



The Dynamics of Negative Correlation Learning

MARK EASTWOOD AND BOGDAN GABRYS

Computational Intelligence Research Group, School of Design, Engineering and Computing, Bournemouth University, Bournemouth, UK

Received: 1 May 2006; Revised: 1 November 2006; Accepted: 2 April 2007

Abstract. In this paper we combine two points made in two previous papers on negative correlation learning (NC) by different authors, which have theoretical implications for the optimal setting of λ , a parameter of the method whose correct choice is critical for stability and good performance. An expression for the optimal λ is derived whose value λ^* depends only on the number of classifiers in the ensemble. This result arises from the form of the ambiguity decomposition of the ensemble error, and the close links between this and the error function used in NC. By analyzing the dynamics of the outputs we find dramatically different behavior for $\lambda < \lambda^*$, $\lambda = \lambda^*$ and $\lambda > \lambda^*$, providing further motivation for our choice of λ and theoretical explanations for some empirical observations in other papers on NC. These results will be illustrated using well known synthetic and medical datasets.

Keywords: negative correlation learning, dynamics, stability, ensemble methods, combination, classification, neural networks

1. Introduction

Combining classifiers has proved successful in reducing generalization error in both the classification and prediction domains. It works best when diverse classifiers are combined [17], therefore the goal of many methods has been de-correlation of the individual outputs. Other methods such as boosting [5] concentrate on actively reducing the training error, as opposed to de-correlation as such, although the resulting ensemble may still be highly de-correlated. Popular examples of the first are input decimation [16], random forests [2], and bagging [1] to name but a few. In these algorithms, a diversification method is put in place and classifiers are trained entirely independently of one another, so to a certain extent we rely on chance to provide complementary classifiers. Other methods generate classi-

fiers sequentially [5, 12], and the current classifier is actively designed to complement the previous ones. Negative correlation learning is an error de-correlation method which instead generates predictors in parallel; *all* members of the ensemble are actively designed to be complementary to each other. This is quite an attractive property, though there are extremely successful examples of both of the other approaches. Other examples of methods in which members are trained in parallel are mixture of experts [7], and an evolutionary method in [13] whose fitness function is very similar to the error function used in NC.

For a given mean individual error rate, negative correlation of classifier outputs holds the potential for even larger reductions in error through combination [14], though this is difficult to achieve in practice. Negative correlation learning as described in [10] is a way of training an ensemble of neural

networks in parallel, in such a way as to enforce de-correlation or even negative correlation of the individual network outputs while retaining accuracy. This is achieved through a modification of the error function for each network in the form of an additive penalty term. Penalty terms are often used for regularization when training neural networks, for example in weight decay [8] to penalize large weights. In the case of NC, this idea is used to penalize correlations between the ensemble members in order to de-correlate the outputs, hopefully reducing ensemble error. The method has a parameter, λ , which controls the relative importance of the penalty term. Setting this correctly is important for good performance, however there are some aspects of how the behavior of NC depends on this choice which are puzzling. The optimal λ has been observed in [3] to tend to 0.5 as the number of networks used in the ensemble increases, but why this would happen and the shape the curve takes has not been understood. In addition, the error has been observed to diverge if λ is too large, but the value at which this rapid rise begins for different datasets seemed to be unpredictable. The theoretical work in this paper aims to understand these observations via an analysis of the dynamics of the individual outputs. This leads us to the derivation of a theoretically optimal choice of λ , potentially allowing us to improve performance and take a step closer to being able to automatically set the parameters of the algorithm.

NC learning at the moment is a method which can only be applied to base learners based on gradient descent of a continuous error function. MLP networks are often used and will be assumed in this paper. The evolutionary method mentioned earlier, while also using neural networks, could be adapted to use any base classifier, resulting in a method that to a certain extent mirrors NC learning for a general base classifier. The only method which translates the ‘penalty term’ idea to general base classifiers and zero–one loss is DECORATE [12], which creates artificial data and labels it probabilistically in *opposition* to the current ensemble prediction. New classifiers are trained on the original data and the artificial data, thus introducing a penalty term into the misclassification rate which penalizes agreement with the ensemble. The ratio of the sizes of the artificial and original datasets can be thought of as playing a similar role to λ in NC. However, individuals are trained sequentially not in parallel, and the lack of an

ambiguity-like decomposition for zero–one loss makes its theoretical foundations less solid than NC.

The structure of the remainder of the paper is as follows. In the next section we will describe the NC algorithm in more detail, while Section 3 will consider the problem of the optimal setting of λ . We will derive an expression for a value, λ^* , which is optimal from one theoretical viewpoint and which explains the decay of the optimal λ to 0.5 as ensemble size is increased. In Section 4 we will investigate how the dynamics of the individual predictor outputs f_i depend on the setting of λ , discovering regions of very different behavior defined by λ^* . This will provide further motivation for our choice of λ as well as insight into the limits of λ and the error divergences observed when λ is too large. Empirical results testing and illustrating the theoretical claims will be presented in Section 5, and the paper will be concluded in Section 6.

2. Negative Correlation Learning

In the NC method [10], an ensemble of M MLP neural networks are trained in parallel using back-propagation. The error function for each network, in addition to the usual squared error term, contains a penalty term p_i proportional to the correlation of the network predictions with those of all the other networks, making the error for a network:

$$\begin{aligned} E_i &= \frac{1}{N} \sum_{n=1}^N E_i(n) \\ &= \frac{1}{N} \sum_{n=1}^N \frac{1}{2} [f_i(n) - d(n)]^2 + \frac{1}{N} \sum_{n=1}^N \lambda p_i(n) \end{aligned} \quad (1)$$

where the sum is over the N training examples, f_i is an individual output, and d is the target. For simplicity of notation we will consider the error function at just a single point from now on, removing the necessity for the index n and the sum over points. The penalty term is:

$$p_i = (f_i - f) \sum_{j \neq i} (f_j - f) \quad (2)$$

where f is the ensemble output. This measures and penalizes correlations between predictors. In fact, if as in [3] we use the fact that the sum of a set of

values around their mean is zero, $\sum_i (f_i - f) = 0$, we can write:

$$p_i = (f_i - f)[-(f_i - f)] = -(f_i - f)^2 \quad (3)$$

which is the ambiguity [9] of the predictor, making the error function

$$E_i = \frac{1}{2}(f_i - d)^2 - \lambda(f_i - f)^2. \quad (4)$$

From the expression for the ambiguity decomposition of the ensemble error with equal weights, we have:

$$\frac{1}{2}(f - d)^2 = \frac{1}{M} \sum_i \left[\frac{1}{2}(f_i - d)^2 - \frac{1}{2}(f_i - f)^2 \right] \quad (5)$$

so we can see that if we set $\lambda = \frac{1}{2}$ in Eq. (4) then the error function we are using to train each network is actually its contribution to the ensemble error as given in the ambiguity decomposition. This is the theoretical grounding of the method; it works because it takes the *whole* of the contribution of the network to the ensemble error into account, not just the component due to the individual error but that due to the correlations also. It allows us to adjust the relative importance of the two terms, though we will argue later that this freedom should not be exercised as the form of the ambiguity decomposition decides the optimal choice of lambda.

