Journal of Machine Learning Research 6 (2005) 1621–1650 Submitted 4/04; Published 9/05

Managing Diversity in Regression Ensembles

Gavin Brown G.BROWN@CS.BHAM.AC.UK
Jeremy L. Wyatt J.L.WYATT@CS.BHAM.AC.UK
Peter Tiňo P.TINO@CS.BHAM.AC.UK
School of Computer Science
University of Birmingham
Birmingham, UK, B15 2TT

Editor: Yoshua Bengio

Abstract

Ensembles are a widely used and effective technique in machine learning—their success is commonly attributed to the degree of disagreement, or 'diversity', within the ensemble. For ensembles where the individual estimators output crisp class labels, this 'diversity' is not well understood and remains an open research issue. For ensembles of regression estimators, the diversity can be exactly formulated in terms of the covariance between individual estimator outputs, and the optimum level is expressed in terms of a *bias-variance-covariance* trade-off. Despite this, most approaches to learning ensembles use heuristics to encourage the right degree of diversity. In this work we show how to explicitly control diversity through the error function. The first contribution of this paper is to show that *by taking the combination mechanism for the ensemble into account we can derive an error function for each individual that balances ensemble diversity with individual accuracy*. We show the relationship between this error function and an existing algorithm called *negative correlation learning*, which uses a heuristic penalty term added to the mean squared error function. It is demonstrated that these methods control the bias-variance-covariance trade-off systematically, and can be utilised with any estimator capable of minimising a quadratic error function, for example MLPs, or RBF networks. As a second contribution, we derive a strict upper bound on the coefficient of the penalty term, which holds for any estimator that can be cast in a generalised linear regression framework, with mild assumptions on the basis functions. Finally we present the results of an empirical study, showing significant improvements over simple ensemble learning, and finding that this technique is competitive with a variety of methods, including boosting, bagging, mixtures of experts, and Gaussian processes, on a number of tasks.

Keywords: ensemble, diversity, regression estimators, neural networks, hessian matrix, negative correlation learning

1. Introduction

The last decade has seen a frenzy of work in so-called *ensemble learning systems*. These are groups of machine learning systems where each learner provides an estimate of a target variable; these estimates are combined in some fashion, hopefully reducing the generalisation error compared to a single learner. The target can be categorical (classification ensembles) or continuous (regression ensembles). The multiple estimates are integrated via a combination function, commonly *majority voting* for classification and a *linear combination* for regression. It is well appreciated in both cases that the individual estimators should exhibit different patterns of generalisation—the very simple intuitive explanation is that a million identical estimators are obviously no better than a single

©2005 Gavin Brown, Jeremy L. Wyatt and Peter Tiňo.

estimator of the same form. Much research has gone into how to encourage this error "diversity"—most commonly manipulating the training data, providing each learner with a different subset of patterns or features (see Brown et al. (2005a) for a recent survey). The main point to note here is that when our estimators output crisp class labels, there is no agreed definition of diversity, and it remains an open research question (Kuncheva and Whitaker (2003)). The problem is somewhat easier if we have estimators that give posterior probabilities, in which case the effect of estimator correlations on classification error rate has been investigated by Tumer and Ghosh (1995) and Fumera and Roli (2003), though there remain several open questions on this topic.

A commonly overlooked point for regression ensembles is that this "diversity" can be explicitly quantified and measured. The *bias-variance-covariance* decomposition from Ueda and Nakano (1996) breaks the mean squared error (MSE) into three components. Quite simply, whereas in a single regression estimator we have the well known bias-variance *two-way* trade-off, in an ensemble of regressors we have the bias-variance-covariance *three-way* trade-off. The optimum "diversity" is that which optimally balances the components to reduce the overall MSE. In this article we focus on *negative correlation (NC) learning*, a successful neural network ensemble learning technique developed in the evolutionary computation literature (Liu (1998)). In a statistical framework, we show that NC uses a penalty coefficient to *explicitly* alter the emphasis on the variance and covariance portions of the MSE. Setting a zero coefficient corresponds to independently training the estimators; a higher coefficient introduces more emphasis on covariance, and at a particular value it corresponds to treating the entire ensemble as a single learning unit. This is an explicit *management* of the ensemble diversity. We will describe how the ensemble error gradient can be broken into a number of individually understandable components, and that NC exploits this to blend smoothly between a group of independent learners and a single large learner, finding the optimal point *between the two*. We will prove an upper bound on the penalty coefficient, provide guidance on how to set it optimally, and show empirical support that this guidance is useful. The NC framework is *applicable to any nonlinear regression estimator* minimising the MSE; we show examples using multi-layer perceptrons and radial basis function networks as the base estimators.

The structure of this article is as follows. We begin in Section 2 with a summary of the underlying theory of regression ensemble learning, describing why there exists a trade-off between ensemble diversity and individual estimator accuracy. We then consider in Section 3 how we might derive an error function that is capable of optimising this trade-off *explicitly*. We do this and note that it can be shown as equivalent to an existing heuristic technique, *negative correlation (NC) learning*. We continue in Section 4 with an introduction to NC learning, summarising the assumptions and properties as published in the original work. Here we provide a statistical interpretation of NC, and derive a strict upper bound on its penalty coefficient—we empirically validate this bound in Sections 5 and 6. Finally in Section 7 we summarise the implications of this work in a broad context.

2. Ensemble Learning for Regression

In this section we review the bias-variance decomposition (Geman et al. (1992)), using it as a vehicle to introduce our notation; we then show how this decomposition naturally extends to a *bias-variance-covariance* decomposition (Ueda and Nakano (1996)) when using an ensemble of regression estimators.

2.1 The Bias-Variance Decomposition

We have a data set of input vectors and output scalars, $z = \{(\mathbf{x}_1, t_1), ..., (\mathbf{x}_N, t_N)\}$, with each element drawn from a random variable Z defined over an unknown distribution $p(\mathbf{x}, t)$. It should be noted that for brevity, and without loss of generality, we have assumed a noise level of zero in the data.[1] The learning problem is to use the set z to approximate the correct mapping from input to output. For this purpose we use a parameterized estimator f, whose set of parameters $\mathbf{w}$ determine how well it approximates the mapping. We would like to find the set of parameters $\mathbf{w}$ that minimise the expected mean squared error,

$$e(f) = \int (f(\mathbf{x};\mathbf{w}) - t)^2 p(\mathbf{x}, t) d(\mathbf{x}, t). \tag{1}$$

Unfortunately we do not have access to the true distribution $p(\mathbf{x}, t)$, so we approximate this integral with a summation over the data set z,

$$e(f) \approx \frac{1}{N} \sum_{n=1}^{N} (f(\mathbf{x}_n;\mathbf{w}) - t_n)^2 \quad , \quad (\mathbf{x}_n, t_n) \in z. \tag{2}$$

We do not necessarily want a set of parameters $\mathbf{w}$ that give us zero error on z; this is because z is only a *sample* from the true distribution, and if we tune $\mathbf{w}$ precisely to z then the estimator f may not perform well on future data (we overfitted). However, if we do not tune $\mathbf{w}$ just *enough*, then we may again not perform well in the future (we underfitted). This is explicitly formulated in the *bias-variance* decomposition (Geman et al. (1992)). Note that from this point forward, in place of the integral notation in Equation (1), we use the shorthand expectation operator $E\{\cdot\}$; additionally we will omit the input and parameter vectors, so where it is unambiguous, instead of $f(\mathbf{x};\mathbf{w})$, we write simply f. The bias-variance decomposition is

$$\begin{aligned} E\{(f-t)^2\} &= (E\{f\} - t)^2 + E\{(f - E\{f\})^2\} \\ &= \mathit{bias}(f)^2 + \mathit{variance}(f). \end{aligned} \tag{3}$$

The decomposition is a property of the *generalisation* error; these two components have to be balanced against each other for best performance. Now let us imagine that instead of a single estimator f, we have a collection of them: $f_1, ..., f_M$, each f_i has its own parameter vector $\mathbf{w}_i$, and M is the total number of estimators. We then train each individual f_i separately, using Equation (2) as the error function; once this is accomplished, the outputs of the individuals are *combined* to give the *ensemble output* for any new datapoint $\mathbf{x}$. The simplest possible combination mechanism is to take a uniformly weighted average, so the output of the ensemble is

$$\bar{f}(\mathbf{x};\mathbf{w}_1, ..., \mathbf{w}_M) = \frac{1}{M} \sum_{i=1}^{M} f_i(\mathbf{x};\mathbf{w}_i). \tag{4}$$

The ensemble $\bar{f}$ can obviously be seen as an estimator in its own right; it will therefore have a bias-variance decomposition; However it transpires that, for this class of estimator, it can be extended to a bias-variance-*covariance* decomposition.

1. In the case of a non-zero noise component, t in the decomposition would be replaced by its expected value $E_T\{t\}$, and a constant (irreducible) term σ^2 would be added, representing the variance of the noise.

