LAMBDA AS A COMPLEXITY CONTROL IN NEGATIVE CORRELATION LEARNING

Mark Eastwood, Bogdan Gabrys
Computational Intelligence Research Group,
School of Design, Engineering and Computing,
Bournemouth University, UK.
Email: {meastwood,BGabrys}@bournemouth.ac.uk

ABSTRACT: The λ parameter in negative correlation learning (NC) controls the degree of co-operation between individual networks. This paper looks at the way the choice of λ in the NC algorithm affects the complexity of the function NC can fit, and shows that it acts as a complexity control allowing smooth adjustment of the network between one large back-propagated network, and many independent networks individually trained before combination. The effect of the base complexity of the individual networks, and the number of networks that are trained co-operatively, on the algorithms overall complexity are also empirically investigated. Empirical results are presented from 4 different datatsets. The way in which the performance changes as parameters change, and the point at which over-fitting sets in give us information about the complexity of the model the algorithm can fit at those parameter settings.

KEYWORDS: Negative Correlation Learning, Classification, Ensemble Methods, Combination, Neural Networks, Ambiguity Decomposition.

Introduction

When solving a classification problem, we have a set of data and their classifications which we assume has been generated by some underlying model (function). This data is used to choose a model from a certain class of models via some algorithm. One of the most important factors deciding how well this algorithm will perform for a given problem is the complexity of the model class the algorithm uses to fit to the data, and how well this matches the complexity of the model underlying the data. Too complex a model class will tend to over-fit the data. A function will be chosen that very well models the data, but which will probably generalize badly. On the other hand if the model is very simple it will not be able to fit the function without bias, although the training error should generalize well. A good general background to the area of pattern classification can be found in for example [4].

Complexity is difficult to measure, and this is reflected in the fact that there is no single accepted way to characterize the model complexity. It is often characterized by the number of free parameters (in the case of a parametric model). Sometimes the kolmogorov complexity [9], or algorithmic complexity, is used. This is the length of the shortest description of the entity in question (the model in our case) in some given language. Another complexity measure widely used is the VC dimension [14], which is the size of the largest set of points that can be arbitrarily split into two classes by the model class the algorithm uses to fit the data.

When looking at combinations of classifiers the situation is even more murky. There are examples where increasing the number of classifiers in an ensemble seems to improves the generalization performance of the ensemble monotonically, despite the fact that in adding classifiers, a naive assumption would be that the complexity should be increasing and therefore generalization performance should eventually degrade. One example of this is bagging [1], in which performance tends to decrease as more classifiers are added, converging to a steady value for a large ensemble. In this case however the individual classifiers are trained entirely independently of one-another, and are each fitting different bootstrap samplings of the data. In terms of kolmogorov complexity it can be explained as follows. A bagging ensemble of any size can be generated via the base algorithm and a simple piece of code that generates a bootstrap resampling. Therefore the ensemble can be described (generated) by code with length only a constant amount larger than the length of the base algorithm, regardless of the ensemble size. Another example is boosting [6]. In this case the classifiers are not

independent, yet the same monotonic decrease in error is usually observed in the case of boosting also. This proved to be a great surprise when it was first noticed, and has still to be completely explained. There are some partial explanations however, see for example [12].

Thus one of the major challenges in building a classifier for a given problem is choosing the appropriate complexity. NC [10] is an ensemble method in which multiple neural networks are trained co-operatively in parallel. The method is described in more detail in the next section. There are a number of different factors which have an effect on the overall complexity of the algorithm. The number of parameters in this method is the number of weights in the ensemble, and this provides a fairly natural indication of the complexity. This depends on the number of networks, and also the number of weights in an individual network. Also highly relevant to the complexity of the method however is the extent to which the the weights are determined co-operatively (i.e the value of λ), and the specific architecture of the network (i.e how many individuals the weights are split between). Ideally we would be able to automatically set the parameters of the method resulting in the appropriate complexity. This is not currently possible, but a better understanding of how the various parameters of NC affect the complexity of model it can fit will help when choosing a suitable parameter set for a given problem.

Despite the difficulties mentioned above in measuring/defining complexity, we can get an indication of how the complexity of the algorithm is changing with changes of parameter settings within the algorithm by looking at empirical data. We can look at how the error on the training set is changing, and how this relates to the generalization error we observe on a test dataset, and thus gain an indication of the complexity of the model the algorithm can fit measured against the baseline of the complexity of the model underlying the data.