NC has been quite successful in both regression and classification problems [10, 11], and is an attractive method due to its explicit link with the ambiguity decomposition. The success of NC has led to the proposal of some variations of the method using different penalty terms in Eq. (1). One method in particular, called root quartic negative correlation learning [11], has been shown capable of outperforming NC on some problems. The penalty term in this method is $p_i = \sqrt{\frac{1}{M} \sum_{j=1}^M (f_i - f_j)^4}$, however the choice of penalty term has little grounding in theory at the moment, and its success is not well understood.

An elaboration of NC in [6], called constructive neural network ensembles (CNNE) extends the NC framework to allow the number of hidden nodes in each network to be determined by the algorithm. Differing numbers of training epochs may also be used for different networks.

These derivative methods are valuable contributions, however some aspects of the behaviour of the original NC algorithm have not been well under-

stood, particularly the behaviour as λ is varied. This has made it difficult to know without exhaustively trying many different settings what a good setting of λ is likely to be for a particular problem.

When building an NC ensemble, an important choice for good performance is the setting of λ . As we can see from Eq. (4) a larger (smaller) value of λ corresponds respectively to a greater or lesser emphasis of the ambiguity term resulting in a larger (smaller) emphasis on the spread of the predictions compared to individual accuracy. A greater understanding of the behavior of NC is needed in order to guide the choice of λ , and has been the motivation for our paper. The dynamics of NC will be explored further in the next few sections, where the derivation and motivation for a particular choice of λ depending only on the number of networks in the ensemble will be presented.

3. Setting the λ Parameter

An initial contemplation of the expression for the error in Eq. (4), may suggest that a natural choice of λ would be $\frac{1}{2}$. With this choice, the sum of the error functions of the individuals is the error of the ensemble as a whole, expressed in its ambiguity decomposition:

$$\begin{aligned} E_{ens} &= \frac{1}{2}(f - d)^2 \\ &= \frac{1}{M} \sum_i \left[\frac{1}{2}(f_i - d)^2 - \frac{1}{2}(f_i - f)^2 \right] = \frac{1}{M} \sum_i E_i \end{aligned} \quad (6)$$

and the individual error functions are simply the contribution of each member to the ensemble error. However, if we seek to minimize these error functions by gradient descent, it is the gradient of the error function and not its actual value that is important as it is this that informs the algorithm. This was noted in Liu's paper [10], where for $\lambda = 1$ it was shown that $\frac{\partial E_i}{\partial f_i} \propto \frac{\partial E_{ens}}{\partial f_i}$, i.e., the gradient of an individual's error function w.r.t f_i is proportional to the gradient of the ensemble error w.r.t f_i . The calculation leading to this however relies on an incorrect assumption, as pointed out in [3]. The original calculation assumed that the ensemble output $f = \frac{1}{M} \sum_{i=1}^M f_i$ was constant w.r.t any single individual output f_i , which is not true and in the context used could not even be assumed for large M .

Taking this correction into account, the calculation proceeds as follows. Starting from the individual error,

$$E_i = \frac{1}{2}(f_i - d)^2 - \lambda(f_i - f)^2 \quad (7)$$

$$\begin{aligned} \frac{\partial E_i}{\partial f_i} &= (f_i - d) - 2\lambda \left(1 - \frac{\partial f}{\partial f_i}\right) (f_i - f) \\ &= (f_i - d) - 2\lambda \left(1 - \frac{1}{M}\right) (f_i - f). \end{aligned} \quad (8)$$

To gain a better understanding of this, we will rearrange the above to give

$$\frac{\partial E_i}{\partial f_i} = (f - d) + \left[1 - 2\lambda \left(1 - \frac{1}{M}\right)\right] (f_i - f). \quad (9)$$

From here on, we will write the expression in square brackets as $\theta = [1 - 2\lambda(1 - \frac{1}{M})]$. When performing gradient descent, the first term causes an individual output to move to reduce the ensemble error (regardless of the direction of the individual error), and the second term acts to move the individual output away or towards the ensemble mean depending on the sign of θ . We will look at the effects of this in more detail later.

We also have $\frac{\partial E_{ens}}{\partial f_i} = \frac{1}{M}(f - d)$, so we see from Eq. (9) that in order to achieve $\frac{\partial E_i}{\partial f_i} \propto \frac{\partial E_{ens}}{\partial f_i}$ we have to choose λ such that $\theta = 0$. This leads to a choice

$$\lambda^* = \frac{1}{2} \left(1 - \frac{1}{M}\right)^{-1}. \quad (10)$$

With this setting, by performing gradient descent on the individual error functions, we perform gradient descent on the ensemble error, which is a highly desirable situation. At each iteration we are updating the f_i to decrease the ensemble error E_{ens} even if individual accuracy may suffer. It is the ensemble error that is the important quantity, so it is clear that this is a situation we should aim for. For any other choice of λ we are not minimizing the ensemble error. Finally, we note that as $M \rightarrow \infty$, $\lambda^* \rightarrow \frac{1}{2}$, the value our initial intuition would suggest from the form of the ambiguity decomposition. For smaller M , the extra multiplicative term reflects the fact that when adjusting f_i , f will also track the adjustment to some extent. This provides an explanation of the empirical observation in [3] that the optimal setting

of λ decays to 0.5 as we add more networks (note our λ is equivalent to γ as used in their paper).

In the empirical Section 5 we will try to see whether this theoretical advantage translates into a reduction of the error on a well-known synthetic dataset, together with two more realistic datasets from the medical domain.

Apart from this optimal setting, we can also set limits on the value a sensible choice of λ would take. A negative value would defeat the point of NC learning, and of an ensemble method in general as it would encourage the individuals to be very similar, thus removing all advantage from combining. For an upper limit, as in [3] we can demand that the second derivative of the error function remains positive so that we retain a minimum of the error function:

$$\frac{\partial^2 E_i}{\partial f_i^2} = 1 - 2\lambda \left(1 - \frac{1}{M}\right)^2 > 0 \quad (11)$$

which gives an upper limit of $\lambda_{upper} = \frac{1}{2} \left(1 - \frac{1}{M}\right)^{-2}$ though this does not take into account the collective nature of NC learning. We will see in the next section that in practice we may be able to give even tighter limits, by looking at the dynamics of the f_i as λ is varied and their implications for the stability of the algorithm.