2.2 The Bias-Variance-Covariance Decomposition

Treating the ensemble as a single learning unit, its bias-variance decomposition can be formulated as

$$\begin{aligned} E\{(\bar{f}-t)^2\} &= (E\{\bar{f}\}-t)^2 + E\{(\bar{f}-E\{\bar{f}\})^2\} \\ &= bias(\bar{f})^2 + variance(\bar{f}). \end{aligned} \tag{5}$$

We will now consider how the bias-variance decomposition for an ensemble can be extended (Ueda and Nakano (1996)).[2] From this point forward, it should be noted that the expectation operator is subtly different to that in the decomposition for a single estimator. We redefine our random variable Z as a set $Z = (Z_1, ..., Z_M)$, so the ith estimator is trained with a training set z_i drawn from its own random variable Z_i. It should be noted that Z_i potentially may be identical for all i, or not. If the training data is identical for two machines i and j, it does not imply that the expected values $E\{f_i\}$ and $E\{f_j\}$ are equal, since other differences may be present between machines i and j, i.e. in the training procedures, or the models. Finally, we note 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 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,

$$\overline{bias} = \frac{1}{M}\sum_i (E\{f_i\} - t). \tag{6}$$

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

$$\overline{var} = \frac{1}{M}\sum_i E\{(f_i - E\{f_i\})^2\}. \tag{7}$$

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

$$\overline{covar} = \frac{1}{M(M-1)}\sum_i \sum_{j\neq i} E\{(f_i - E\{f_i\})(f_j - E\{f_j\})\}. \tag{8}$$

We then have

$$E\{(\bar{f}-t)^2\} = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1 - \frac{1}{M}\right)\overline{covar}. \tag{9}$$

What does this decomposition tell us? It illustrates that in addition to the bias and variance of the individual estimators, the generalisation error of an ensemble also depends on the *covariance* between the individuals. This raises the interesting issue of why we should ever train ensemble members separately; why shouldn't we try to find some way to capture the effect of the covariance in the error function? Given the decomposition (9), it is not immediately obvious what form this should take—this will be our next topic for consideration.

2. It is interesting to note that this was the first appearance of the decomposition only for the ML literature—in fact an equivalent decomposition can be found in Markowitz (1952), which was instrumental for modern financial portfolio theory, and subsequently won the 1990 Nobel Prize for Economics.

3. How Can We Optimise Diversity with an Error Function?

For a single regression estimator, generalisation error is determined by a two-way *bias-variance* trade-off; for an ensemble of regression estimators, the 'diversity' issue is simply a *three-way* bias-variance-*covariance* trade-off. We know how to quantify diversity, but we have not yet considered how to achieve it and balance it against individual accuracy—the fundamental issue of ensemble learning. The decompositions we have considered so far consist of integrals over *all possible data sets* of a fixed size—we require a computable approximation to these in order to minimise an error function on a limited data set. It turns out that another decomposition in the literature, significantly more well-known, provides the missing link. We will review this decomposition and its relation to the ones we have already considered, then show how we can use it to train an ensemble whilst controlling the bias-variance-covariance trade-off.

3.1 The Ambiguity Decompositon

Krogh and Vedelsby (1995) showed that *at a single arbitrary datapoint, the quadratic error of the ensemble estimator is guaranteed to be less than or equal to the weighted average quadratic error of the component estimators*,

$$(f_{ens}-t)^2 = \sum_i c_i (f_i-t)^2 - \sum_i c_i (f_i - f_{ens})^2. \tag{10}$$

where t is the target value of an arbitrary datapoint, $\sum_i c_i = 1$, $c_i \geq 0$, and f_{ens} is the convex combination of the M component estimators $f_{ens} = \sum_{i=1}^{M} c_i f_i$. Preceding the bias-variance-covariance decomposition, this was a very encouraging result for ensemble research, providing a very simple expression for the effect of error correlations in an ensemble. The decomposition is made up of two terms. The first, $\sum_i c_i (f_i - t)^2$, is the weighted average error of the individuals. The second, $\sum_i c_i (f_i - f_{ens})^2$ is referred to as the *Ambiguity*, measuring the amount of variability among the ensemble member answers for this particular $(\mathbf{x}, t)$ pair. The trade-off between these two determines how well the ensemble performs at this datapoint.

We have now seen two decompositions, Equation (9) and Equation (10), expressing the effect of correlations on ensemble error in two different ways. It is therefore sensible to ask what the relationship is between these two. The very similar structure of the two decompositions (5) and (10) is no coincidence; the proofs are virtually identical (Brown et al. (2005a)); see also Hansen (2000) for an alternative treatment of this relationship. Assuming a uniform weighting, we substitute the right hand side of equation (10) into the left hand side of equation (9), giving us

$$E\{\frac{1}{M}\sum_i (f_i - t)^2 - \frac{1}{M}\sum_i (f_i - \bar{f})^2\} = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1 - \frac{1}{M}\right)\overline{covar}. \tag{11}$$

What portions of the bias-variance-covariance decomposition correspond to the Ambiguity term? After some manipulations (see Appendix B for details) we can show

$$E\{\frac{1}{M}\sum_i (f_i - t)^2\} = \overline{bias}^2 + \Omega \tag{12}$$

$$E\{\frac{1}{M}\sum_i (f_i - \bar{f})^2\} = \Omega - \left[\frac{1}{M}\overline{var} + \left(1 - \frac{1}{M}\right)\overline{covar}\right]. \tag{13}$$

where Ω is the interaction between the two sides,

$$\Omega \quad = \quad \overline{var} + \frac{1}{M}\sum_i (E\{f_i\} - E\{\bar{f}\})^2. \tag{14}$$

Since the Ω is present in both sides, when we combine them by subtracting the Ambiguity in equation (13), from the average MSE in equation (12), the Ωs cancel out, and we get the original bias-variance-covariance decomposition back, as in the RHS of equation (11). This Ω term is the average variance of the estimators, plus the average squared deviation of the expectations of the individuals from the expectation of the ensemble. The fact that the Ω term exists illustrates again that we cannot simply maximise diversity without affecting the other parts of the error—in effect, this interaction *quantifies* the diversity trade-off for regression ensembles.

3.2 Using the Decompositions to Optimise Diversity

In a simple ensemble, the norm is to train learners separately—the *i*th member of the ensemble would have the error function[3]

$$e_i = \frac{1}{2}(f_i - t)^2. \tag{15}$$

In light of the decompositions we have seen, this is rather odd. Why wouldn't we want to directly minimise the *full* ensemble error?

$$e_{ens} = \frac{1}{2}(\bar{f} - t)^2 = \frac{1}{M}\sum_i \frac{1}{2}(f_i - t)^2 - \frac{1}{M}\sum_i \frac{1}{2}(f_i - \bar{f})^2. \tag{16}$$

One easy answer to this is that we are adopting the "division of labor" approach, simplifying the learning problem by breaking it into M smaller problems. However, according to Equation (11), this error function should account for the bias, the variance, and critically also the covariance of the ensemble. The point to remember is that these components should be *balanced* against each another. Given the relationship shown in Equation (12) and Equation (13), we could imagine a "diversity-encouraging" error function of the form

$$e_i^{div} = \frac{1}{M}\sum_i \frac{1}{2}(f_i - t)^2 - \kappa\frac{1}{M}\sum_i \frac{1}{2}(f_i - \bar{f})^2. \tag{17}$$

where κ is a scaling coefficient in $[0,1]$ and allows us to vary the emphasis on the covariance component. If we adopt a gradient descent procedure for training, we note

$$\frac{\partial e_i^{div}}{\partial f_i} = \frac{1}{M}\Big[(f_i - t) - \kappa(f_i - \bar{f})\Big]. \tag{18}$$

When $\kappa = 0$ here, the gradient of our error function is proportional to the gradient of the error of a single learner, Equation (15). At the other extreme, when $\kappa = 1$, the f_i terms in Equation (18) cancel out, and we have the gradient of the entire ensemble as a single unit,

$$\kappa = 0 \quad , \quad \frac{\partial e_i^{div}}{\partial f_i} \quad = \quad \frac{1}{M}\Big[(f_i - t)\Big] \quad = \frac{1}{M}\frac{\partial e_i}{\partial f_i} \tag{19}$$

$$\kappa = 1 \quad , \quad \frac{\partial e_i^{div}}{\partial f_i} \quad = \quad \frac{1}{M}\Big[(\bar{f} - t)\Big] \quad = \frac{\partial e_{ens}}{\partial f_i}. \tag{20}$$

3. As we will shortly be using a gradient descent procedure, by convention with the existing literature we multiply by $\frac{1}{2}$.

By scaling the κ term we would be able to vary smoothly between the two extremes of training learners separately and training the ensemble as a single unit. Further analysis of these gradient components is provided in Appendix C. Using this scaling parameter corresponds to explicitly varying our emphasis on minimising the covariance term within the ensemble MSE, balancing it against our emphasis on the bias and variance terms; hence we are explicitly *managing* the bias-variance-covariance trade-off. The reader may now justifiably expect an empirical investigation of this error function; however, it conveniently transpires that an existing heuristic method in the literature (derived independently of the observations above) can be shown to be equivalent to this, and has undergone extensive empirical tests showing its utility in a number of domains. The theoretical results we have derived in this section form a solid foundation to explain the success of this technique and link it to others in the literature. We will now consider this, *Negative Correlation Learning*, and show precisely how it relates to the derivations we have provided in this section.

4. Negative Correlation Learning

Negative correlation (NC) learning (Liu (1998)) is a 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 (Yao et al. (2001)), classification problems (McKay and Abbass (2001)), and time-series prediction (Liu (1998)). It has consistently demonstrated significant performance improvements over a simple ensemble system, showing very competitive results with other techniques like mixtures of experts, bagging, and boosting (Liu and Yao (1997); McKay and Abbass (2001)). Though empirical successes have been found with classification problems, it should be noted that the discussion here concerns only the regression case.

