcting output codes: a general method for improving multiclass inductive learning programs. In T. L. Dean and K. McKeown, editors, *Proceedings of the Ninth AAAI National Conference on Artificial Intelligence*, pages 572–577, Menlo Park, CA, 1991. AAAI Press.

[32] P. Domingos. Why does bagging work? A bayesian account and its implications. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 155. AAAI Press, 1997.

[33] Pedro Domingos. A unified bias-variance decomposition and its applications. In *Proc. 17th International Conf. on Machine Learning*, pages 231–238. Morgan Kaufmann, San Francisco, CA, 2000.

[34] Richard Duda, Peter Hart, and David Stork. *Pattern Classification*. John Wiley and Sons, 2001. 0-471-05669-3.

[35] Robert P. W. Duin and David M. J. Tax. Experiments with classifier combining rules. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 1857)*, pages 16–29, Calgiari, Italy, June 2000. Springer.

[36] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

[37] Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. AdaCost: misclassification cost-sensitive boosting. In *Proc. 16th International Conf. on Machine Learning*, pages 97–105. Morgan Kaufmann, San Francisco, CA, 1999.

[38] Raphaël Feraud and Olivier Bernier. Ensemble and modular approaches for face detection: A comparison. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[39] Yoav Freund, Yishay Mansour, and Robert Schapire. Why averaging classifiers can protect against overfitting. In *Eighth International Workshop on Artificial Intelligence and Statistics*, 2001.

[40] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[41] J. Friedman and P. Hall. On bagging and nonlinear estimation - available online at http://citeseer.nj.nec.com/friedman99bagging.html. Technical report, Stanford University, 1999.

[42] J.H. Friedman. Bias, variance, 0-1 loss and the curse of dimensionality. Technical report, Stanford University, 1996.

[43] Giorgio Fumera and Fabio Roli. Error rejection in linearly combined multiple classifiers. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2096)*, pages 329–338, Cambridge, UK, June 2001. Springer.

[44] Giorgio Fumera and Fabio Roli. Linear combiners for classifier fusion: Some theoretical and experimental results. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2709)*, pages 74–83, Guildford, Surrey, June 2003. Springer.

[45] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[46] Y. Grandvalet. Bagging can stabilize without reducing variance. In *ICANN'01*, Lecture Notes in Computer Science. Springer, 2001.

[47] University College London Neural Network Group. The Elena Project. http://www.dice.ucl.ac.be/neural-nets/Research/Projects/ELENA/elena.htm.

[48] Jakob Vogdrup Hansen. *Combining Predictors: Meta Machine Learning Methods and Bias/Variance and Ambiguity Decompositions*. PhD thesis, Aarhus Universitet, Datalogisk Institut, 2000.

[49] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[50] Sherif Hashem. *Optimal Linear Combinations of Neural Networks*. PhD thesis, School of Industrial Engineering, University of Purdue, 1993.

[51] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan Co., New York, 1994.

[52] Jonathan E. Higgins, Tony J. Dodd, and Robert I. Damper. Application of multiple classifier techniques to subband speaker identification with an hmm/ann system. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2096)*, pages 369–377, Cambridge, UK, June 2001. Springer.

[53] Md. M. Islam, X. Yao, and K. Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks*, 14(4):820–834, July 2003.

[54] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[55] R.A. Jacobs and M.A. Tanner. *Combining Articial Neural Nets*, chapter Mixtures of X. Springer-Verlag, London, 1999.

[56] Robert Jacobs. Bias-variance analyses of mixture-of-experts architectures. *Neural Computation*, 9:369–383, 1997.

[57] Robert A. Jacobs, Michael I. Jordan, and Andrew G. Barto. Task decomposition through competition in a modular connectionist architecture - the what and where vision tasks. *Cognitive Science*, 15:219–250, 1991.

[58] Gareth James. Variance and bias for general loss functions. *Machine Learning*, 51:115–135, 2003.

[59] J.H.Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19:1–141, 1991.