4. Dynamics of the f_i as λ Varies

In training a network by gradient descent we aim, at each timestep (denoted by a bracketed superscript), to update the output of the network so that

$$f_i^{(n+1)} = f_i^{(n)} - \eta \frac{\partial E_i}{\partial f_i} \quad (12)$$

with $\eta > 0$ the learning rate. In the case of MLP networks considered, we have to perform back-propagation to find the weight adjustments which will result in this desired update, but for the moment we will imagine that we can update the f_i exactly according to this formula.

As mentioned earlier, the error function for a network and its derivative are

$$E_i = \frac{1}{2}(f_i - d)^2 - \lambda(f_i - f)^2 \quad (13)$$

$$\frac{\partial E_i}{\partial f_i} = (f - d) + \theta(f_i - f) \quad (14)$$

recalling that $\theta = [1 - 2\lambda(1 - \frac{1}{M})]$. Now, let us consider how the difference between an individual output f_j and the mean of the remaining outputs $\frac{1}{M-1} \sum_{i \neq j} f_i$ evolves as we adjust the f_i over timesteps. We will express this difference $y_j = f_j - \frac{1}{M-1} \sum_{i \neq j} f_i$ after an update, $y_j^{(n+1)}$, in terms of its value at the previous timestep $y_j^{(n)}$. We have

$$y_j^{(n+1)} = f_j^{(n+1)} - \frac{1}{M-1} \sum_{i \neq j} f_i^{(n+1)} \quad (15)$$

which, using Eq. (12) to express the $f_i^{(n+1)}$ in terms of the $f_i^{(n)}$, becomes

$$\begin{aligned} y_j^{(n+1)} &= f_j^{(n)} - \eta \left[(f^{(n)} - d) + \theta (f_j^{(n)} - f^{(n)}) \right] \\ &\quad - \frac{1}{M-1} \sum_{i \neq j} \left[f_i^{(n)} - \eta (f^{(n)} - d) - \eta \theta (f_i^{(n)} - f^{(n)}) \right]. \end{aligned} \quad (16)$$

The non-subscripted terms cancel out, leaving us with

$$\begin{aligned} y_j^{(n+1)} &= f_j^{(n)} - \frac{1}{M-1} f_i^{(n)} \\ &\quad - \eta \theta \left(f_j^{(n)} - \frac{1}{M-1} \sum_{i \neq j} f_i^{(n)} \right) \\ &= (1 - \eta \theta) y_j^{(n)} \end{aligned} \quad (17)$$

so we have after n steps that $y_j^{(n)} = (1 - \eta \theta)^n y_j^{(0)}$, or taking the limit of continuous time and integrating in Eq. (17), we can express the result as

$$y_j(t) = y_j(0) e^{-\eta \theta t}. \quad (18)$$

What we are most interested in is the behavior of $(f_i - f)$, for two reasons. Firstly it appears in the expression for $\frac{\partial E_j}{\partial f_i}$ in Eq. (14) and so has an important effect on the dynamics. We are also interested in how the f_i are spread about their mean for its own sake. We can relate this to the above result by noting that $f_i - f = (1 - \frac{1}{M}) y_i$, so we can see from Eq. (18) that we have three different behaviors of $f_i - f$ depending on θ . For $\theta > 0$, $f_i - f$ decreases exponentially, and the individual f_i converge to a single value over time. For $\theta < 0$, $f_i - f$ increases exponentially, and the f_i will spread ever further away from their mean. If $\theta = 0$, the training

algorithm has no effect on $f_i - f$. Recalling that $\theta = [1 - 2\lambda(1 - \frac{1}{M})]$ and $\lambda^* = \frac{1}{2}(1 - \frac{1}{M})^{-1}$ from Section 3, we find that the values of λ corresponding to these three domains are $\lambda < \lambda^*$, $\lambda > \lambda^*$, and $\lambda = \lambda^*$ respectively. We will come back to the consequences of these observations a little later.

We can also look at how the ensemble error evolves over time. A similar calculation to that above results in

$$f^{(n+1)} - d = (1 - \eta)(f^{(n)} - d) \quad (19)$$

with the corresponding expression in continuous time

$$E_{ens}(t) = E_{ens}(0) e^{-\eta t}. \quad (20)$$

Regardless of how we choose λ , we can see from this that the ensemble error decreases exponentially with time. We re-iterate here however, that this is in an ideal situation where we can update the f_i exactly according to Eq. (12). We also note that to maintain a certain E_{ens} , the updates to the f_i must be accurate to within approximately E_{ens} . With $\lambda > \lambda^*$, the update $-\eta \frac{\partial E_j}{\partial f_i}$ is the sum of a term exponentially increasing with time, and one exponentially decreasing with time. This means that beyond a certain time (which can be shown to be fairly small for typical initial conditions and λ more than a few % above λ^*) the absolute size of the updates is monotonically (and approximately exponentially) increasing over time. Maintaining an ensemble error less than some E_{ens} therefore depends on making updates Δf_i with ever decreasing relative error of order $\frac{E_{ens}}{\Delta f_i} \sim e^{\eta \theta t}$.

We now return to the real world and consider the effect on this analysis, and especially the last point above, of the fact that we cannot just update the f_i at will. Each f_i is the output of a complex mathematical model (the network), and given a desired update we must perform back-propagation to estimate the weight updates resulting in the desired change in f_i . The update to f_i we actually achieve at a point will be a noisy approximation to the ideal update, for various reasons such as potentially competing weight updates from other training points, and limitations on the form of function possible with the chosen architecture. The practical consequences of this are that, over time, for $\lambda > \lambda^*$ the increasing relative accuracy of updates necessary to maintain a given E_{ens} becomes impossible to achieve, and the error will diverge. This provides further motivation for our choice $\lambda = \lambda^*$. If $\lambda < \lambda^*$ the individual outputs will tend to converge to a very similar value, reducing the

advantages of combining (though the algorithm is at least stable), and for $\lambda > \lambda^*$ we have unstable behaviour, neither being properties we would generally like the algorithm to have.

In experiments with architectures with linear output nodes, the error divergence above has been observed consistently at values of λ only slightly larger than λ^* . This does not contradict observations in previous papers of divergence occurring at varying, much less predictable values of $\lambda > \lambda^*$, because in these papers sigmoid output nodes have been used. This obviously limits how spread the outputs can be, but does not remove the problem. Forcing the output nodes to operate near to their saturation values will certainly cause its own problems, though this will be less clear-cut, resulting in error divergences at much more unpredictable values of λ dependent on the problem at hand.