4.1 The History

The fact that correlations between ensemble members affects performance has been known for a long time. The first such reference to appear in the machine learning literature was Perrone (1993), showing that we obtain a $\frac{1}{M}$ variance reduction if correlation between learners is zero. The first reference in the literature to explicitly use this idea in a learning algorithm was Rosen (1996), who trained networks sequentially using a penalty and scaling coefficient λ added to the error term,

$$e_i = \frac{1}{2}(f_i - t)^2 + \lambda p_i \tag{21}$$

$$p_i = (f_i - t)\sum_{j=1}^{i-1}(f_j - t). \tag{22}$$

Attempting to extend this work, Liu and Yao (1997) trained the networks in parallel, and used a number of alternative penalty terms[4] including one where the t is replaced by $\bar{f}$,

$$p_i = (f_i - t)\sum_{j\neq i}(f_j - t) \tag{23}$$

$$p_i = (f_i - \bar{f})\sum_{j\neq i}(f_j - \bar{f}). \tag{24}$$

4. A companion work to this article, Brown et al. (2005b), gives a similar analysis to penalty (23).

The λ parameter is problem-dependent, controlling the trade-off between the objective and penalty terms during the gradient descent training procedure. Figure 1 shows NC using backpropagation to update the network weights. A point to note here is that the authors calculate the gradient using the assumption *"that the output of the ensemble $\bar{f}$ has constant value with respect to f_i"* (Liu, 1998, p.29)., i.e.

$$\frac{\partial \bar{f}}{\partial f_i} = 0. \tag{25}$$

Using this, and the penalty Equation (24), the following gradient was derived,

$$e_i = \frac{1}{2}(f_i - t)^2 + \lambda(f_i - \bar{f}) \sum_{j \neq i} (f_j - \bar{f}) \tag{26}$$

$$\frac{\partial e_i}{\partial f_i} = (f_i - t) + \lambda \sum_{j \neq i} (f_j - \bar{f}). \tag{27}$$

This is clearly an incorrect assumption—in the next section we will examine the reasoning behind it,

1. Let M be the final number of predictors required.
2. Take a training set $z = \{(\mathbf{x}_1, t_1), ..., (\mathbf{x}_N, t_N)\}$.
3. For each training pattern in z 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 w in network i, using a learning rate α (set as 0.1 in our experiments), and:

$$\Delta w = -\alpha \left[(f_i(\mathbf{x}_n) - t_n) - \lambda (f_i(\mathbf{x}_n) - \bar{f}) \right] \cdot \frac{\partial f_i}{\partial w}$$

4. Repeat from step 3 for a desired number of iterations.

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 1: Pseudocode for negative correlation learning. Note the relationship between the λ term and the γ term in Equation (30).

the implications it brings, and show how NC relates to the error decompositions we have discussed so far.

4.2 A Theoretical Grounding for NC Learning

We would like to provide a rigorous foundation for the NC method, so it seems sensible to observe what happens when we *remove* the assumption of constant $\bar{f}$. We now introduce a term γ in place of λ, to indicate when we perform the gradient calculations *without the assumption*. Re-deriving the gradient, we have

$$e_i = \frac{1}{2}(f_i - t)^2 + \gamma(f_i - \bar{f})\sum_{j \neq i}(f_j - \bar{f}) \tag{28}$$

$$\frac{\partial e_i}{\partial f_i} = (f_i - t) - \gamma\Big[2(1 - \frac{1}{M})(f_i - \bar{f})\Big]. \tag{29}$$

We understand now that λ in fact has the deterministic component $2(1 - \frac{1}{M})$; to avoid confusion we now refer to the parameters in the following context,

$$\lambda = 2\gamma(1 - \frac{1}{M}). \tag{30}$$

where γ is still a problem-dependent scaling parameter. According to communications with the original authors, the assumption was introduced for two reasons. Firstly because the term $2(1 - \frac{1}{M})$ is a constant for any fixed ensemble of size M, so can be precalculated for efficiency. Secondly, it allowed the appealing property that when $\lambda = 1$, the gradient in Equation (27) reduces

$$\begin{aligned} \frac{\partial e_i}{\partial f_i} &= (f_i - t) + \lambda\sum_{j \neq i}(f_j - \bar{f}) \\ &= (f_i - t) - \lambda(f_i - \bar{f}) \\ &= (\bar{f} - t) \\ &= M \cdot \frac{\partial e_{ens}}{\partial f_i}. \end{aligned} \tag{31}$$

Here it can be seen that the identity $\sum_{j \neq i}(f_j - \bar{f}) = -(f_i - \bar{f})$ was used—the sum of deviations around a mean is equal to zero. However, for this to hold we have to now *violate* the constant $\bar{f}$ assumption, as the sum of deviations around a constant is *not* equal to zero. The reader will see an immediate similarity in (31) to the observations we have made in the previous section, specifically equations (18), (19), and (20). It emerges that by introducing the assumption, and subsequently violating it, the NC gradient becomes proportional to the gradient of the "diversity-encouraging" error function (18) suggested earlier, where we use λ in place of κ,

$$\frac{\partial e_i}{\partial f_i} = (f_i - t) - \lambda(f_i - \bar{f}) = M \cdot \frac{\partial e_i^{div}}{\partial f_i}. \tag{32}$$

¿From the observations we have made here, the connection between NC and the Bias-Variance-Covariance decomposition should be apparent. By introducing the assumption (25), NC was inadvertently provided with the missing gradient components that correspond to the variance and covariance terms within the ensemble MSE. It can therefore be concluded that NC succeeds because it trains the individual networks with error functions which more closely approximate the individual's contribution to ensemble error, than that used by simple ensemble learning. Using the

penalty coefficient, we then balance the trade-off between those individual errors and the ensemble covariance. The relationship to the Ambiguity decomposition is made even more apparent by noting that the penalty term can be rearranged,

$$p_i = (f_i - \bar{f}) \sum_{j \neq i} (f_j - \bar{f}) = -(f_i - \bar{f})^2. \tag{33}$$

This leads us to a restatement of the NC error function,

$$e_i = \frac{1}{2}(f_i - t)^2 - \gamma (f_i - \bar{f})^2. \tag{34}$$

Remembering the breakdown of the ensemble error from earlier,

$$e_{ens} = \frac{1}{M} \sum_i \left[\frac{1}{2}(f_i - t)^2 - \frac{1}{2}(f_i - \bar{f})^2 \right], \tag{35}$$

we see that the MSE of an ensemble can be decomposed into a weighted summation, where the i^{th} term is the backpropagation error function plus the NC-learning penalty function, with the γ parameter set at 0.5. Here we note an important point, that there are additional effects that f_i has on Equation (35), that are *not contained* in Equation (34). This is via the $\bar{f}$ term, which obviously depends on f_i, and can be found in each component of the summation in Equation (35). Therefore, simply setting $\gamma = 0.5$ would mean we are not taking account of these effects, and setting $\gamma > 0.5$ allows us to include them and find the appropriate problem-dependent balance for best generalisation. These observations are supported by further gradient analysis in Appendix C.

To summarise, in this section we have shown that there exist two quite different error functions, which yield gradients (18) and (29) differing only in a scalar constant. Each incorporates sufficient information to allow the individual learners to optimise the bias-variance-covariance trade-off. We now understand how NC balances accuracy against diversity; however, we do not yet understand what the *correct* balance is, i.e. how do we set the penalty coefficient? We consider this problem in the next section.

4.3 Understanding and Defining Bounds on the Penalty Coefficient

The original work on NC (Liu and Yao (1997)) showed that a λ value greater than zero can encourage a decrease in covariance, however it is also observed 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 behaviour, and as such we do not yet have a clear picture of the exact dynamics of the parameter. It was stated that the bounds of λ should be $[0,1]$, based on the following calculation,

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

It is stated: *"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"* (Liu, 1998, p.29). In practice this bound seemed to be applicable; however,

the justification is questionable, and again here we see the assumption of constant $\bar{f}$ is violated—if constant $\bar{f}$ is assumed, then the deviations around $\bar{f}$ result cannot be used. In this section we provide more concrete theoretical evidence for an upper bound.

The NC penalty term 'warps' the error landscape of the network, making the global minimum hopefully easier to locate. However, if the landscape is warped *too* much, it could eliminate any useful gradient information. This state is indicated by the positive-definiteness (PD) of the Hessian matrix. If the Hessian matrix, evaluated at a given point, is non-PD, 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. We acknowledge an important point here, that the state of the Hessian during training says *nothing* about the *generalisation error*. We simply note that if we have a non-PD Hessian during the training, there will be no minimum to converge to *on the datapoint at which it was evaluated*, in which case training can only cause weight divergence. We would therefore like to know conditions under which the Hessian will be non-PD.

