

Anti-correlation: A Diversity Promoting Mechanism in Ensemble Learning

R. I. (Bob) McKay¹ and Hussein A. Abbass²

School of Computer Science,
University of New South Wales,
Australian Defence Force Academy Campus
Northcott Drive, Canberra 2600, Australia,

(E-mail : ¹rim@cs.adfa.edu.au, ²h.abbass@adfa.edu.au)

Abstract

Anti-correlation has been used in training neural network ensembles. Negative correlation learning (NCL) is the state of the art anti-correlation measure. We present an alternative anti-correlation measure, RTQRT-NCL, which shows significant improvements on our test examples for both artificial neural networks (ANN) and genetic programming (GP) learning machines. We analyze the behavior of the negative correlation measure and derive a theoretical explanation of the improved performance of RTQRT-NCL in larger ensembles.

Keywords: Anti-correlation, Artificial Neural Networks, committee learning, Ensemble learning, fitness sharing, genetic programming, diversity

1 Anti-Correlation Learning

Committee learning refers to a form of learning algorithm where a committee of learning machines is used to learn a task. The hope is that each member of the committee will specialize on a part of the task. In anti-correlation learning, a specific mechanism is incorporated in the learning mechanism to reduce correlation between the committee members, without unduly sacrificing the accuracy of prediction. The error function of each committee member needs, therefore, to have an additional penalty term which accomplishes the following

1. it maximizes the distance between all networks; and therefore achieves a nice spread in the ensemble space.
2. it is dimensionally consistent with the error function.
3. it does not have a larger magnitude than the original error function; otherwise the effect will be dramatic (ie. the networks will be so different that they contradict each other and the performance becomes chaotic).

One approach to achieve this is *Negative Correlation Learning* (NCL), proposed by Liu and Yao (1999b, 1999a).

The paper is organized as follows. In Section 2, negative correlation learning is discussed, and an alternative, RTQRT-NCL, is introduced in subsection 2.3. We then use RTQRT-NCL for two learning machines; artificial neural networks in Section 3 and genetic programming in Section 4. The results and possible theoretical explanations are discussed in Section 5, and conclusions are drawn in Section 6.

2 Negative Correlation Learning

2.1 Nomenclatures

From here on, the following notations will be used for an ensemble of single hidden layer artificial neural networks:

- I , H , and M are the number of input and hidden units, and the number of networks in the ensemble, respectively.
- $\mathbf{X}^p = (x_1^p, x_2^p, \dots, x_I^p)$, $p = 1, \dots, P \in \mathbf{X}$, is the p^{th} pattern in the input feature space \mathbf{X} of dimension I , and P is the total number of patterns.
- Without any loss of generality, $\mathbf{Y}^p \in \mathbf{Y}$ is the corresponding scalar of pattern \mathbf{X}^p in the hypothesis space \mathbf{Y} .
- $w_{ih}(m)$ and $w_h(m)$ are, respectively, the weights connecting input unit i , $i = 1 \dots I$, to hidden unit h , $h = 1 \dots H$, and hidden unit h to the output unit for network m , $m = 1 \dots M$.
- $\Theta_h(\mathbf{X}^p, m) = \sigma(a_h(m))$, where $a_h(m) = \sum_{i=0}^I w_{ih}(m)x_i^p$, $h = 1 \dots H$, $m = 1 \dots M$, is the h^{th} hidden unit's output corresponding to the

input pattern \mathbf{X}^p and network m , where $a_h(m)$ is the activation of hidden unit h for network m , and $\sigma(\cdot)$ is the activation function, which is taken in this paper to be the logistic function $\sigma(z) = \frac{1}{1+e^{-Dz}}$, with D the function's sharpness or steepness, taken to be 1 unless otherwise mentioned.

- $\hat{Y}^p(m) = \sigma(a(m))$, where $a(m) = \sum_{h=0}^H w_h(m)$
 $\Theta_h(\mathbf{X}^p, m)$ is network m 's output and $a(m)$ is the activation of the output unit corresponding to the input pattern \mathbf{X}^p and network m .
- F^p is the average output of the ensemble corresponding to input pattern p .
- $Error$ is the average error of the ensemble and $Error(m)$ is the average error of network m .

2.2 Liu and Yao Approach

Negative Correlation Learning was proposed by Liu and Yao (1999b) for training an ensemble of ANNs using Backpropagation (Liu and Yao 1999a) within an evolutionary approach (Liu, Yao, and Higuchi 2000). With an ensemble size of 4, the approach shows some success in improving the ensemble performance. To delineate the approach, let us first define the problem carefully. The ensemble is trained using the training set, where the output of the ensemble F^p is given by the following equation:

$$F^p = \frac{1}{M} \sum_{m=1}^M \hat{Y}^p(m) \quad (1)$$

The expected error of a new pattern is given by the average error of the ensemble $Error$ as defined by the following two Equations:

$$Error = \frac{1}{M} \sum_{m=1}^M Error(m) \quad (2)$$

$$Error(m) = \frac{1}{P} \sum_{p=1}^P \frac{1}{2} (\hat{Y}^p(m) - Y^p)^2 + \frac{1}{P} \sum_{p=1}^P \lambda \Phi^p(m) \quad (3)$$