We can gain some intuition for what form this problem may take by considering a simple case. Take $M = 2$ and imagine $\lambda = 0$, i.e. we are just training each network independently. In the expressions for the E_i 's in Eq. (4) only the first term remains and in the ideal case to minimize these we would have $f_1 = d$ and $f_2 = d$, giving an ensemble output $f = d$. If λ was large, essentially only the second, spreading term would remain in Eq. (4) and we can imagine that the stable state we would reach minimizing these would have $f_1 = 1$ and $f_2 = 0$ or vice-versa, giving an ensemble output $f = \frac{1}{2}$ regardless of the target d , even in the ideal case. This gives us a hint that perhaps the problem when choosing λ too large using sigmoid output nodes will manifest itself in a displacement of the stable solutions of the system away from the desired target d .

To develop this further, we can again consider the dynamics of the system. A similar analysis to the above for the ensemble error in this case is much more difficult to interpret, and in the general case does not seem to reduce to anything useful. However if we confine our interest to λ near λ^* and f near d we can make some progress. This is the most interesting area anyway as we wish to look at the stability of the algorithm near $f = d$ for values $\lambda < \lambda^*$, $\lambda = \lambda^*$ and $\lambda > \lambda^*$. The details of the calculation is relegated to the [Appendix](#); in brief, a similar strategy is followed to that above, but the outputs are $f_i = \phi(x_i) = \frac{1}{1+e^{-x_i}}$ and we assume during gradient descent we can update the x_i as we wish according to $x_i^{(n+1)} = x_i^{(n)} - \eta \frac{\partial E_i}{\partial x_i}$. The proof relies on the fact that with λ

near λ^* and f near d , both θ and $(f - d)$ are small, allowing expansion of various expressions to first order.

The result we end up with is

$$y^{(n+1)} = (1 - K)y^{(n)} - \frac{\theta\eta}{M} \sum_i f_{i,(n)}^2 (1 - f_i^{(n)})^2 (f_i^{(n)} - f^{(n)}) \quad (21)$$

where $y = f - d$ and $K = \frac{\eta}{M} \sum_i f_{i,(n)}^2 (1 - f_i^{(n)})^2 > 0$. We see that once again $f = d$ (or $y = 0$) is a stable solution of the system for $\theta = 0$, i.e. for $\lambda = \lambda^*$. However, we notice an interesting thing for $\theta \neq 0$. The state $f = d$ is no longer a steady state of the system in general! Now, an additional constraint must be satisfied. For a solution with $f = d$, we must also have

$$\sum_i f_{i,(n)}^2 (1 - f_i^{(n)})^2 (f_i^{(n)} - f^{(n)}) = 0. \quad (22)$$

The only obvious solutions to this are states where all $f_i \in \{0, 1\}$, or all the individual outputs are equal, $f_i = f \forall f_i$. There would probably be other solutions too, but these would have to be unstable for $f = d$, as the second term in Eq. (24) is the only surviving term if $f = d$ and forces spreading or convergence of the individual outputs (depending upon the sign of θ) until one of the two conditions above are met, or we no longer have $f = d$. In the case of $\theta > 0$, where individual solutions will tend to become very similar, we have no serious problems because no matter what the target d is, we can have the stable solution with all $f_i = d$. This is not ideal though because we lose any advantages that could be gained by combining. For $\theta < 0$ we do have a problem, because this solution is no longer stable. Unless the target is of the form $d = \frac{m}{M}$ for some $m \in \{0, \dots, M\}$, we cannot find a stable solution with $f = d$. We will instead find a state where f is displaced away from d towards one of a few discrete values of the form $\frac{m}{M}$ to some extent which will depend on the magnitude of θ and the target d . As λ is increased, more and more of the input space will be forced very near to one of these discrete values. This is the reason that the error divergences observed for sigmoid output nodes are less dramatic and much less predictable. A certain displacement of the stable state of the system away from the target will not immediately cause huge errors, especially in a classification context.

The Dynamics of Negative Correlation Learning

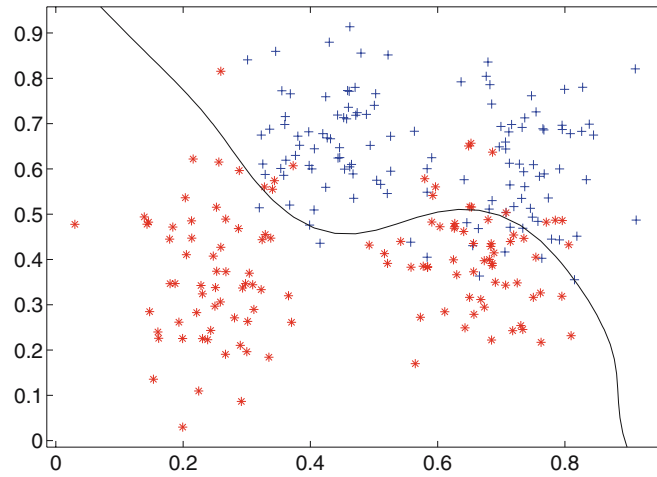


Figure 1. The synthetic dataset, and an example NC classification.