If the Hessian matrix is positive definite, then all elements on the leading diagonal are positive-valued; therefore if any element on that diagonal is zero or less, the entire matrix *cannot* be positive definite. Assume we have an estimator that is a linear combination of a number of nonlinear functions ϕ, so

$$f_i = \sum_{k=1}^{K} w_{ki}\phi_{ki}. \tag{36}$$

Examples of estimators in this class are Multi-Layer Perceptrons using linear output nodes, Polynomial Neural Networks, and Radial Basis Functions. Now, for an arbitrary Hessian diagonal element corresponding to the qth weight in the output layer of the ith network, w_{qi}, we can show (derivation given in Appendix A) that

$$\frac{\partial^2 e_i}{\partial {w_{qi}}^2} = \left[1-\lambda(1-\frac{1}{M})\right]{\phi_{qi}}^2. \tag{37}$$

where in the case of RBF networks, ${\phi_{qi}}^2$ is the squared output of the qth basis function in the ith network. If this element, Equation (37), equates to zero or less, the entire Hessian matrix *is guaranteed to be non-positive definite*. Therefore we would like the following inequality to hold,

$$\begin{aligned}
0 &< \left[1-\lambda(1-\frac{1}{M})\right]{\phi_{qi}}^2 \\
0 &< {\phi_{qi}}^2 - \lambda{\phi_{qi}}^2(\frac{M-1}{M}) \\
\lambda{\phi_{qi}}^2(\frac{M-1}{M}) &< {\phi_{qi}}^2 \\
\lambda &< \frac{{\phi_{qi}}^2}{{\phi_{qi}}^2(\frac{M-1}{M})} \\
\lambda &< \frac{M}{M-1}.
\end{aligned} \tag{38}$$

Since the effect of ϕ_{qi} cancels out,[5] we find that this inequality is *independent* of all other network parameters, so it is *a constant for any ensemble architecture using estimators of this form*

5. We note that we assume a basis function $\phi_{qi} \neq 0$, to avoid divide by zero problems—this does not always hold, for example when using hyperbolic tangent activations; however here we assume either sigmoid activation or a Gaussian RBF.

and a simple average combination function. This defines an upper bound for λ and, since we know the relationship between the two strength parameters from Equation (30), we can also show a bound for γ,

$$\lambda_{upper} = \frac{M}{M-1} \qquad \gamma_{upper} = \frac{M^2}{2(M-1)^2}. \tag{39}$$

When λ or γ is varied beyond these upper bounds, the Hessian matrix is guaranteed to be non-positive definite. Figure 2 plots λ_{upper} and the equivalent γ_{upper} for different ensemble sizes. We see that as the size increases, λ_{upper} asymptotes to 1, and γ_{upper} to 0.5. For larger ensembles, e.g. $M > 10$, this therefore lends concrete theoretical evidence to Liu's proposed bound of $\lambda = 1$. However, for small M, the bound shows values larger than $\lambda = 1$ may still retain a positive definite Hessian matrix.

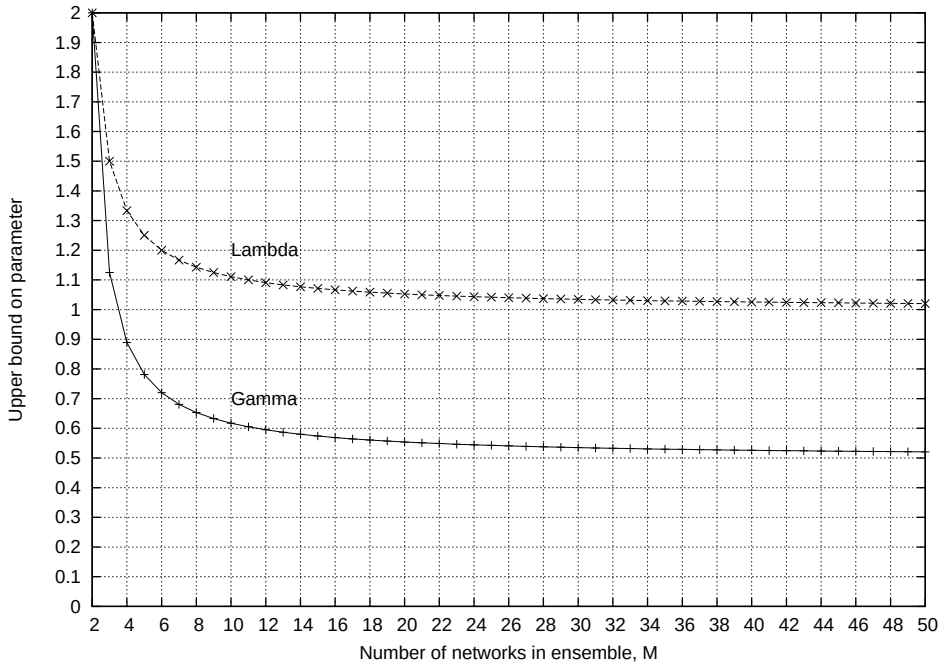

Figure 2: The Upper bound on γ and λ.

Our bound was determined on the premise that the leading diagonal containing negative elements implies a non-PD Hessian matrix. However, it could easily be the case that the leading diagonal is all positive, yet the entire matrix is still non-PD. This implies that the matrix could become non-PD *before* our upper bound is reached. 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?"*. Any such analytical conditions based on the Hessian will almost certainly be input-dependent—the advantage of our bound is that it is a constant for a given ensemble, dependent only on the number of ensemble members. However, the utility of the bound depends entirely on how tight it is—using

a neural network ensemble it would be pointless if the weights diverged significantly *before* the bound is reached. To validate this hypothesis we now engage in empirical testing.

5. Empirically Validating the Proposed Bound

The purpose of this section is to determine how useful our theoretical upper bound can be in practice. We remind the reader again that our bound is not computed in reference to generalisation error, and we now wish to evaluate whether it can be practically useful in this context. When varying γ, if the network weights diverge significantly before the upper bound is reached, then the bound is not tight and therefore of little use. Alternatively, it could simply be that certain ensemble configurations do not show any benefit from using NC, in which case NC *itself* is of no use, and neither is our parameter bound. We now investigate these issues.

5.1 Data Sets

We use the Boston Housing data set, where 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 $[-1,1]$. A five-fold cross validation procedure was used, so keeping 20% of the data as a holdout set, and using the remaining 80% for training and validation. With the Boston data set this equates to 304 for training, 101 for validation, and 101 for testing. The validation data was used to perform early stopping by the following procedure: train while noting the validation error every 50 epochs; if the validation error has risen in comparison to 500 epochs ago, terminate training and reset the weights to the best point within that 500 epoch window (at a resolution of 50 epochs) according to the validation data.

The second data set was generated (Friedman (1991)) by the function

$$h(\mathbf{x}) = 10sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5 + \eta, \tag{40}$$

where $\mathbf{x} = [x_1,..,x_{10}]$ is an input vector whose components are drawn uniformly at random from $[0,1]$, and η is a noise component drawn from $N(0,1)$, i.e. mean zero and variance 1.0. 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. We used a data set of size 1000, using the same five-fold cross validation procedure as described above.

The third data set used was the *LogP* data, recently used in (Tino et al. (2004)). This is a highly nonlinear pharmaceutical data set, where the task is to predict the *partition coefficient* of a chemical compound, allowing one to determine certain uptake properties of the molecule. The data set has 14 continuous input variables, and 1 continuous output variable in the range $[-4.2,+9.9]$. There are 6912 examples, which we used in the same cross-validation procedure as above.

For all three data sets, each ensemble was evaluated over the 5 data folds and over 30 trials of random weights, giving 150 trials for each run.

5.2 When Does the NC Technique Work Well?

Empirical analyses of NC have been shown on several other occasions and on several other data sets (Liu et al. (2000); Liu and Yao (1997); Brown (2004)). The point of this section is to characterise some general conditions of ensemble architecture under which NC seems to succeed in comparison to a simple ensemble.

We train several different ensemble architectures using a range of γ values (at a resolution of 0.05), the optimum γ value was located according to the validation data, and finally evaluated on the testing data. This was compared to using $\gamma = 0$, where it should be remembered that $\gamma = 0$ is exactly equivalent to simple ensemble learning, i.e. training each network independently of the others *without* NC learning. We first varied the number of networks in the ensemble, using a fixed individual network size of 6 hidden nodes. Figure 3 shows results for the Friedman data, figure 4 for Boston, and figure 5 for LogP; 95% confidence intervals are indicated. With the Friedman and Boston data sets, a general trend that can be noted is that larger relative gains seem to be made with larger ensemble sizes.

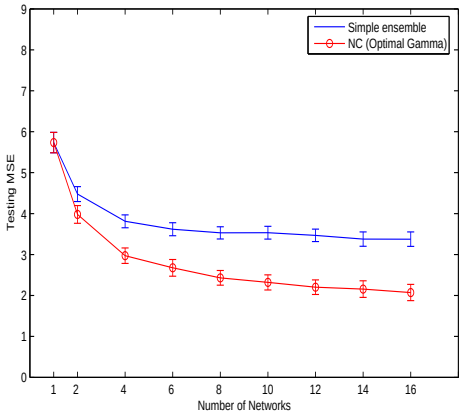

Figure 3: Friedman, $\gamma = 0$ versus optimal γ, 6 hidden nodes per network

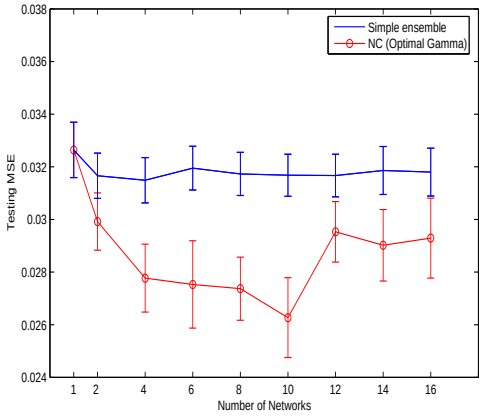

Figure 4: Boston, $\gamma = 0$ versus optimal γ, 6 hidden nodes per network

However, with the LogP data, relative performance does not seem to increase with the size of the ensemble. If we make the component networks much simpler, 2 hidden nodes as in figure 6, we see the same recognisable trends as in the other data sets. The general rule here, supported by previous empirical work on NC, seems to be to use very simple networks—in this case we can see that an ensemble of 16 networks, each with 2 hidden nodes, has equalled the performance of a similarly sized ensemble, using 6 hidden nodes per network.