[60] D. Jimenez and N. Walsh. Dynamically weighted ensemble neural networks for classification. In *Proceedings of the International Joint Conference on Neural Networks*, 1998.

[61] Rong Jin, Yan Liu, Luo Si, Jaime Carbonell, and Alexander Hauptmann. A new boosting algorithm using input-dependent regularizer. In *20th International Conference on Machine Learning*, 2003.

[62] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[63] Mahesh V. Joshi, Vipin Kumar, and Ramesh C. Agarwal. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *ICDM*, pages 257–264, 2001.

[64] Vineet Khare and Xin Yao. Artificial speciation of neural network ensembles. In J.A.Bullinaria, editor, *Proc. of the 2002 UK Workshop on Computational Intelligence (UKCI'02)*, pages 96–103. University of Birmingham, UK, September 2002.

[65] Josef Kittler and Fuad M. Alkoot. Relationship of sum and vote fusion strategies. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2096)*, pages 339–348, Cambridge, UK, June 2001. Springer.

[66] Josef Kittler and Fabio Roli, editors. *Lecture Notes in Computer Science : First International Workshop on Multiple Classifier Systems*, volume 1857, Cagliari, Italy, June 2000.

[67] Ron Kohavi and David H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In Lorenza Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 275–283. Morgan Kaufmann, 1996.

[68] E. B. Kong and T. G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the 12th International Conference on Machine Learning*, pages 313–321. Morgan Kaufmann, 1995.

[69] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[70] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. *NIPS*, 7:231–238, 1995.

[71] L. Kuncheva. Switching between selection and fusion in combining classifiers: An experiment. *IEEE Transactions On Systems Man And Cybernetics*, 32(2):146–156, 2002.

[72] L. Kuncheva, C.Whitaker, C.Shipp, and R.Duin. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis and Applications*, 6(1):22–31, April 2003.

[73] L. Kuncheva, C. Whitaker, C. Shipp, and R. Duin. Is independence good for combining classiers. In *Proceedings of the 15th International Conference on Pattern Recognition, Barcelona, Spain*, pages 168–171, 2000.

[74] L.I. Kuncheva. That Elusive Diversity in Classifier Ensembles. In *First Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA), available as LNCS volume 2652*, pages 1126–1138, 2003.

[75] L.I. Kuncheva and C.Whitaker. Measures of diversity in classifier ensembles. *Machine Learning*, (51):181–207, 2003.

[76] L.I. Kuncheva and R.K. Kountchev. Generating classifier outputs of fixed accuracy and diversity. *Pattern Recognition Letters*, (23):593–600, 2002.

[77] Ludmila Kuncheva. Error bounds for aggressive and conservative adaboost. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2709)*, pages 25–34, Guildford, Surrey, June 2003. Springer.

[78] Ludmila Kuncheva and Christopher J. Whitaker. Using diversity with three variants of boosting: Aggressive, conservative, and inverse. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2364)*, pages 81–90, Calgiari, Italy, June 2002. Springer.

[79] Ludmila I. Kuncheva and Chris J. Whitaker. Ten measures of diversity in classifier ensembles: Limits for two classifiers. In *IEE Workshop on Intelligent Sensor Processing.* IEE, February 2001. Birmingham, UK.

[80] Edward Landesman and Magnus Hestenes. *Linear Algebra for Mathematics, Science and Engineering.* Prentice Hall, 1992.

[81] W. B. Langdon and B. F. Buxton. Genetic programming for combining classifiers. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 66–73, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[82] William B. Langdon, S. J. Barrett, and B. F. Buxton. Combining decision trees and neural networks for drug discovery. In *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, pages 60–70, Kinsale, Ireland, 3-5 April 2002.

[83] William B. Langdon and Bernard F. Buxton. Evolving receiver operating characteristics for data fusion. In *Genetic Programming, Proceedings of EuroGP'2001*, volume 2038, pages 87–96, Lake Como, Italy, 18-20 April 2001. Springer-Verlag.