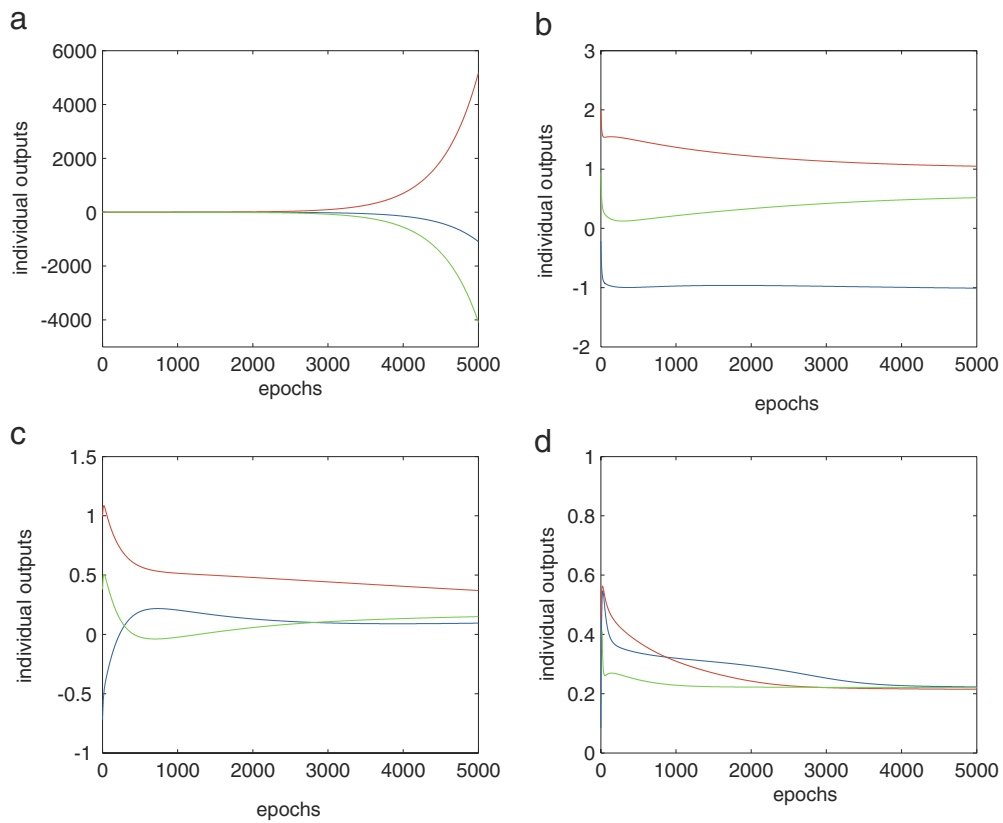


Figure 2. The dynamics of the individual outputs for various λ . The curves show, for each individual, the average output of a single output node over all points of the corresponding class. The ensemble size is 3. From top left we have a) $\lambda = 0.76$, b) $\lambda = \lambda^* = 0.75$, c) $\lambda = 0.7$, and d) $\lambda = 0$.

Although we still do not have an exact picture of the dynamics for non-linear output nodes, hopefully this has provided some intuition for the behavior in this case, and the problems which may still arise for $\lambda > \lambda^*$. The impact of these problems is much less predictable as details of the specific targets for the problem, and the initial f_i , will tend to affect the size and direction of the displacement of stable solutions from $f = d$ in a complicated way.

5. Experimental Analysis

In this section we will present empirical results which illustrate and support the theoretical claims made in the previous sections. We will take a look at examples of the dynamics on the synthetic dataset, a 2-class, 2-D dataset with each class consisting of two overlapping multivariate Gaussians. The classes overlap significantly, with the Bayes error at roughly 8%. The training set consists of 250 patterns, with a test set of 1,000. Figure 1 shows the training set, and an example classification produced by NC.

The experimental setup was as follows. An ensemble of three MLP neural networks was used, each with five hidden nodes with the tansig transfer function. Linear output nodes were used, and the networks were trained for 5,000 epochs at learning rate $\eta = 0.05$. Weights and biases are initialized via the Nguyen–Widrow algorithm. The targets are in one-of-k form. We will focus on the output of just one of the two output nodes for the training points, so the targets are 0 for all points of class 1, and 1 for all points of class 2. We will plot the average output over all training points of class 1, for each of the individual networks, to see the overall trends as training progresses. The results for various values of λ are shown in Fig. 2.

What we notice straight away is the dramatic divergence of the individual outputs for $\lambda = 0.76$. This is a little over 1% above the value λ^* , beyond which our theoretical results predict an exponential divergence of the inputs, resulting rapidly in a divergence for the error also. We can see the effects of this divergence on the error in Fig. 3. This

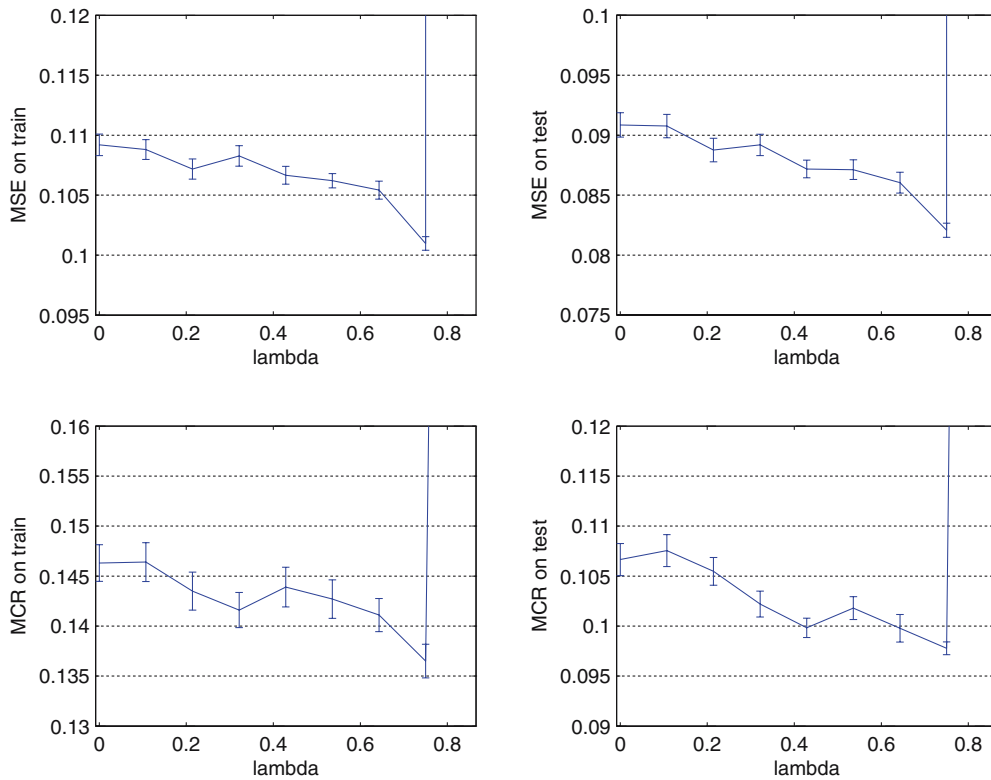


Figure 3. MSE and MCR on both training and testing sets as λ increases on the synthetic dataset.

divergence becomes quickly more rapid as λ is increased still further. For $\lambda = \lambda^* = 0.75$ we see no pronounced convergence or divergence of the individual outputs—they are being adjusted in a complementary way to reduce the ensemble error with preference being shown neither for similarity nor unnecessary spread. For λ a little below λ^* , by just 0.05, we already see a definite tendency of the outputs to converge to a similar value, and by $\lambda = 0$ corresponding to independent training of the networks, the individuals become very similar indeed. This illustrates the three different characteristic behaviors discussed in Section 4. What we would like to know now is whether this translates to an optimal setting of λ in terms of the ensemble error.