Figures 7 and 8 show the gains as we vary the *complexity* (i.e. number of hidden nodes per network) of the individual ensemble members. We can note here that the gain from using NC *decreases* as we increase the complexity of the networks. Regarding again figures 3 to 6 these results indicate that NC is of most use when we have large ensembles of relatively low complexity ensemble members. This is emphasized further looking at figure 10, where we see that an ensemble of 6 networks using 2 hidden nodes, and using NC, can equal the performance of the same ensemble using

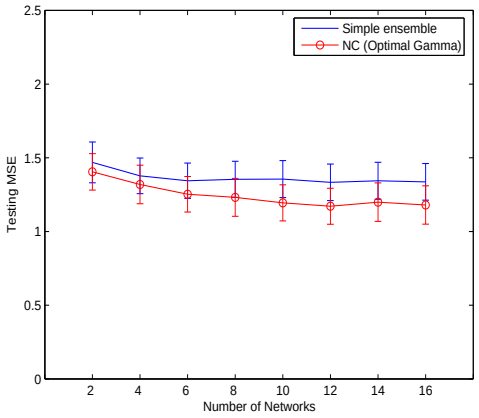

Figure 5: LogP, $\gamma = 0$ versus optimal γ, 6 hidden nodes per network

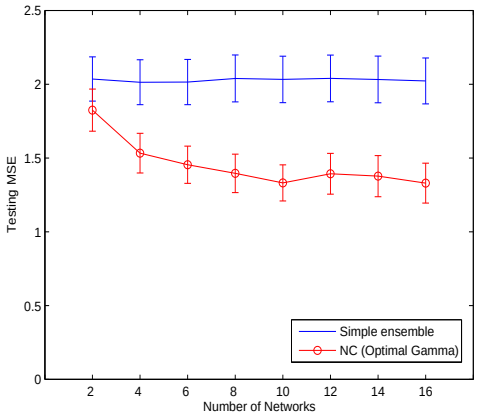

Figure 6: LogP, $\gamma = 0$ versus optimal γ, 2 hidden nodes per network

far more complex networks. Additionally, in these situations, when γ is set optimally, significantly faster convergence and lower generalisation error for a fixed number of epochs were observed.

5.3 How Tight is the Bound?

We now turn to examining the behaviour of the *generalization error* as we move γ toward its upper bound. Figures 11, 13 and 15 show the performance as γ is changed, with several different sizes of ensemble—each network has fixed complexity at 6 hidden nodes. A distinctive pattern is observed: a virtually monotonic decrease in error as we increase γ, up to a particular "threshold", beyond which the error rises rapidly. On closer examination of the networks trained with these high γ values, it was observed that the network weights had diverged to excessively large values. The point at which divergence occurs seems to move downward as we increase the size of the ensemble. Figures 12, 14 and 16 show the behaviour with a fixed ensemble size, $M = 6$, as we vary the individual complexity between 2 and 12 hidden nodes. Here we see a distinction from the results varying ensemble size: the divergence point seems largely unaffected by the complexity of the networks.

Using these results as a guide, we searched the range of γ at the finer resolution of 0.01 to locate the divergence point. Figure 18 shows this, illustrating that divergence seems extremely *invariant* to the choice of data set. We superimpose the predicted upper bound and note that as the number of networks increases, the divergence point and the upper bound both asymptote to 0.5, confirming that our bound is tight. We have also superimposed $\gamma = \frac{M}{2(M-1)}$, corresponding to when $\lambda = 1$. Zooming in on part of the plot allows us to see that the $\lambda = 1$ original bound is obeyed in most instances, but not all. We acknowledge of course that the *exact* location of the divergence point is of little consequence; the real point we wish to locate is the *optimum* γ value, and see if it provides significant improvements relative to other ensemble techniques; we will explore this in the next section.

In conclusion to the 'upper bound' issue, we note that we have provided theoretical evidence that supports that $\lambda = 1$ bound in the case of large M, but the bound remains loose for small M, and

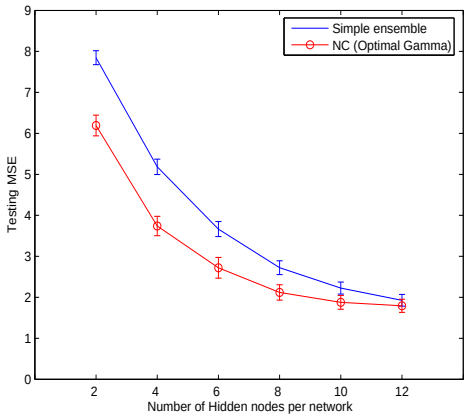

Figure 7: Friedman, $\gamma = 0$ versus optimal γ for 6 networks

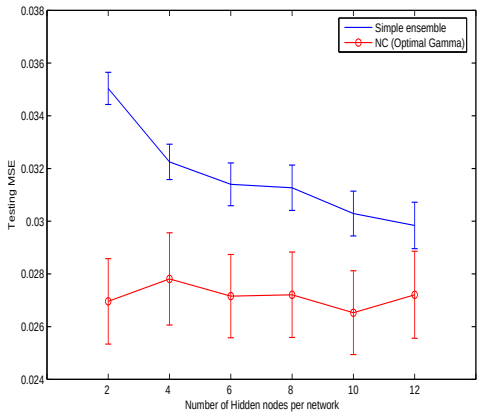

Figure 8: Boston, $\gamma = 0$ versus optimal γ for 6 networks

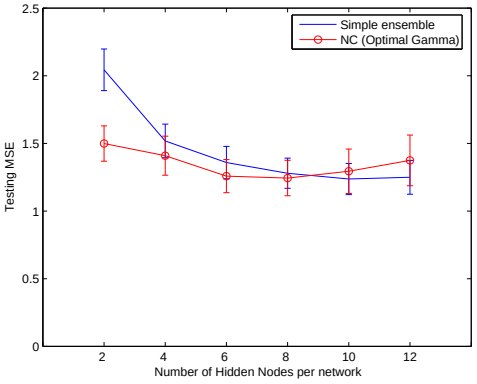

Figure 9: LogP, $\gamma = 0$ versus optimal γ for 6 networks

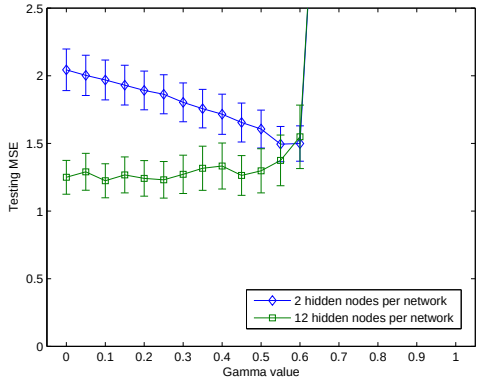

Figure 10: LogP, plotting testing MSE against Gamma for an ensemble of 6 networks

$\lambda = 1$ (or equivalently $\gamma = \frac{M}{2(M-1)}$) seems to be a useful heuristic bound. A possible justification for this is to remember that as we approach $\lambda = 1$, we treat the ensemble more and more as a single learning unit—beyond this we would be introducing a greater emphasis on covariance than is in the overall ensemble objective function; whether this bound can be strictly proved remains an open question.