[84] Yuansong Liao and John Moody. Constructing heterogeneous committees using input feature grouping. *Advances in Neural Information Processing Systems*, 12, 1999.

[85] Yong Liu. *Negative Correlation Learning and Evolutionary Neural Network Ensembles.* PhD thesis, University College, The University of New South Wales, Australian Defence Force Academy, Canberra, Australia, 1998.

[86] Yong Liu and Xin Yao. Negatively correlated neural networks can produce best ensembles. *Australian Journal of Intelligent Information Processing Systems*, 4(3/4):176–185, 1997.

[87] Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.

[88] Xycoon Ltd. Online Econometrics Textbook - Regression Extensions - Multicollinearity, available at: http://www.xycoon.com/multicollinearity.htm.

[89] R. Maclin and J. W. Shavlik. Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Canada*, pages 524–530, 1995.

[90] Richard Maclin and David Opitz. An empirical evaluation of bagging and boosting. In *AAAI/IAAI*, pages 546–551, 1997.

[91] E. Mandler and J. Schuermann. *Pattern Recognition and Artificial Intelligence*, chapter Combining the Classification Results of independent classifiers based on the Dempster/Shafer theory of evidence, pages 381–393. North Holland, Amsterdam, 1988.

[92] J. McClelland and D. Rumelhart. *Parallel Distributed Processing.* MIT Press, 1986.

[93] Bob McKay. Fitness sharing in genetic programming. In Darrell Whitley, David Goldberg, Erick Cantu-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 435–442, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[94] Robert McKay and Hussein Abbass. Analyzing anticorrelation in ensemble learning. In *Proceedings of 2001 Conference on Artificial Neural Networks and Expert Systems*, pages 22–27, Otago, New Zealand, 2001.

[95] Robert McKay and Hussein Abbass. Anticorrelation measures in genetic programming. In *Australasia-Japan Workshop on Intelligent and Evolutionary Systems*, pages 45–51, 2001.

[96] Ron Meir, Ran El-Yaniv, and Shai Ben-David. Localized boosting. In *Proc. 13th Annu. Conference on Comput. Learning Theory*, pages 190–199. Morgan Kaufmann, San Francisco, 2000.

[97] Prem Melville and Ray Mooney. Constructing diverse classifier ensembles using artificial training examples. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 505–510, Mexico, August 2003.

[98] P. Moerland and E. Mayoraz. Dynaboost: Combining boosted hypotheses in a dynamic way. Technical Report RR 99-09, IDIAP, Switzerland, May 1999.

[99] N.J.Nilsson. *Learning Machines: Foundations of Trainable Pattern-Classifying Systems.* McGraw-Hill, 1965.

[100] David Opitz. Feature selection for ensembles. In *Proceedings of 16th National Conference on Artificial Intelligence (AAAI)*, pages 379–384, 1999.

[101] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[102] David W. Opitz and Jude W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. *NIPS*, 8:535–541, 1996.

[103] N. Oza and K. Tumer. Dimensionality reduction through classifier ensembles. Technical Report NASA-ARC-IC-1999-126, NASA Ames Labs, 1999.

[104] Nikunj C. Oza. Boosting with averaged weight vectors. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2709)*, pages 15–24, Guildford, Surrey, June 2003. Springer.

[105] Nikunj C. Oza and Kagan Tumer. Input decimation ensembles: Decorrelation through dimensionality reduction. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2096)*, pages 238–247, Cambridge, UK, June 2001. Springer.

[106] D. Partridge. Network generalization differences quantified. *Neural Networks*, 9(2):263–271, 1996.

[107] D. Partridge and W. B. Yates. Engineering multiversion neural-net systems. *Neural Computation*, 8(4):869–893, 1996.

[108] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142. Chapman & Hall, London, 1993.

[109] M.P. Perrone. *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization.* PhD thesis, Brown University, Institute for Brain and Neural Systems, 1993.

[110] Tomaso Poggio, Ryan Rifkin, Sayan Mukherjee, and Alex Rakhlin. Bagging regularizes. Technical Report AI Memo 2002-003, CBCL Memo 214, MIT AI Lab, 2002.