We will again use the synthetic dataset, and two more realistic datasets from the medical domain. These datasets, and further information about them can be found in the UCI machine learning database [4]. The liver dataset is a 2 class, 6 feature dataset with 345 examples taken from male patients. Five of

these features are numerical values corresponding to the results of various blood tests thought to be sensitive to liver disorders, the sixth is the average number of half-pint equivalent drinks per day. The cancer dataset is also a 2 class problem, with 30 features and 569 examples. The features are computed from digitized images of the cell, and the classes are defined by their diagnosis as malignant (212 examples) or benign (357 examples). Further information on this dataset can be found in [15]. Both datasets were split (as nearly as possible) into equally sized training and test sets.

On the synthetic dataset, negative correlation ensembles were generated for $M = 2, \dots, 6$, and for each case nine values of λ were used, eight evenly spaced in the range 0 to λ^* , and the final one the same step size above λ^* serving to illustrate the error divergence when $\lambda > \lambda^*$ when using linear output nodes. The experimental setup is as above apart from the fact that 2,500 training epochs were used. The results for $M = 3$ (for which $\lambda^* = 0.75$) can be found in Fig. 3.

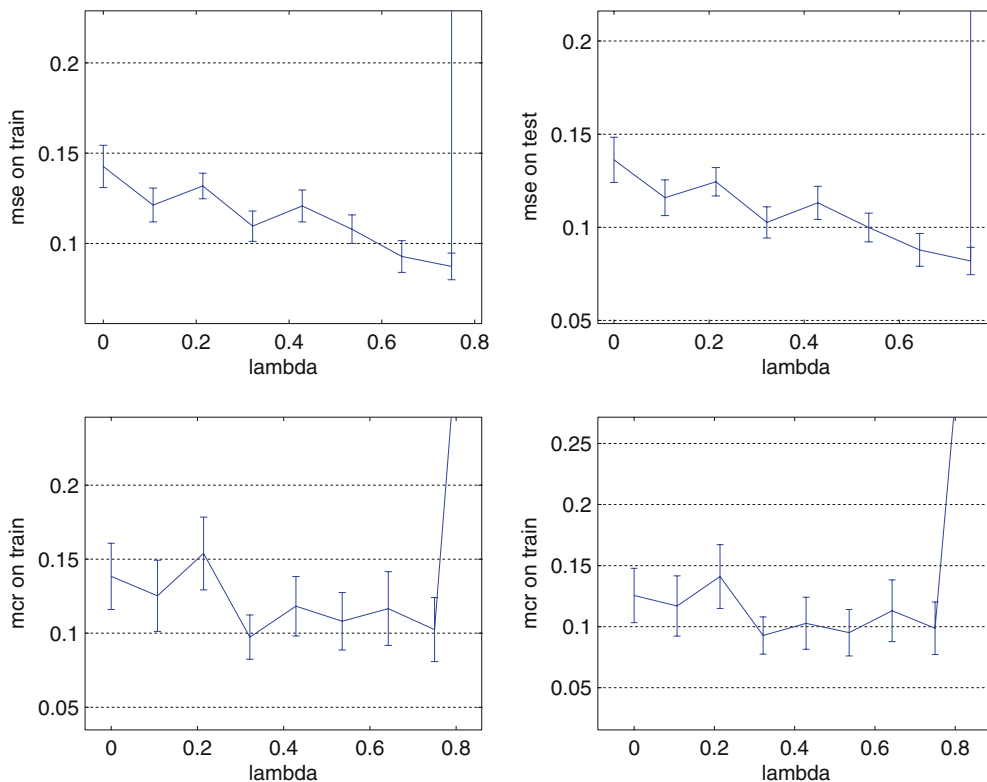


Figure 4. MSE and MCR on both training and testing sets as λ increases on the cancer dataset.

Results for all the ensemble sizes tested showed very similar qualitative behavior. We can see the results of the divergence of the individual outputs and the instability this causes very clearly in the huge increase in error beyond λ^* , and a definite tendency of the error to decrease as λ is increased to λ^* , showing that an NC ensemble is significantly better than a combination of independently trained networks for this dataset.

On the liver and cancer datasets, ensembles of three networks were generated. The parameter settings were respectively 5 nodes, 5,000 epochs with learning rate 0.0005, and 10 nodes, 7,000 epochs with learning rate 0.00004. The parameter settings used have been chosen purely for illustrative purposes to demonstrate the different behaviour one can expect from NC, without any particular attempt at optimizing performance. On the cancer dataset, we can see from Fig. 4 that for MSE we have a similar trend to that seen in Fig. 3 for the synthetic dataset, though it is hard to see in the case of MCR, illustrating the fact that MSE often does not correspond particularly closely to MCR. However, the optimal λ again seems

to be $\lambda = \lambda^*$, with a rapid increase in error beyond this value.

If we look at the results in Fig. 5 for the liver dataset, we see different behaviour. On the training set the story is similar to the results for the other datasets, and we see decreasing error as λ is increased down to a minimum at λ^* , followed by the characteristic divergence in error beyond λ^* . However on the testing sets, we see a minimum in the error before λ^* is reached, followed by a gentle increase up to λ^* and a rapid increase beyond λ^* . In this case λ^* is not optimal. A decreasing trend in the training error accompanied by the opposite trend in testing error is the classic sign of overfitting. Our choice λ^* is optimal in the sense that for this value the individual networks will cooperate, and their outputs be de-correlated, to the greatest extent compatible with stability. This cooperative adjustment of the weights allows more complex functions to be fit, leading to improved performance if greater complexity is needed but also the potential for overfitting if it is not. In some sense the value of λ acts to control the complexity of the ensemble classifier. Thus our choice