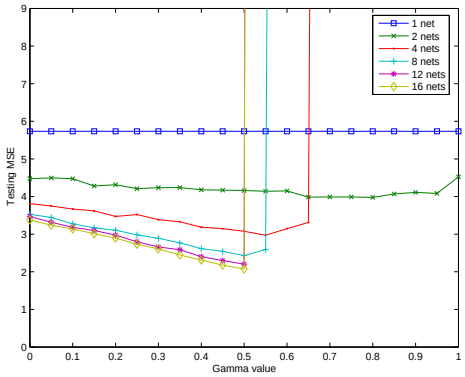

Figure 11: Friedman, varying γ with 6 hidden nodes per network

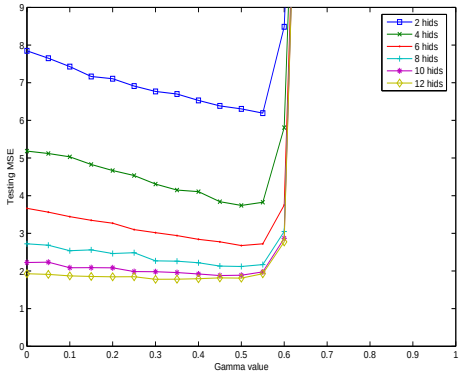

Figure 12: Friedman, varying γ with 6 networks

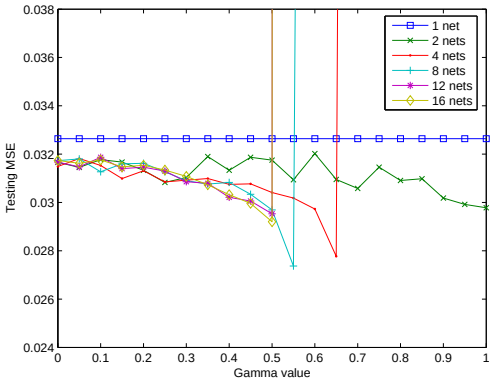

Figure 13: Boston, varying γ with 6 hidden nodes per network

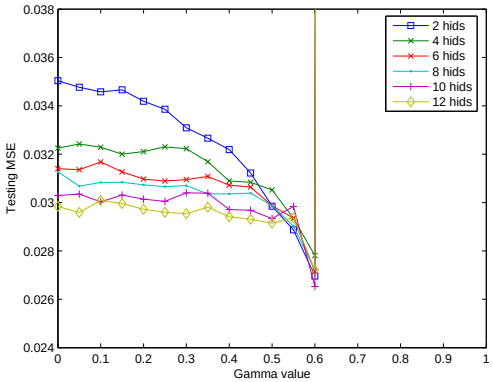

Figure 14: Boston, varying γ with 6 networks

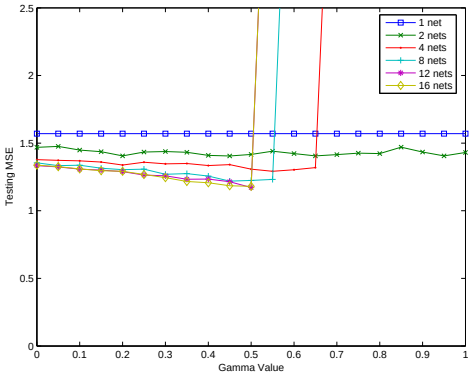

Figure 15: LogP, varying γ with 6 hidden nodes per network

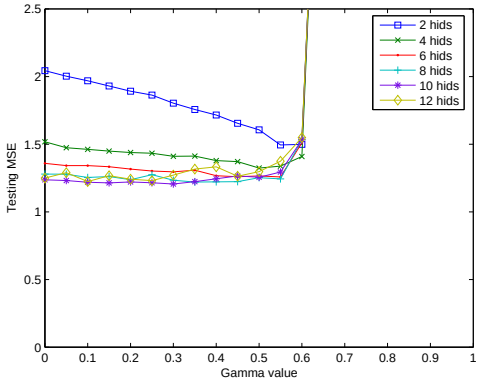

Figure 16: LogP, varying γ with 6 networks

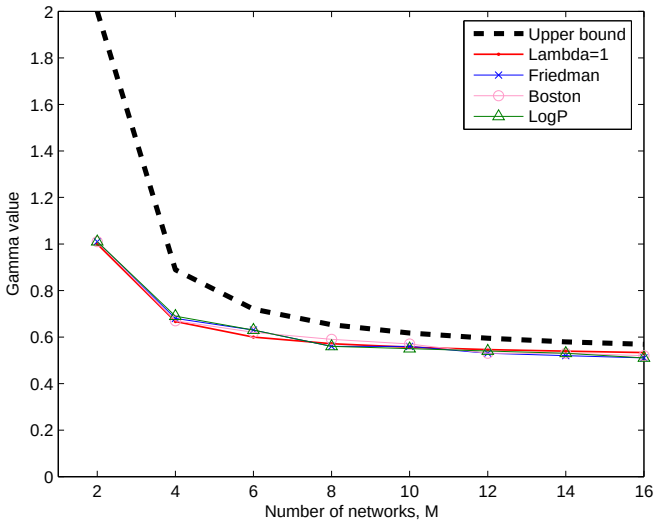

Figure 17: γ-value at which divergence of weights was observed for the data sets.

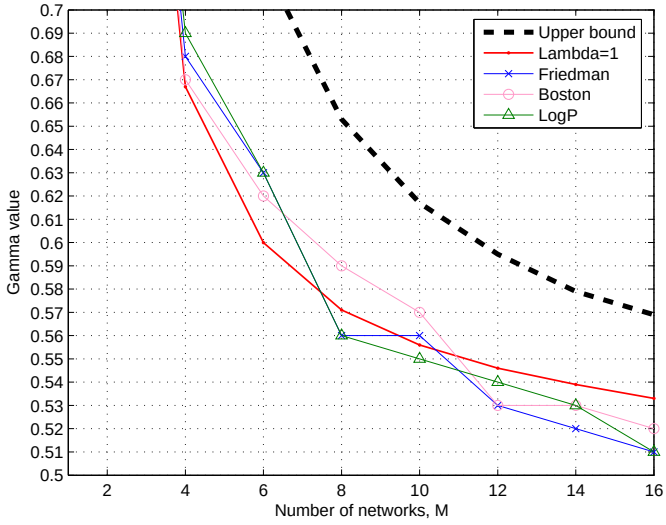

Figure 18: γ-value at which divergence occurred, zoomed in.

6. Further Empirical Comparisons

In this section we will compare the NC framework to other competitive ensemble approaches, using MLPs as the base estimator. Additionally we will illustrate that the NC framework is indeed general, by using RBF regressors, and showing that very similar empirical patterns emerge, obeying our upper bound for the γ parameter.

6.1 Comparing NC to Other Popular Ensemble Approaches

Performing valid empirical comparisons with existing works in the literature is a notoriously difficult task; slight differences in experimental setup can easily invalidate the procedure. In particular, training/testing data must be exactly the same between two systems to be compared. The LogP data has been used in previous work (Tino et al. (2004)) by one of the current authors—we obtained the exact data split used in that work to test NC against their results. Of the 6912 examples, 5530 were used for training, 691 for validation and 691 for testing. We used 12 networks, each with 6 hidden nodes. Using the validation data, the optimum $\gamma = 0.5$ was determined. Results in table 2 show how NC compares with other state-of-the-art techniques; 95% confidence intervals are indicated where available. As an additional useful statistic, Tino et al. (2004) computed the ION (Improvement over Naive) value. This is the percentage improvement relative to a naive predictor (with an MSE of 2.69) which predicts a constant for any input, equal to the mean target value in the training data. The best achieveable improvement in their experiments was 77.7%, the Gaussian Process learner, while here we see NC achieves 78.2%.

System	*Testing MSE (conf)*	ION %
NC, 12 MLPs, $\gamma = 0.5$	$0.5866(\pm 0.0168)$	78.2%
Gaussian Processes	0.601	77.7%
Hierarchical Mixture of Experts	0.658	75.5%
Simple ensemble, 12 MLPs	$0.7692(\pm 0.0154)$	71.4%

Table 1: Comparing NC to other state-of-the-art learning techniques on the *LogP* data.

To further empirically verify NC, we now compare it to two other popular ensemble techniques, Adaboost.R2 and bagging. Figures 19 to 22 show results, again following the empirical procedures described in section 5 - all Boosted and Bagged networks were trained with early stopping. We note that on the Friedman data, NC significantly outperforms both boosting and bagging, increasing its lead as the ensemble size is increased. The Boston data shows that NC is obviously not a panacea technique - boosting and bagging significantly outperform it in this situation. From this and previous experiments with NC, we hypothesize that the noisy nature of the Friedman data is ideally suited to the flexibility allowed by NC's γ parameter, explicitly varying the *fit of the ensemble model* to the data as needed, whereas boosting and bagging do not have this extra free parameter. We note that a full empirical benchmarking of NC and its behaviour with noisy data is underway, but outside the scope of this article.

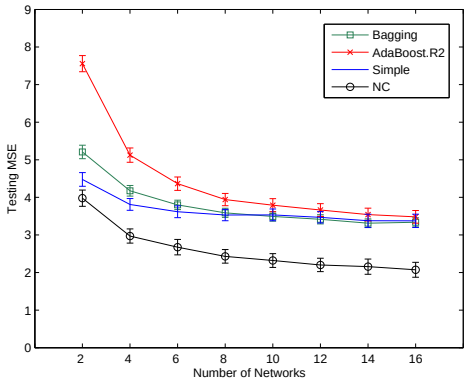

Figure 19: Friedman, varying number of networks (6 hidden nodes in each)

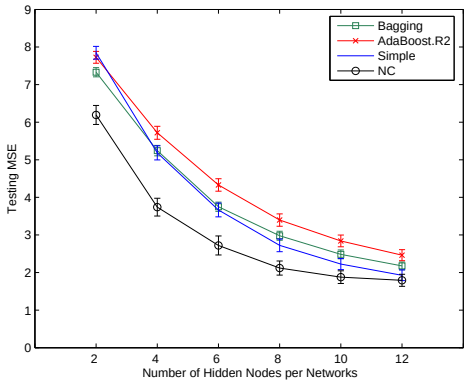

Figure 20: Friedman, varying number of hidden nodes per network (ensemble of 6 networks)

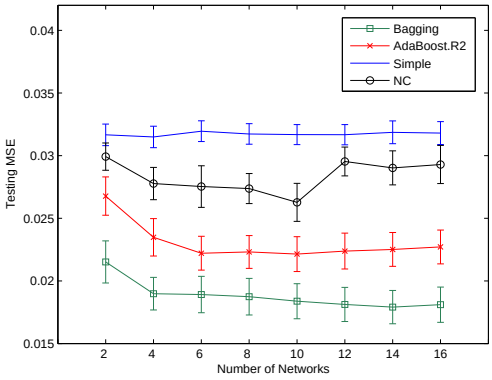

Figure 21: Boston, varying number of networks (6 hidden nodes in each)