This paper will take an empirical look at the complexity of the NC method in this way. We will look empirically at the effects of varying the above parameters in NC, and we will be able to compare two different paradigms; the combination of few complex networks and the combination of larger numbers of simpler networks, as both of these scenarios are easily achievable in NC. We will also show that the λ parameter provides a convenient way of controlling the complexity of the network to some extent without architectural changes.

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:

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

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:

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

where f is the ensemble output. This measures and penalizes correlations between predictors. In fact, if as in [2] 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:

$$p_i = (f_i - f)[-(f_i - f)] = -(f_i - f)^2 \tag{3}$$

which is the ambiguity [8] of the predictor, making the error function

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

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

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

so we can see that if we set $\lambda = \frac{1}{2}$ in (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. Example NC classifications can be seen in Figs. 1 and 2, the datasets are described in more detail in Sect. .

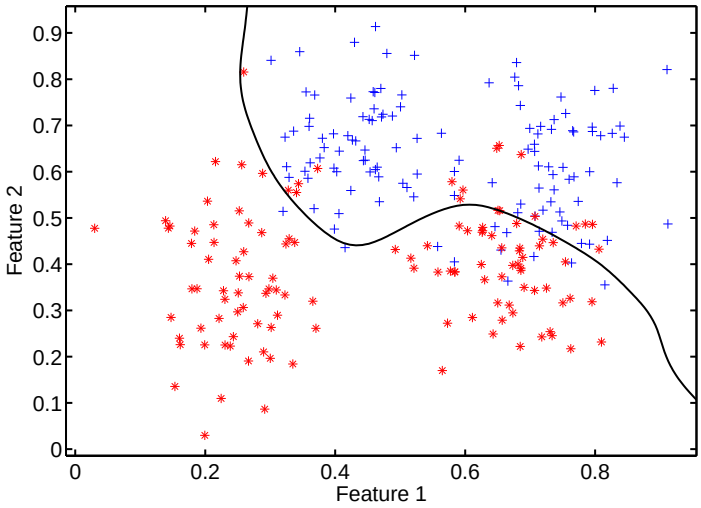

Figure 1: Example NC classification on the Synthetic dataset

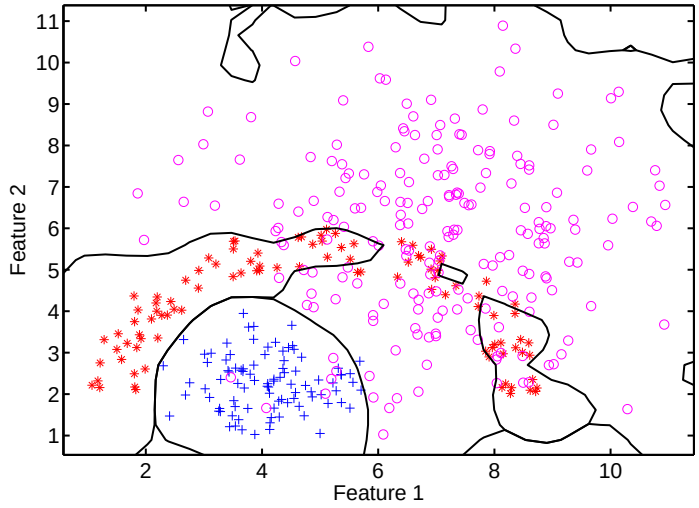

Figure 2: Example NC classification on the Cone-Torus dataset

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 (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 [7], 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 this paper will concentrate on the basic method. In [5] we showed a value of lambda $\lambda^* = \frac{1}{2}(1 - \frac{1}{M})^{-1}$ optimal in the sense that it results in the largest degree of co-operation between the individuals, consistent with stability. This however does not neces-

sarily mean it is optimal for general problems. We will show empirically in this paper that its main effect is to maximise the the complexity of the algorithm, and that as in all such cases the increased complexity is only useful if the model underlying the data is complex too. Otherwise we simply get over-fitting. We will therefore argue that λ provides us with an easy way in which to vary the complexity of the network without changing the architecture itself.