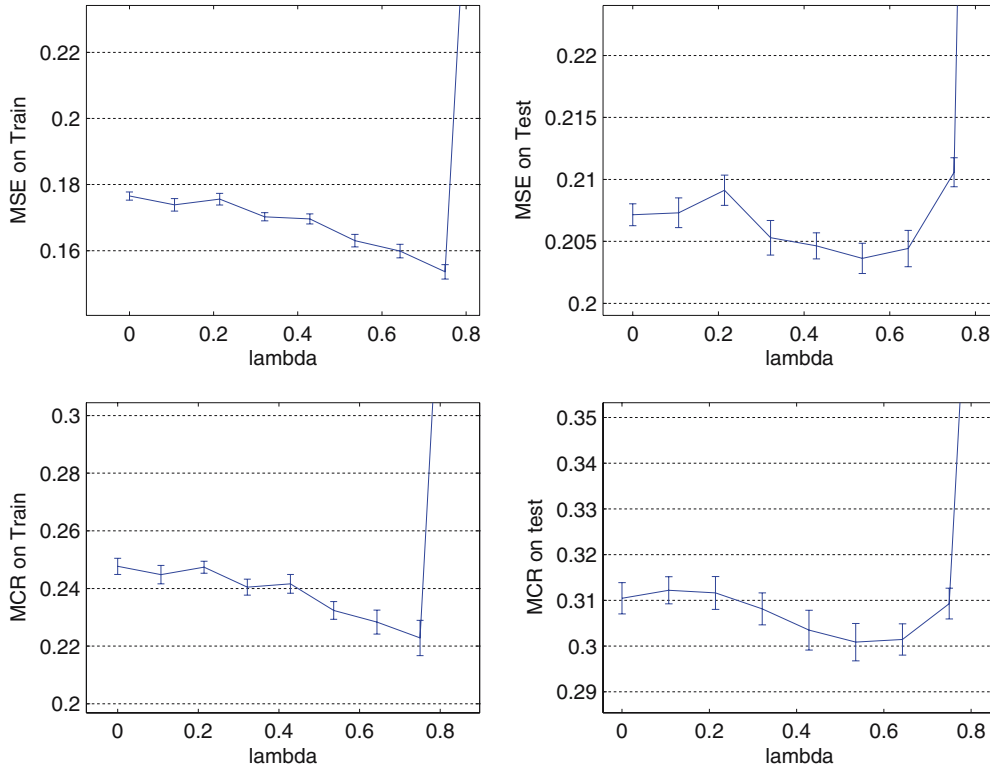


Figure 5. MSE and MCR on both training and testing sets as λ increases on the liver dataset.

λ^* is not optimal in an absolute sense but must be chosen in conjunction with a suitable number of hidden nodes and ensemble size. What is ‘suitable’ for a given problem is an area requiring further work.

The results for $\lambda = 0$, corresponding to independent training of the networks, and for $\lambda = \lambda^*$, for each dataset are summarised in Table 1.

6. Conclusions

In this paper we have addressed the question of how to choose the λ parameter in NC learning, by investigating how the dynamics of the algorithm are affected by this choice. We have argued that $\frac{\partial E_{\text{ens}}}{\partial f_i} \propto \frac{\partial E_i}{\partial f_i}$ is a situation we should aim for, and derived the value λ^* depending only on M for which this is true. We have then proceeded to investigate the effects of the choice of λ on the stability of NC learning, with λ^* appearing again as a limiting value for stability of the algorithm. This has provided further motivation for our choice of λ and we have found that for $\lambda > \lambda^*$ we have unstable behaviour, manifesting itself in different ways depending on the output nodes used. This provides explanations for some previously observed phenomena, although we still cannot characterize the exact value of λ beyond which the error will diverge for sigmoid output nodes in the same way that we can for linear output nodes. To understand how this depends on the specific problem at hand, and probably on the initial values of the f_i , is an area needing further research. However so long as we choose $\lambda \leq \lambda^*$ (as of course we always can as $\lambda^* = \frac{1}{2}(1 - \frac{1}{M})^{-1}$ is known) we can be assured of a stable algorithm. Finally, we have illustrated these theoretical results with empirical

Table 1. Summary of results for $\lambda = 0$ (independently trained networks) and $\lambda = \lambda^*$ (NC ensemble with de-correlated outputs).

	MSE	Var($\times 10^{-4}$)	MCR	Var($\times 10^{-3}$)
$\lambda = 0$				
Synth	0.091	0.414	0.107	0.104
Cancer	0.136	29.0	0.126	9.90
Liver	0.207	0.155	0.312	0.236
$\lambda = \lambda^*$				
Synth	0.082	0.139	0.098	0.016
Cancer	0.082	11.0	0.098	9.30
Liver	0.211	10.273	0.309	0.226

results showing the existence in practice of some of the implications of our theory, and suggesting that up to a certain complexity of individual network our choice is optimal. However, λ^* is optimal only in the sense that the individual networks co-operate most fully for this value, resulting in the ability to fit more complex models. The algorithm provides no protection against over-fitting. Thus if the added complexity introduced by this co-operation is not appropriate to the complexity of the problem at hand, we may see over-fitting with λ^* optimal on the training set and a lesser value of λ optimal in terms of performance on the testing set.

Appendix

We will proceed in a similar manner to that of Section 4, skipping a few of the details for brevity. The outputs f_i are given now by $f_i = \phi(x_i)$, with $\phi(x) = \frac{1}{1+e^{-x}}$. We will allow ourselves to update the x_i according to

$$x_i^{(n+1)} = x_i^{(n)} - \eta \frac{\partial E_i}{\partial x_i} \quad (23)$$

though again in practice approximate updates would be made by adjusting weights via back-propagation. From the chain rule, we have $\frac{\partial E_i}{\partial x_i} = \frac{\partial E_i}{\partial f_i} \frac{\partial f_i}{\partial x_i} = \frac{\partial E_i}{\partial f_i} f_i(1-f_i)$. In the following calculations we will write

$$\begin{aligned} A_i &= -\eta \frac{\partial E_i}{\partial f_i} f_i(1-f_i) \\ &= -\eta[(f-d) + \theta(f_i-f)]f_i(1-f_i) \end{aligned} \quad (24)$$

so that $x_i^{(n+1)} = x_i^{(n)} + A_i^{(n)}$. We want to know how the ensemble error $y = f - d$ evolves over time so we will try to express $y^{(n+1)}$ in terms of $y^{(n)}$. We have

$$y^{(n+1)} = \frac{1}{M} \sum_i f_i^{(n+1)} - d \quad (25)$$

and using Eq. (23)

$$f_i^{(n+1)} = \phi(x_i^{(n+1)}) = \phi(x_i^{(n)} + A_i^{(n)}). \quad (26)$$

Note that because ϕ is monotonically increasing with its argument, the update to x_i and the corresponding update to f_i will have the same sign, so the second term in Eq. (24) still has a spreading or converging (depending on sign of θ) effect on the f_i as in the linear

case. Substituting the expression for $f_i^{(n+1)}$ in Eq. (26) into Eq. (25) we have

$$y^{(n+1)} = \frac{1}{M} \sum_i \phi \left(x_i^{(n)} + A_i^{(n)} \right) - d. \quad (27)$$

Using the fact that $\phi(a+b) = \frac{\phi_a \phi_b}{\phi_a \phi_b + (\phi_a - 1)(\phi_b - 1)}$ for any a and b (writing ϕ_a for $\phi(a)$ etc), which can be proved easily from the definition of ϕ , we have

$$y^{(n+1)} = \frac{1}{M} \sum_i \left(\frac{\phi_i^{(n)} \phi_{A_i}^{(n)}}{\phi_i^{(n)} \phi_{A_i}^{(n)} + (\phi_i^{(n)} - 1)(\phi_{A_i}^{(n)} - 1)} \right) - d. \quad (28)$$