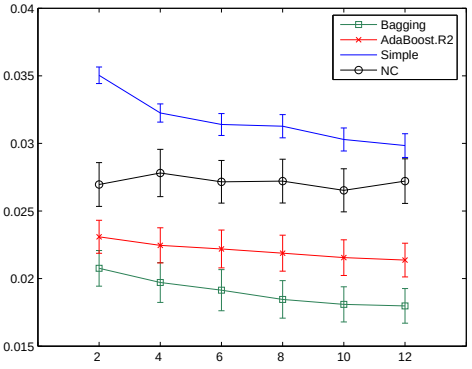

Figure 22: Boston, varying number of hidden nodes per network (ensemble of 6 networks)

6.2 Using NC with an Ensemble of RBF Networks

We now briefly illustrate that NC can be applied to regression estimators of other types, not just multi-layer perceptrons. We use an ensemble of Radial Basis Function networks, using Gaussian basis functions. Centres and widths are initialised randomly, then a full gradient descent is performed on all parameters, using the NC penalty framework as previously described. Table 2 shows that an ensemble of RBF networks each with 50 centres can outperform an MLP ensemble each with 50 sigmoidal hidden nodes, and applying NC to the RBF ensemble allows further gain. Finally in figure 23 we see the effect of varying γ on both the MLP and RBF ensemble. As previously observed, with very complex individuals NC cannot provide further error reduction. Here we note two points. Firstly, though an MLP ensemble cannot benefit, an RBF ensemble of the same size *can*

benefit from NC. Secondly, and most importantly, we see it again obeys our predicted upper bound on γ.

System	*Testing MSE (conf)*
RBF: 5 x 50 basis functions, NC $\gamma = 0.5$	$\mathbf{0.0229}(\pm 0.001)$
RBF: 5 x 50 basis functions	$0.0263(\pm 0.001)$
MLP: 5 x 50 hidden nodes, NC $\gamma = 0.5$	$0.0313(\pm 0.001)$
MLP: 5 x 50 hidden nodes	$0.0319(\pm 0.001)$

Table 2: Using NC with an ensemble of RBF networks on the Boston data set

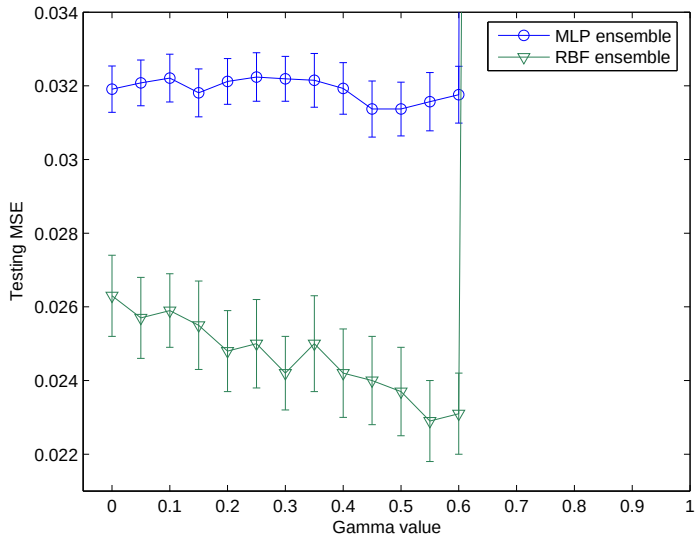

Figure 23: LogP data: The effect of varying the NC γ parameter on an RBF and MLP ensemble of size $M = 5$, noting that our predicted upper bound $\gamma_{upper} = 0.78125$ holds for the RBF ensemble.

7. Conclusions

We have investigated the issue of how to explicitly *manage* the correlations in an ensemble of regression estimators. We made important observations on the relationships between the Ambiguity decomposition (Krogh and Vedelsby (1995)) and the bias-variance-covariance decomposition (Ueda and Nakano (1996)). From this base, we provided a thorough critique of negative correlation (NC) learning (Liu (1998)), a technique that extended from Rosen (1996), and developed in the evolu-

tionary computation literature. We showed that using a penalty term and coefficient, NC *explicitly includes the covariance portion of the ensemble MSE in its error function*. This article has served to illustrate that 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 as an algorithm. The NC framework can be applied to ensembles of *any nonlinear regression estimator* combined in an ensemble $\bar{f}$ of the form

$$\bar{f}(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}). \tag{41}$$

In addition, an upper bound on the strength parameter was shown to apply when each estimator f_i is of the form

$$f_i(\mathbf{x}) = \sum_{k=1}^{K} w_{ki}\phi_{ki}(\mathbf{x}). \tag{42}$$

Examples of estimators in this class are multi-layer perceptrons with linear output nodes, and Radial Basis Function networks, indeed any estimator that can be cast in a generalised linear regression framework. To derive the bound, we observed that positive definiteness of the Hessian matrix can be determined by checking just the first K leading diagonal elements. We verified this bound empirically, and although the bound is tight for larger ensembles, it remains loose for size $M < 10$, and a useful empirical bound of $\gamma = \frac{M}{2(M-1)}$ seems to apply. These results seem to suggest a general set of guidelines for application of the NC framework. The common trend was to see increasing utility of NC with larger ensembles of relatively low individual complexity, with optimum γ tending to 0.5. We therefore recommend a starting point as: ensemble size $M \geq 10$, number of hidden nodes between 2 and 5, and a penalty strength parameter of $\gamma = 0.5$. This will of course be problem dependent, most significantly the number of hidden nodes—what is 'low complexity' for one task will not be for another—but we believe it does provide good general guidance. In addition, it seems sensible from our investigations that some sort of annealing of the parameter during the learning process, from zero up towards the bound, may show further performance benefits.

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 that NC allows a smooth transition of the error gradients between that of a fully parallel ensemble system and a single estimator. This raises a point on the nature of *overfitting* in ensembles. It is well known that overfitting of the individual estimators can be beneficial in an ensemble system (Sollich and Krogh (1996)), but obviously overfitting the entire ensemble as a unit is an undesirable prospect. With this new information about NC, *what* should we overfit?

Appendix A. Calculations Supporting the Strength Parameter Bound

We now present additional calculations supporting the work on the upper bound for the λ and γ parameters, as in Section 4.3. Assuming an estimator which is a linear combination of other functions ϕ, we wish to derive one of the entries in the leading diagonal of the Hessian matrix. The diagonal element corresponding to the qth weight in the ith estimator is $\frac{\partial^2 e_i}{\partial w_{qi}{}^2}$. If this is zero or less, then the Hessian is guaranteed to be non-positive definite, an undesirable prospect. Making use of

the product rule, we have

$$\begin{aligned}\frac{\partial e_i}{\partial w_{qi}} &= \frac{\partial e_i}{\partial f_i}\frac{\partial f_i}{\partial w_{qi}} \\ \frac{\partial^2 e_i}{\partial {w_{qi}}^2} &= \Big[\frac{\partial}{\partial w_{qi}}\frac{\partial e_i}{\partial f_i}\Big]\frac{\partial f_i}{\partial w_{qi}} + \Big[\frac{\partial}{\partial w_{qi}}\frac{\partial f_i}{\partial w_{qi}}\Big]\frac{\partial e_i}{\partial f_i}. \end{aligned} \tag{43}$$

Taking the first term on the right hand side,

$$\begin{aligned}\frac{\partial e_i}{\partial f_i} &= (f_i - t) - \lambda(f_i - \bar{f}) \\ \frac{\partial}{\partial w_{qi}}\frac{\partial e_i}{\partial f_i} &= \phi_{qi} - \lambda(\phi_{qi} - \frac{1}{M}\phi_{qi}) \\ &= \Big(1 - \lambda(1 - \frac{1}{M})\Big)\phi_{qi}. \end{aligned} \tag{44}$$

Now for the second term, remembering eq (42), we have

$$\frac{\partial f_i}{\partial w_{qi}} = \phi_{qi} \tag{45}$$

$$\frac{\partial}{\partial w_{qi}}\frac{\partial f_i}{\partial w_{qi}} = \frac{\partial^2 f_i}{\partial {w_{qi}}^2} = 0. \tag{46}$$

Therefore we simply have

$$\frac{\partial^2 e_i}{\partial {w_{qi}}^2} = \Big[\Big(1 - \lambda(1 - \frac{1}{M})\Big)\phi_{qi}\Big]\phi_{qi} + \Big[0\Big]\frac{\partial e_i}{\partial f_i} \tag{47}$$

$$= \Big(1 - \lambda(1 - \frac{1}{M})\Big){\phi_{qi}}^2. \tag{48}$$

It is interesting to observe that since we have

$$\frac{\partial e_i}{\partial f_i} = (f_i - t) - \lambda(f_i - \bar{f}) \tag{49}$$

$$\frac{\partial^2 e_i}{\partial {f_i}^2} = 1 - \lambda(1 - \frac{1}{M}). \tag{50}$$

then we can see

$$\frac{\partial^2 e_i}{\partial {w_{qi}}^2} = \frac{\partial^2 e_i}{\partial {f_i}^2}{\phi_{qi}}^2. \tag{51}$$

This demonstrates that, since ${\phi_{qi}}^2$ is positive, the sign of the leading diagonal entry $\frac{\partial^2 e_i}{\partial {w_{qi}}^2}$ in the Hessian is decided by the sign of $\frac{\partial^2 e_i}{\partial {f_i}^2}$.