We can see what setting $\lambda = \lambda^*$ means for the algorithm by looking at the effect of changing f_j on the other error functions E_i for $i \neq j$. The error function for individual i is

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

so we have

$$\frac{\partial E_i}{\partial f_j} = (f_i - d)\delta_{ij} - 2\lambda(f_i - f)\left(\delta_{ij} - \frac{1}{M}\right)$$

where δ_{ij} is the dirac delta function. For $i = j$ we have

$$\frac{\partial E_i}{\partial f_i} = (f_i - d) - 2\lambda(f_i - f)\left(1 - \frac{1}{M}\right). \tag{6}$$

In previous papers [5],[10] we and others have simply noted that $\frac{\partial E}{\partial f_i} = (f - d)$ and chosen $\lambda = \lambda^*$ to make $\frac{\partial E_i}{\partial f_i} = \frac{\partial E}{\partial f_i}$. In [5] we showed that λ^* is the value giving maximum co-operation and diversification between the individual networks. We can gain insight into what it is we are actually doing in choosing $\lambda = \lambda^*$ by realising is that each f_j is present in each error function E_i, so we should consider the effects of changing f_i on all the error functions if we wish to train them co-operatively. For $i \neq j$ we have

$$\frac{\partial E_i}{\partial f_j} = \frac{2\lambda}{M}(f_i - f) \tag{7}$$

and so for $\lambda = \frac{1}{2}$ we have from 6 and 7

$$\sum_i \frac{\partial E_i}{\partial f_j} = (f - d) = \left.\frac{\partial E_i}{\partial f_i}\right|_{\lambda=\lambda^*}$$

as of course we must have because by the ambiguity decomposition when $\lambda = \frac{1}{2}$ we have $E = \sum_i E_i$. Thus choosing $\lambda = \lambda^*$ in the individual error functions is equivalent to including the information about the gradient w.r.t f_i of the error functions E_j for $i \neq j$ into the single error function E_i. When this is done we are simply using the ensemble error E during training, treating the ensemble as a single network as shown in Fig. 3 to be trained by back-propagation. The other extreme, of $\lambda = 0$, corresponds to treating the ensemble as individual networks as shown by the dotted lines in Fig. 3, each trained independently by back-propagation and then combined. This latter case is equivalent to diversifying the individuals simply via different weight initializations. Thus we can see that λ provides a convenient way of adjusting the complexity of the network between these two extremes without the need for changes in the architecture of the network.

Experimental setup and results

Experiments were conducted on four diverse datasets. Two artificial datasets were used. The synthetic dataset consists of 250 training and 1000 testing examples, and is a two class problem, each class a mixture of two overlapping gaussians. There is some class overlap as well; the bayes error is 0.08. The cone-torus dataset of 800 examples, split equally between training and testing sets, is a dataset with three classes, two of which are greatly overlapping. Both datasets are 2-D, and can be seen in Figs. 1 and 2. Two more realistic datasets from the medical domain were also used. These datasets, and further information about them can be found in the UCI machine learning database [3]. 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 [13]. Both datasets were split (as nearly as possible) into equally sized training and test sets.

In the following, parameters are given in the format [ensemble size][number of nodes in each individual], and lambda has been set to 8 equally spaced values between 0 and λ^*. Each parameter setting is repeated 25 times.

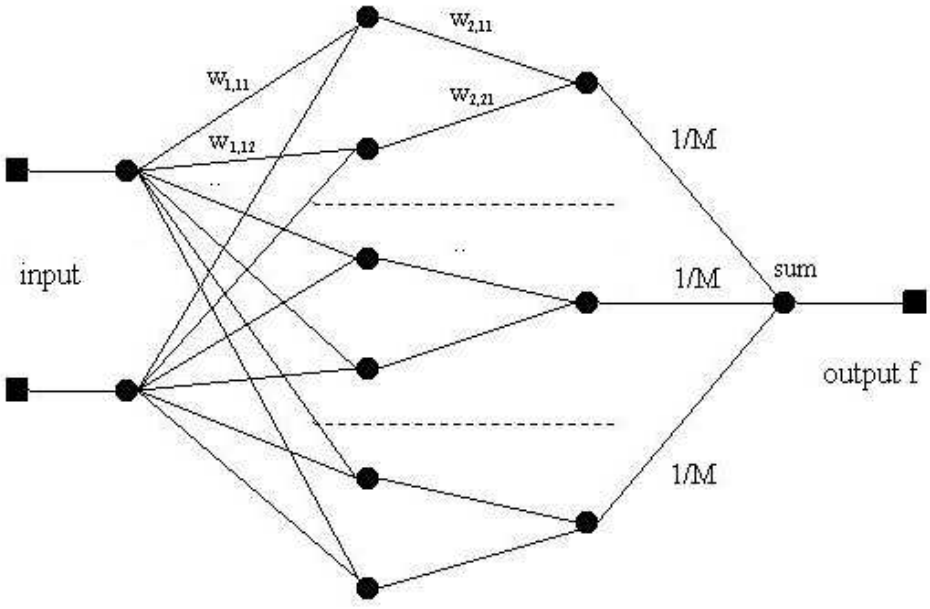

Figure 3: An illustration of the architecture of an NC network. The weights shown as $\frac{1}{M}$ are fixed. For $\lambda = 0$ the networks are trained as individuals as indicated by the dotted lines. For $\lambda = \lambda^*$ the network is trained as a whole.