[111] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998.

[112] Sarunas Raudys and Fabio Roli. The behavior knowledge space fusion method: Analysis of generalization error and strategies for performance improvement. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2709)*, pages 55–64, Guildford, Surrey, June 2003. Springer.

[113] Yuval Raviv and Nathan Intrator. Bootstrapping with noise: An effective regularisation technique. *Connection Science*, 8:355–372, 1996.

[114] Fabio Roli and Giorgio Fumera. Analysis of linear and order statistics combiners for fusion of imbalanced classifiers. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2364)*, pages 252–261, Calgiari, Italy, June 2002. Springer.

[115] Fabio Roli and Josef Kittler, editors. *Lecture Notes in Computer Science : Second International Workshop on Multiple Classifier Systems*, volume 2096, Cambridge, UK, July 2001.

[116] Fabio Roli and Josef Kittler, editors. *Lecture Notes in Computer Science : Third International Workshop on Multiple Classifier Systems*, volume 2364, Cagliari, Italy, June 2002.

[117] Bruce E. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science - Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3 and 4):373–384, 1996.

[118] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999.

[119] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[120] Robert E. Schapire. Theoretical views of boosting and applications. In *Algorithmic Learning Theory, 10th International Conference, ALT '99, Tokyo, Japan, December 1999, Proceedings*, volume 1720, pages 13–25. Springer, 1999.

[121] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Computational Learing Theory*, pages 80–91, 1998.

[122] M. Scott, M. Niranjan, and R. Prager. Parcel: Feature subset selection in variable cost domains. Technical Report CUED/F-INFENG/TR 323, Cambridge University, 1998.

[123] A. Sharkey and N. Sharkey. Combining diverse neural networks. *The Knowledge Engineering Review*, 12(3):231–247, 1997.

[124] Amanda Sharkey. *Multi-Net Systems*, chapter Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems, pages 1–30. Springer-Verlag, 1999.

[125] Amanda Sharkey and Noel Sharkey. Diversity, selection, and ensembles of artificial neural nets. In *Neural Networks and their Applications (NEURAP'97)*, pages 205–212, 1997.

[126] Amanda Sharkey, Noel Sharkey, and Gopinath Chandroth. Diverse neural net solutions to a fault diagnosis problem. *Neural Computing and Applications*, 4:218–227, 1996.

[127] Amanda J. C. Sharkey, Noel E. Sharkey, Uwe Gerecke, and G. O. Chandroth. The test and select approach to ensemble combination. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 1857)*, pages 30–44, Calgiari, Italy, June 2000. Springer.

[128] N.E. Sharkey, J. Neary, and A.J.C. Sharkey. Searching Weight Space for Backpropagation Solution Types. *Current Trends in Connectionism: Proceedings of the 1995 Swedish Conference on Connectionism*, pages 103–120, 1995.

[129] C.A. Shipp and L.I. Kuncheva. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, (3):135–148, 2002.

[130] Marina Skurichina, Ludmila Kuncheva, and Robert P. W. Duin. Bagging and boosting for the nearest mean classifier: Effects of sample size on diversity and accuracy. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2364)*, pages 62–71, Calgiari, Italy, June 2002. Springer.

[131] C. Y. Suen and L. Lam. Multiple classifier combination methodologies for different output levels. In *Proceedings of First International Workshop on Multiple Classifier Systems (MCS 2000)*, pages 52–66, 2000.

[132] A. Swann and N. Allinson. Fast committee learning: Preliminary results. *Electronics Letters*, 34(14):1408–1410, July 1998.

[133] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed problems.* W.H.Winston and Sons, Washington D.C., 1977.

[134] Volker Tresp. A bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

[135] Volker Tresp and Michiaki Taniguchi. Combining estimators using non-constant weighting functions. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 419–426. The MIT Press, 1995.

[136] K. Tumer and J. Ghosh. Order statistics combiners for neural classifiers. *World Congress on Neural Networks*, Vol. I:31–34, July 1995.