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

$$E\{(\bar{f}-t)^2\} = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1-\frac{1}{M}\right)\overline{covar}. \tag{52}$$

Now using the Ambiguity decomposition, we have the result

$$E\{\frac{1}{M}\sum_i (f_i-t)^2 - \frac{1}{M}\sum_i (f_i-\bar{f})^2\} = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1-\frac{1}{M}\right)\overline{covar}. \tag{53}$$

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 (53). Wherever the α appears in the derivation will indicate how the Ambiguity term plays a role in the bias-variance-covariance decomposition. We have

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

now multiply out the brackets, thus

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

and evaluate the expectation and summation, giving us

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

and finally by rearranging the last term we obtain

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

Obviously now if we remove the α term that we have been using, this would simplify to give us the squared 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,

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

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

$$\begin{aligned} E\{\frac{1}{M}\sum_i (f_i - t)^2\} &= \frac{1}{M}\sum_i E\left\{(f_i - E\{\bar{f}\})^2\right\} + (E\{\bar{f}\} - t)^2 \\ &= \Omega + bias(\bar{f})^2. \end{aligned}$$

This interaction term, Ω, 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, we see

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

where we have used that $E\{f_i E\{f_i\}\} = E\{f_i\}^2$ and also $E\{f_i E\{\bar{f}\}\} = E\{f_i\}E\{\bar{f}\}$. This shows that the interaction term, Ω, is the average variance of the estimators, plus the average squared deviation of the expectations of the individuals from the expectation of the ensemble.

Appendix C. Further Gradient Analysis of NC Learning

We have seen that the MSE of an ensemble system can be interpreted in two ways: firstly with the Ambiguity decomposition, and secondly with the bias-variance-covariance decomposition. We now present a third way to understand the dynamics of a regression ensemble, in reference to the gradient of the error function. Regard the architecture in figure 24. This is an ensemble of three

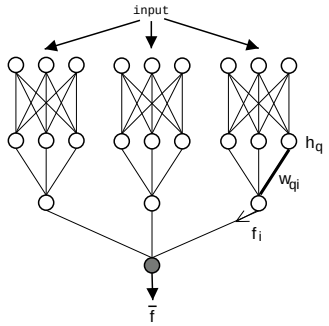

Figure 24: A typical ensemble architecture

MLPs, with three inputs and three hidden nodes each, using a uniformly weighted combination as 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 consider the ensemble as a single entity, then the error of this system at a single point is defined as

$$e_{ens} = \frac{1}{2}(\bar{f} - t)^2. \tag{54}$$

In this case, an update to the weight w_{qi} would involve the gradient

$$\begin{aligned} \frac{\partial e_{ens}}{\partial w_{qi}} &= \frac{\partial e_{ens}}{\partial f_i}\frac{\partial f_i}{\partial w_{qi}} \\ &= \frac{1}{M}(\bar{f} - d)\frac{\partial f_i}{\partial w_{qi}}. \end{aligned} \tag{55}$$

Note that we are assuming an ensemble of networks with linear output functions—if this is the case, the second term in the above error gradient, $\frac{\partial f_i}{\partial w_{qi}}$ evaluates to simply the output of the relevant hidden node. 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 calculated (55) in one simple step using the chain rule, treating the ensemble as a single unit—if we perform this instead starting from the decomposed form of the ensemble error, it highlights more interesting results. We use the Ambiguity decomposition, and additionally break the error into two components, where the first term concerns estimator i, and the second concerns all the other estimators $j \neq i$,

$$\begin{aligned} \frac{1}{2}(\bar{f} - t)^2 &= \frac{1}{M}\sum_i \left[\frac{1}{2}(f_i - t)^2 - \frac{1}{2}(f_i - \bar{f})^2\right] \\ &= \frac{1}{M}\left[\frac{1}{2}(f_i - t)^2 - \frac{1}{2}(f_i - \bar{f})^2\right] \\ &\quad + \frac{1}{M}\sum_{j\neq i}\left[\frac{1}{2}(f_j - t)^2 - \frac{1}{2}(f_j - \bar{f})^2\right]. \end{aligned} \tag{56}$$

If we do this we discover that the gradient of the ensemble error function is a sum of four distinct components, shown and described in table 3. Each of these components contributes to the gradient of the ensemble error in eq. (55). If we take the $\frac{1}{M}$ on the outside and label the components, we can make an interesting observation.

$$\frac{\partial e_{ens}}{\partial f_i} = \frac{1}{M}\Big[\underbrace{(f_i - t)}_{A} \underbrace{-(f_i - \bar{f})}_{B} \underbrace{+\frac{1}{M}(f_i - \bar{f})}_{C} \underbrace{+\frac{1}{M}\sum_{j\neq i}(f_j - \bar{f})}_{D}\Big] \tag{57}$$

We now see that the gradient of the individual, and the gradient of the ensemble as a single unit, can be expressed as combinations of these components; thus we have

$$\frac{\partial e_i}{\partial f_i} = (f_i - t) = A \tag{58}$$

Component	*Interpretation*
$\frac{1}{M}(f_i - t)$	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 3: Ensemble gradient components

$$\frac{\partial e_{ens}}{\partial f_i} = \frac{1}{M}(\bar{f} - t) = \frac{1}{M}(A - B). \tag{59}$$

Furthermore, a simple rearrangement now shows the error gradient for f_i in an ensemble using NC is

$$\begin{aligned} \frac{\partial}{\partial f_i}\left[\frac{1}{2}(f_i - t)^2 - \gamma(f_i - \bar{f})^2\right] &= (f_i - t) - 2\gamma(1 - \frac{1}{M})(f_i - \bar{f}) \\ &= (f_i - t) - 2\gamma\left[(f_i - \bar{f}) - \frac{1}{M}(f_i - \bar{f})\right] \\ &= A - 2\gamma(B - C). \end{aligned} \tag{60}$$

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, a single framework, the relationships between minimising the simple ensemble error, the NC ensemble error, and a single network, described in table 4.

If we set $\lambda = 1$, or equivalently $\gamma = \frac{M}{2(M-1)}$, we see that the gradient of the individual error with NC is directly proportional to the gradient for the ensemble seen as a single entity, i.e.

$$\frac{\partial e_{ens}}{\partial w_{qi}} = \frac{1}{M}\frac{\partial e_i}{\partial w_{qi}}. \tag{61}$$

An alternative way of thinking about this is that 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 change γ within a certain range, we scale smoothly between the gradient of a single large entity, and that of a set of independently trained networks. The choice

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 4: Components of the Ensemble Error Gradient under different Algorithms

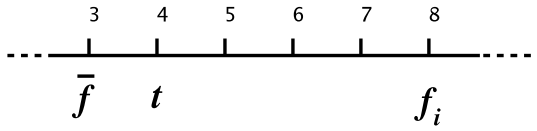

Figure 25: An regression example to illustrate how NC affects the error gradient.

of γ is problem-dependent, and it emerges that we can also understand the *optimal* setting of the parameter in this gradient-based context—consider the scenario in figure 25.

On a single datapoint, the network f_i is estimating too high at 8.0, which is right of the target $t=4$. 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 $(f_i-t)=4$, resulting in f_i being shifted left, towards the target. However, this will cause the ensemble output $\bar{f}$ to also shift *left*, moving *away from the target*. An ensemble using NC will include three gradient components,

$$\begin{aligned} A-2\gamma(B-C) &= (f_i-t)-2\gamma\Big[(f_i-\bar{f})-\frac{1}{M}(f_i-\bar{f})\Big] \\ &= 4-2\gamma(5-\frac{1}{5}5) \\ &= 4-\gamma 8. \end{aligned} \tag{62}$$

If we choose $\gamma=0.4$, this sum evaluates to 0.8, still a positive gradient for f_i, meaning the ensemble output will still be moved away from the target. If however we choose $\gamma=0.6$, it evaluates to -0.8, 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 t. This is obviously a purely hypothetical situation, and finding the optimal γ that allows this correct trade-off will be more difficult.

References

G. Brown. *Diversity in Neural Network Ensembles*. PhD thesis, School of Computer Science, University of Birmingham, 2004.

G. Brown, J. L. Wyatt, R. Harris, and X. Yao. Diversity creation methods: A survey and categorisation. *Journal of Information Fusion*, 6(1):5–20, 2005a.

G. Brown, J. L. Wyatt, and P. Sun. Between two extremes: Examining decompositions of the ensemble objective function. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 3541)*, Monterey, California, 2005b. Springer.

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

G. Fumera and F. 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, 2003. Springer.

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

J. V. Hansen. *Combining Predictors: Meta Machine Learning Methods and Bias/Variance and Ambiguity Decompositions*. PhD thesis, Aarhus Universitet, Datalogisk Institut, 2000.

A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. *NIPS*, 7:231–238, 1995.

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

Y. 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.

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

Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4), 2000.

H. Markowitz. Portfolio selection. *Journal of Finance*, 7, 1952.

R. McKay and H. 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.

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.

B. 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.

P. Sollich and A. Krogh. Learning with ensembles: How overfitting can be useful. 8:190–196, 1996.

P. Tino, I. Nabney, B. S. Williams, J. Losel, and Y. Sun. Non-linear prediction of quantitative structure-activity relationships. *Journal of Chemical Information and Computer Sciences*, 44(5): 1647–1653, 2004.

K. Tumer and J. 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.

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

X. Yao, M. Fischer, and G. 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.