We attempted to choose parameter ranges for ensemble size and number of nodes which were large enough to show changes of behaviour over their range, for example overfitting. In some cases though we have been limited by the computationally expensive nature of these experiments. Experiments have been conducted on each dataset firstly for a fixed, small ensemble size, varying the complexity (number of nodes) of the individuals. Further experiments have then been conducted with variously sized ensembles keeping the individual complexity at a small fixed number of nodes. This allows us to compare two differing methodologies, that of combining many weak individuals, and combining fewer, more complex individuals.

On the synthetic dataset, an experiment was run with parameter settings [3][5,10,20,40]. The second experiment was performed with settings [3,5,7,10,15][5]. For both these experiments, the networks were trained for 3000 epochs and with learning rate $\eta = 0.05$. Results are shown in Figs. 6 and 7.

Similar experiments were conducted on the other datasets. For the cone-torus dataset, an experiment was conducted with settings [3][5,10,20,40,60,80,100]. A second experiment was conducted with [3,5,7,10,15][5]. The number of epochs/learning rate were the same as above. Results are shown in Figs. 10 and 11.

On the liver dataset, the settings used were [3][3,6,12,24]. A further experiment was conducted with [3,5,7,10,15][5]. Here the networks are trained for 3000 epochs at learning rate 0.0006. Results are shown in Figs. 4 and 5.

For the cancer dataset, we used [3][5,10,20,40,60,80,100]. The second experiment was conducted with [3,5,7,10][20]. 5000 training epochs were conducted with a learning rate of 0.00005. Results are shown in Figs. 8 and 9.

Discussion

We notice from the figures that the number of nodes in the individual network seems to have a slightly greater effect on the complexity of the NC network then the number of individuals in the ensemble. This can be particularly seen in the synthetic dataset comparing Figs 6 and 7. On the test set over-fitting is seen in Fig. 6 that is not observed in 7, and on the training set it can be seen that increasing the number of nodes is causing a larger decreace in error than that observed when increasing the number of nets. We can also see that λ has a more pronounced effect with larger numbers of nets, as would be expected as in this case the co-operation (controlled by λ) between the networks is more important

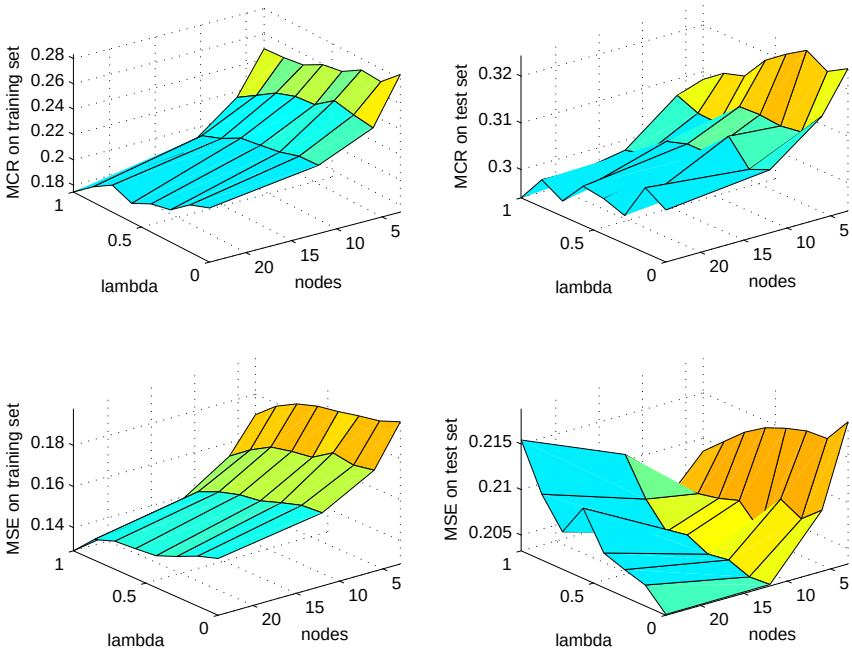

Figure 4: MCR and MSE for differing numbers of nodes on the liver dataset. The ensemble size is 3.