Now, we consider λ near λ^* and $f^{(n)}$ near d , so that A_i is small. In this case, we can expand $\phi_{A_i} \approx \frac{1}{2} \left(1 + \frac{A_i}{2} \right)$ to first order, and substitute in above. After a few lines of rearrangement and further expansions of the form $(1 + \delta)^{-1} \approx (1 - \delta)$ for small δ , we arrive at

$$y^{(n+1)} \approx \frac{1}{M} \sum_i \phi_i^{(n)} \left(1 + A_i^{(n)} (1 - \phi_i^{(n)}) \right) - d. \quad (29)$$

Recalling that by definition $f_i = \phi_i$ and $y = f - d$, the above becomes

$$y^{(n+1)} \approx y^{(n)} + \frac{1}{M} \sum_i A_i (1 - f_i). \quad (30)$$

Substituting in for A_i and rearranging gives the result:

$$y^{(n+1)} \approx (1 - K)y^{(n)} - \frac{\theta\eta}{M} \sum_i f_{i,(n)}^2 (1 - f_i^{(n)})^2 (f_i^{(n)} - f^{(n)}) \quad (31)$$

where $K = \frac{\eta}{M} \sum_i f_{i,(n)}^2 (1 - f_i^{(n)})^2 > 0$.

References

1. L. Breiman, "Bagging Predictors," *Machine Learning*, vol. 24, no. 2, 1996, pp. 123–140.
2. L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, 2001, pp. 5–32.
3. G. Brown, and J.L. Wyatt, "The Use of the Ambiguity Decomposition in Neural Network Ensemble Learning Methods," in *20th International Conference on Machine Learning (ICML'03)*, T. Fawcett and N. Mishra (Eds.), Washington DC, USA, August 2003.
4. C.L. Blake D.J. Newman, S. Hettich, and C.J. Merz, *UCI Repository of Machine Learning Databases*, 1998.
5. Y. Freund and R.E. Schapire, "Experiments with a New Boosting Algorithm," in *Proceedings of the 13th International Conference on Machine Learning*, Morgan Kaufmann, 1996, pp. 148–156.
6. Md.M. Islam, X. Yao, and K. Murase, "A Constructive Algorithm for Training Cooperative Neural Network Ensembles," *IEEE Transactions on Neural Networks*, vol. 14, no. 4, 2003, pp. 820–834 (July).
7. R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, vol. 3, no. 1, 1991, pp. 79–87.
8. A. Krogh, and J.A. Hertz, "A Simple Weight Decay Can Improve Generalization," in *Advances in Neural Information Processing Systems, volume 4*, J.E. Moody, S.J. Hanson, and R.P. Lippmann (Eds.), Morgan Kaufmann Publishers, Inc., 1992, pp. 950–957.
9. A. Krogh and J. Vedelsby, "Neural Network Ensembles, Cross Validation, and Active Learning," *Advances in Neural Information Processing Systems*, vol. 7, 1995, pp. 231–238.
10. Y. Liu and X. Yao, "Ensemble Learning Via Negative Correlation," *Neural Networks*, vol. 12, 1999, pp. 1399–1404.
11. R. McKay and H. Abbass, "Analyzing Anticorrelation in Ensemble Learning," in *Proceedings of 2001 Conference on Artificial Neural Networks and Expert Systems*, Otago, New Zealand, 2001, pp. 22–27.
12. P. Melville and R. Mooney, "Constructing diverse classifier ensembles using artificial training examples," in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, Mexico, August 2003, pp. 505–510.
13. D. Opitz and J. Shavlik, "A genetic Algorithm Approach for Creating Neural Network Ensembles," *Combining Artificial Neural Nets*, Springer, 1999, pp. 79–99.
14. D. Ruta and B. Gabrys, "A Theoretical Analysis of the Limits of Majority Voting Errors for Multiple Classifier Systems," *Pattern Analysis and Applications*, vol. 5, 2002, pp. 333–350.
15. W.N. Street, W.H. Wolberg, and O.L. Mangasarian, "Nuclear Feature Extraction for Breast Tumour Diagnosis," *International Symposium on Electronic Imaging: Science and Technology*, vol. 1905, 1993, pp. 861–870.
16. K. Tumer and N.C. Oza, "Input Decimated Ensembles," *Pattern Analysis and Applications*, vol. 6, no. 1, 2003, pp. 65–77.
17. K. Tumer and J. Ghosh, "Error Correlation and Error Reduction in Ensemble Classifiers," *Connection Science*, vol. 8, no. 3–4, 1996, pp. 385–403.

The Dynamics of Negative Correlation Learning



Mark Eastwood is a 2nd year Ph.D. student at Bournemouth University within the Design, Engineering and Computing department. The title of the project is 'High Performance Fusion Systems', an ESPRC industrial CASE funded project with industrial partner BT. He graduated as a BA/Msci in Natural Sciences from Cambridge University in 2003. His interests are Pattern Recognition and Machine Learning, with a focus on ensemble methods.



Bogdan Gabrys received his M.Sc. degree in Electronics and Telecommunication (Specialization: Computer Control Systems) from the Silesian Technical University, Poland in 1994

and a Ph.D. in Computer Science from the Nottingham Trent University, UK in 1998. After many years of working at different Universities, Professor Gabrys moved to the Bournemouth University in January 2003 where he acts as a Head of the Computational Intelligence Research Group within the School of Design, Engineering & Computing. His current research interests include a wide range of machine learning and hybrid intelligent techniques encompassing data and information fusion, multiple classifier and prediction systems, processing and modelling of uncertainty in pattern recognition, diagnostic analysis and decision support systems. He published numerous research papers and is regularly invited to give talks at universities, companies and international conferences in UK and abroad. Professor Gabrys has also reviewed for various journals, edited special issues of journals, chaired international conferences, workshops and sessions and been on programme committees of a number of international conferences with the Computational Intelligence and Soft Computing theme. Professor Gabrys is a Co-Editor in Chief of the International Journal of Knowledge Based Intelligent Engineering Systems and the Chair (Academic Affairs) and a member of KES organisation Executive Advisory Board. He currently acts as a Co-Chair of the Nature-inspired Data Technology (NiDT) focus group within the European CA project on Nature-inspired Smart Information Systems (NiSIS). In the recent years, he also acted as a Corresponding Person for a Key Node in the European Network on Intelligent Technologies for Smart Adaptive Systems (EUNITE) and a Co-Chairman of the Research Theory & Development Group on Integration of Methods. He is a senior member of the Institute of Electrical and Electronics Engineers (IEEE) and the IEEE Computational Intelligence Society and a Fellow of UK's Higher Education Academy (HEA).