[137] K Tumer and J Ghosh. Robust combining of disparate classifiers through order statistics. *To appear in Pattern Analysis and Applications, Special issue on Fusion of Multiple Classifiers*, 5(2), 2002.

[138] Kagan Tumer and Joydeep Ghosh. Theoretical foundations of linear and order statistics combiners for neural pattern classifiers. Technical Report TR-95-02-98, Computer and Vision Research Center, University of Texas, Austin, 1995.

[139] Kagan Tumer and Joydeep Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, February 1996.

[140] Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3-4):385–403, 1996.

[141] N. Ueda and R. Nakano. Generalization error of ensemble estimators. In *Proceedings of International Conference on Neural Networks*, pages 90–95, 1996.

[142] Giorgio Valentini and Thomas G. Dietterich. Bias-variance analysis and ensembles of SVM. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2364)*, pages 222–231, Calgiari, Italy, June 2002. Springer.

[143] Giorgio Valentini and Thomas G. Dietterich. Low bias bagged support vector machines. In Tom Fawcett and Nina Mishra, editors, *20th International Conference on Machine Learning (ICML'03)*, Washington DC, USA, August 2003.

[144] G. Wahba, X. Lin, F. Gao, D. Xiang, R. Klein, and B. Klein. The bias-variance tradeoff and the randomized GACV. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems*, number 11, pages 620–626. MIT Press, 1999.

[145] Eric A. Wan. Combining fossil and sunspot data: Committee predictions. In *International Conference On Neural Networks (ICNN97)*, 1997.

[146] Wenjia Wang, Phillis Jones, and Derek Partridge. Diversity between neural networks and decision trees for building multiple classifier systems. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 1857)*, pages 240–249, Calgiari, Italy, June 2000. Springer.

[147] Eric Weisstein. Mathworld, entry for positive definite matrices : http://mathworld.wolfram.com/PositiveDefiniteMatrix.html.

[148] Jeevani Wickramaratna, Sean B. Holden, and Bernard F. Buxton. Performance degradation in boosting. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2096)*, pages 11–21, Cambridge, UK, June 2001. Springer.

[149] Terry Windeatt and Fabio Roli, editors. *Multiple Classifier Systems, 4th International Workshop, MCS 2003, Guilford, UK, June 11-13, 2003, Proceedings*, volume 2709 of *Lecture Notes in Computer Science*. Springer, 2003.

[150] K. Woods, W.P. Kegelmeyer, and K. Bowyer. Combination of multiple classiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:405–410, 1997.

[151] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, July 1999.

[152] Xin Yao, Manfred Fischer, and Gavin Brown. Neural network ensembles and their application to traffic flow prediction in telecommunications networks. In *Proceedings of International Joint Conference on Neural Networks*, pages 693–698. IEEE Press, 2001. Washington DC.

[153] Xin Yao and Yong Liu. Making use of population information in evolutionary artificial neural networks. In *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, volume 28, pages 417–425. IEEE Press, June 1998.

[154] Xin Yao and Yong Liu. Neural networks for breast cancer diagnosis. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 1760–1767. IEEE Press, July 1999.

[155] W. Yates and D. Partridge. Use of methodological diversity to improve neural network generalization. *Neural Computing and Applications*, 4(2):114–128, 1996.

[156] P Young. Condorcet's theory of voting. *American Political Science Review*, 82(1231-1244), 1988.

[157] Gabriele Zenobi and Pádraig Cunningham. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. *Lecture Notes in Computer Science*, 2167:576–587, 2001.

[158] Xiao-Hua Zhou, Donna K. McClish, and Nancy A. Obuchowski. *Statistical Methods in Diagnostic Medicine*. WileyEurope, July 2002. ISBN: 0-471-34772-8.

[159] Z.H. Zhou, J.Wu, W.Tang, and Z.Q. Chen. Selectively ensembling neural classifiers. In *International Joint Conference on Neural Networks*, volume 2, 2002.