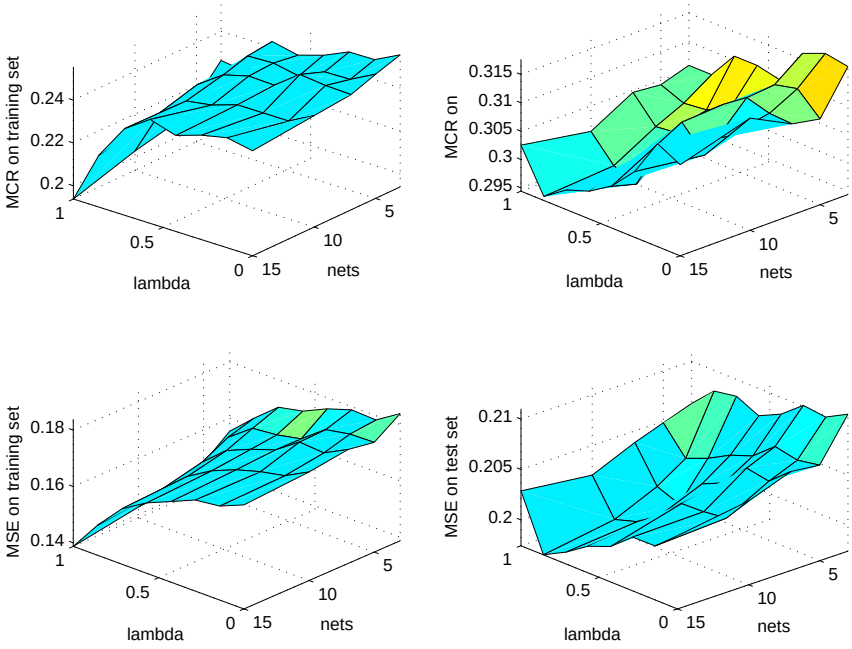

Figure 5: MCR and MSE for differing numbers of networks on the liver dataset. There are 5 nodes per net.

	λ / λ^*	Nets	Nodes	MCR on Test
Liver	1	3	24	0.294
Cancer	1	10	20	0.0692
Synthetic	$\frac{2}{7}$	3	20	0.0929
Cone-Torus	0	3	60	0.1182

Table 1: Table of the optimal (in terms of MCR on testing set) parameter settings out of the experiments performed, for each dataset.

Looking at the liver dataset (Figs. 4 and 5), we notice that the error on the training set decreases both with increacing ensemble size, number of nodes in the individual networks, and also with increacing λ. This indicates that the complexity of the model the algorithm can fit is increasing with these parameters as we would expect. We see overfitting in the MSE (mean squared error) in Fig. 4, but this does not appear in the MCR (misclassification rate). It is the MSE and not the MCR that is being directly optimised, so this is not a so surprising and serves to illustrate one of the issues with an algorithm such as this. The quantity we are optimising (MSE) and the one we are most interested in (MCR) do not always correspond closely. Finding an error function which more closely follows MCR but which can still be used in an algorithm such as this is an outstanding challenge.

On the synthetic dataset (Figs. 6 and 7), the results are similar. Again we see the trend of decreasing training error as M, λ and number of nodes are increased, with over-fitting beginning to occur as they are increased further. Here the MCR seems to track the MSE a little more closely, with over-fitting observed in both error functions when increasing the number of nodes. These results emphasise the fact that λ^* is not optimal in a general sense. for relatively weak individuals, for example with 5 nodes, we can see from Fig. 6 that $\lambda = \lambda^*$ is indeed optimal. If we make the individuals sufficiently complex however, we can see that $\lambda = 0$ is optimal. The added complexity that co-operation would introduce is no longer suitable, and simply results in over-fitting.

The cancer dataset (Figs. 8 and 9) behaves a little differently in that increasing the number of nodes in the individuals seems to have little effect. The dominant parameter seems to be λ, and the number of nets seems to have a significant effect, in contrast to the case for the Liver and Synthetic datasets. It is not clear why this should be so.

We see some odd results on the cone-torus dataset (Figs. 10 and 11). There is a very significant drop in MSE between 40 and 60 nodes, with increased complexity beyond this seeming to result in slight overfitting. This significant drop however is not really reflected at all in the MCR. The choice of λ seems to have only a minor effect. When varying the number of nets, MCR and MSE do not seem to track each-other very closely at all. The number of nets seems to be the dominant factor in determining the MSE, with the error increasing quite quickly up to 10 networks. The value of λ has only a very slight effect on the MSE. However in the MCR, the dominant factor is λ, with a large decrease in error observed as λ increases for larger numbers of networks. The number of nets has little effect. The strange behaviour of NC on this dataset is something we have no explanation for, though it is probably related to the highly over-lapping nature of the dataset.

The optimal values for each dataset from all the parameter settings tried are summarised in table 1. We can see that the character of the optimal settings varies between datasets, with highly co-operative individuals being prefered in the liver and cancer datasets whereas a combination of more complex, independently trained networks was optimal for the Cone-Torus dataset.

Conclusions

In conclusion, we find that although in general increasing λ,M, and the number of nodes in the individuals all seem to increase the complexity of the algorithm, their relative effects seem to depend in part on the details of the problem. We have shown that λ can be used as a convenient way to adjust the complexity of the network without having to change the architecture, although the architecture defines the two extremes we can achieve via adjusting λ. For $\lambda = 0$ we have a complexity equivalent to one of the individual networks. The algorithm simply trains them independently with diversity only introduced through the initialisation of weights, and they are then combined. For $\lambda = \lambda^*$ we have complexity equivalent to a larger network, with architecture as shown in Fig. 3. This value is not optimal for general problems as the increased complexity the co-operation allows is not always appropriate to the problem given a particular choice of number of nodes/nets. Knowing when this added complexity is needed is an ongoing problem in pattern classification. We have also noted that the

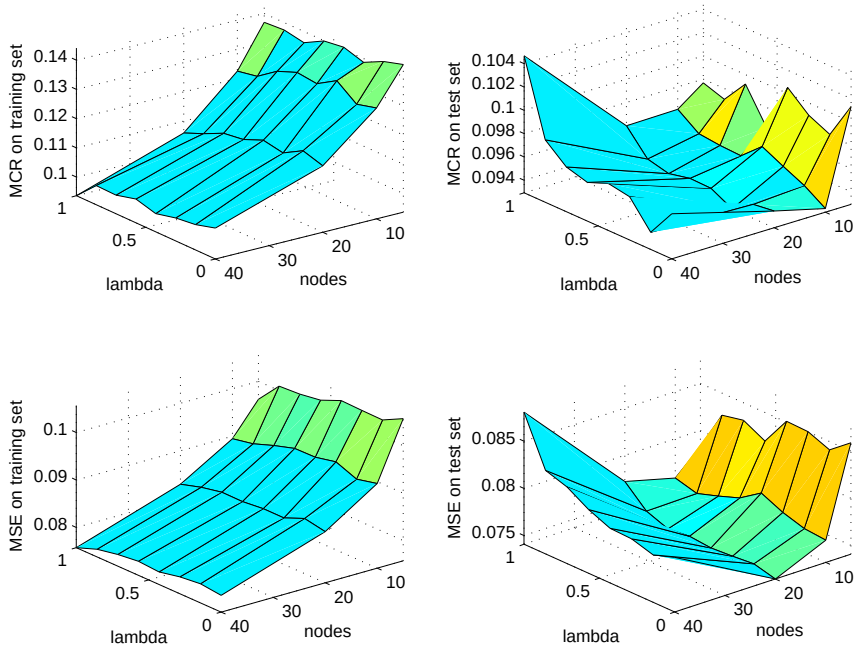

Figure 6: MCR and MSE for differing numbers of nodes on the synthetic dataset. The ensemble size is 3.

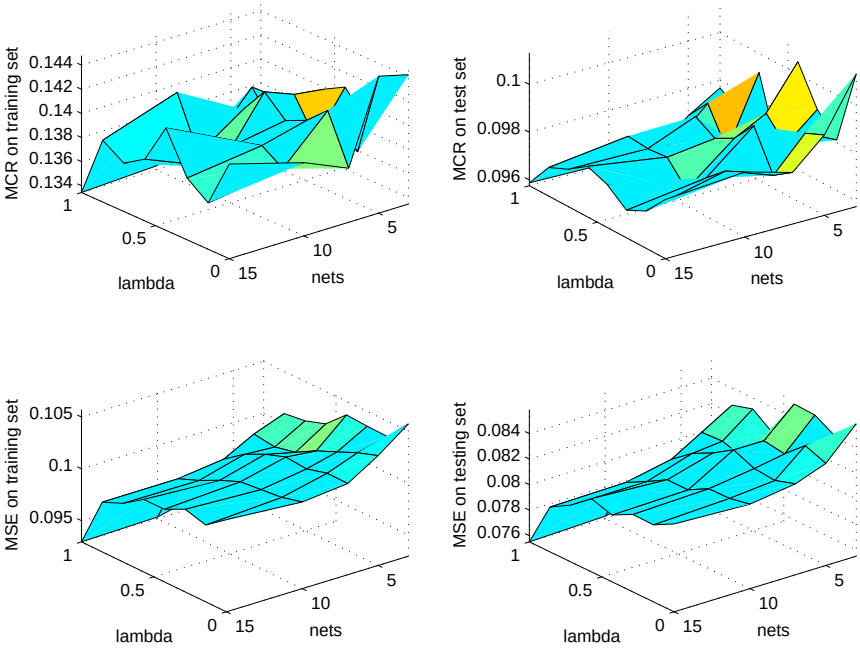

Figure 7: MCR and MSE for differing numbers of nets on the synthetic dataset. There are 5 nodes per net.

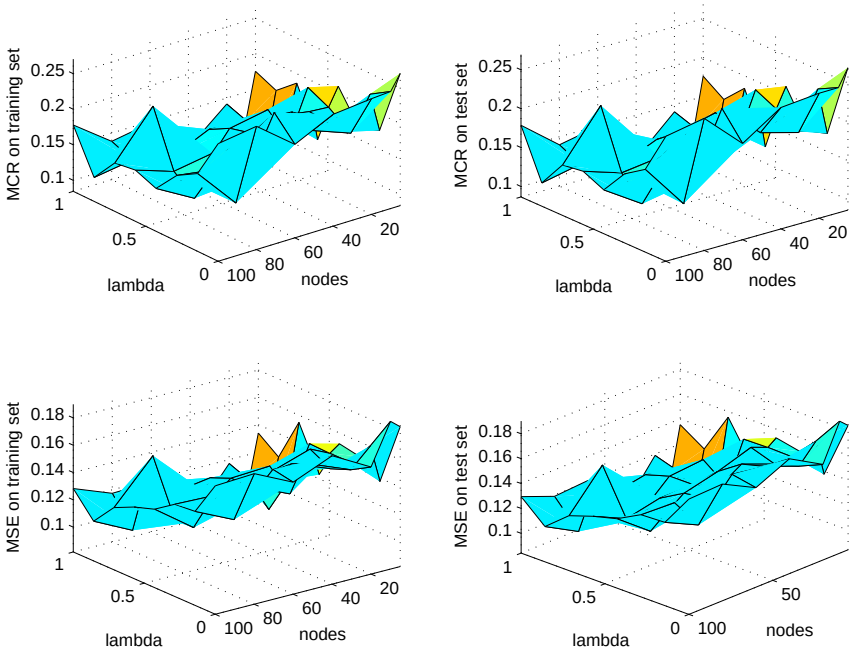

Figure 8: MCR and MSE for differing numbers of nodes on the cancer dataset. The ensemble size is 3.

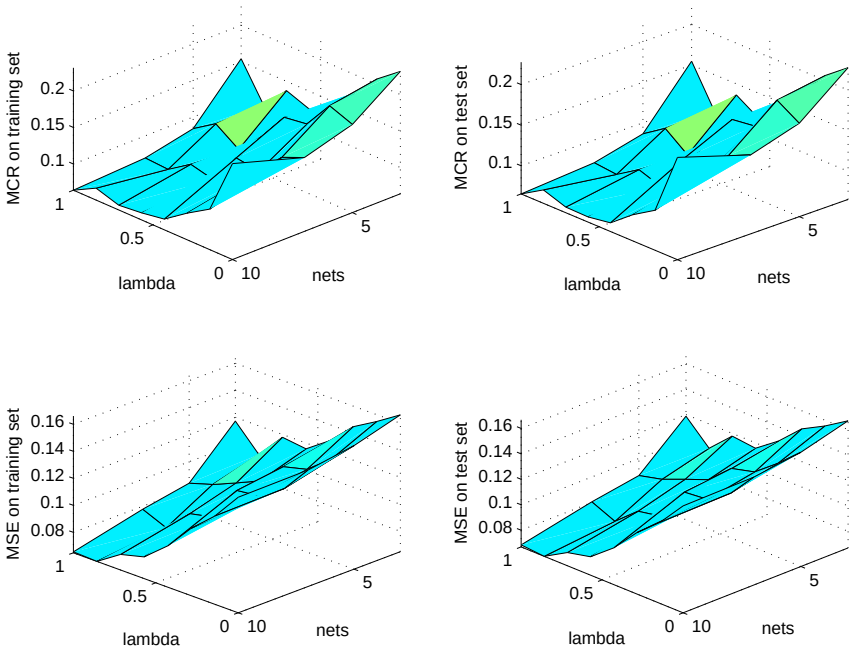

Figure 9: MCR and MSE for differing numbers of nets on the cancer dataset. There are 20 nodes per net.

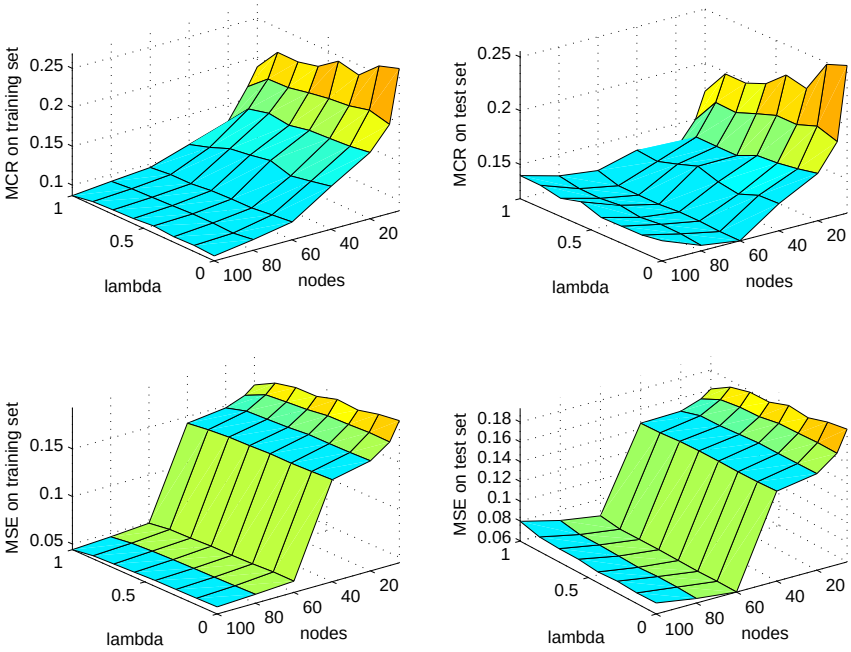

Figure 10: MCR and MSE for differing numbers of nodes on the cone-torus dataset. The ensemble size is 3.

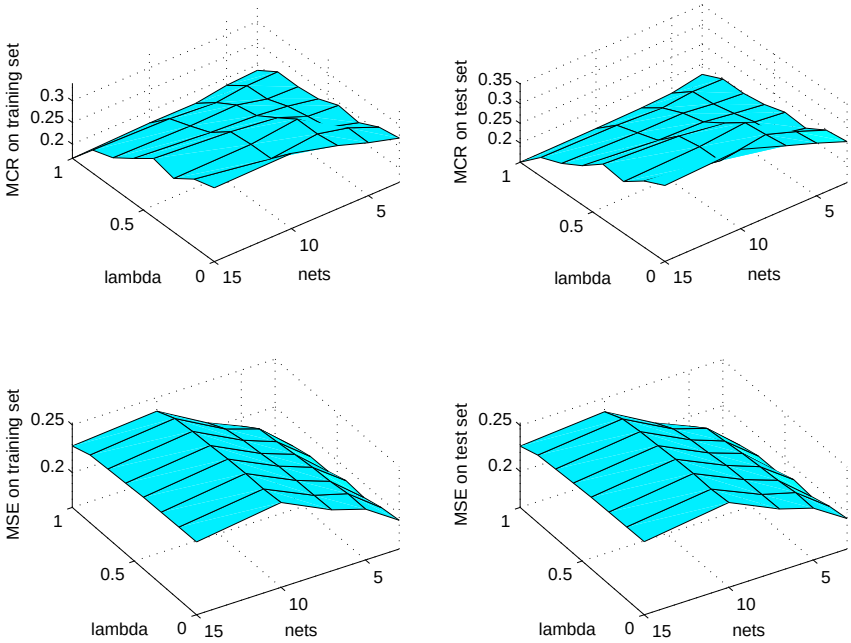

Figure 11: MCR and MSE for differing numbers of networks on the cone-torus dataset. There are 5 nodes per net.

optimisation criterion for this algorithm, MSE, does not always correspond well to the main criterion of interest, MCR. This points out an area perhaps in need of further research.

References

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

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

[3] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

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

[5] Mark Eastwood and Bogdan Gabrys. The dynamics of negative correlation learning. *Journal of VLSI Signal Processing, to be published*.

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

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

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

[9] Ming Li and Paul M. B. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, Berlin, 1993.

[10] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12:1399–1404, 1999.

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

[12] Robert Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation of the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[13] 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*, 1905:861–870, 1993.

[14] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.