Here, $\Phi^p(m)$ is the penalty function of network m and pattern p . This function represents the correlation term that we need to minimize. In NCL, the following is used as the anti-correlation measure:

$$\Phi^p(m) = (\hat{Y}^p(m) - F^p) \sum_{l \neq m} (\hat{Y}^p(l) - F^p) \quad (4)$$

The previous function has some desirable characteristics, as when it is combined with the mean square error, the

result will be a nice tradeoff between the bias, variance, and co-variance. The expected result is an ensemble comprising greater diversity, and hence giving better generalization, than the corresponding non-penalty formulation. It has the additional advantage that it does not change the Backpropagation algorithm much as it only adds a simple term to the derivative of the error as follows:

$$\frac{\partial Error(m)}{\partial \hat{Y}^p(m)} = (\hat{Y}^p(m) - Y^p) - \lambda (\hat{Y}^p(m) - F^p) \quad (5)$$

However in our work using NCL with the larger committees used in genetic programming, the degree of dispersion of the committee was rather less than expected. We investigated a number of alternative measures of anti-correlation. Of these, one in particular gave promising initial results, and seemed theoretically acceptable. That measure, which we have named root-quartic Negative Correlation Learning (RTQRT-NCL), is described below.

2.3 RTQRT-NCL: A Proposed Measure

The proposed measure is

$$\Phi^p(m) = \sqrt{\frac{1}{M} \sum_{l=1}^M (\hat{Y}^p(m) - \hat{Y}^p(l))^4} \quad (6)$$

The derivative of the corresponding error function is as follows:

$$\begin{aligned} \frac{\partial Error(m)}{\partial \hat{Y}^p(m)} &= (\hat{Y}^p(m) - Y^p) + \\ &\frac{\lambda}{\sqrt{M}} \times \frac{2 \sum_{l=1}^M (\hat{Y}^p(m) - \hat{Y}^p(l))^3}{\sqrt{\sum_{l=1}^M (\hat{Y}^p(m) - \hat{Y}^p(l))^4}} \end{aligned} \quad (7)$$

This function is dimensionally consistent with the mean square error and has a good pressure on the networks within the ensemble to spread in the ensemble space. In the following sections, the performance of the proposed penalty function is compared against the original NCL penalty function using a neural network ensemble on the Australian credit card dataset, and using genetic programming on the 6-multiplexer problem.

In the context of genetic programming, where the penalty function is applied directly, the λ values for NCL and RTQRT-NCL are directly comparable as constants, so in those experiments, we directly compare corresponding values of λ . However in ANN applications, the penalty function is applied through the derivative, hence it is less clear whether the λ values are directly comparable. In the main experiments reported here, we directly compare the same λ values. It is clear from the experiments (and theoretical considerations) that the optimal values for NCL are less

than one, hence within the ranges of the experiments conducted. This is less clear for RTQRT-NCL; the constant in the backpropagation formula is reduced by $\frac{1}{\sqrt{M}}$, so that perhaps increasing values of λ are appropriate as M increases. This is confirmed by the experiments - the optimal λ values clearly exceed 1 as M increases. Hence further experiments were conducted with larger values of λ .

3 Experiment 1: Artificial Neural Networks

3.1 Experimental Design

We ran some experiments with a benchmark dataset, the Australian credit card assessment problem. The dataset is available by anonymous ftp from ice.uci.edu (Blake and Merz 1998). This dataset contains 690 patterns with 14 attributes; 6 of them are numeric and 8 discrete (with 2 to 14 possible values). The predicted class is binary - 1 for awarding the credit and 0 for not. The objective is to assess applications for credit cards (Michie, Spiegelhalter, and Taylor 1994).

The Australian credit card assessment data set is divided into 10 folds with equal class distributions. One fold is reserved for testing and the other nine folds are used to create a validation set (four folds) and a training set (five folds). We created 30 different validation and training sets.

The ensemble size varied from 2 to 10 with an increment of 2 and the penalty parameter (λ) varied between 0 and 1 with an increment of 0.2. The neural network representation used a single hidden layer and a single output. The maximum number of epochs (complete presentation of the training set) is set to 250, and the learning rate for backpropagation is 0.1.

3.2 Results

In this section, we present the results of our experiments on the Australian credit card dataset. Figures 1, 2, 3, 4, and 5 present the average test error for the Australian credit card using NCL and RTQRT-NCL for a range of λ values and three strategies (vote, average, and winner takes all).

From Figure 1, the performance of NCL and RTQRT-NCL is not very good; although NCL did improve the performance a little using the “vote” strategy with λ values of 0.2 and 0.6. RTQRT-NCL also improved the performance with the “vote” strategy with λ value of 0.2 and with the other two strategies with λ value of 0.8. The nature of these results is expected as the committee size is very small. In this case, the penalty is quite aggressive, so that it differentiates between the two networks at the cost of losing accuracy. This can be clearly seen with NCL; whereas with RTQRT-NCL, the trend is quite stable on the “average” and “winner-take-all” strategies and quite bad on

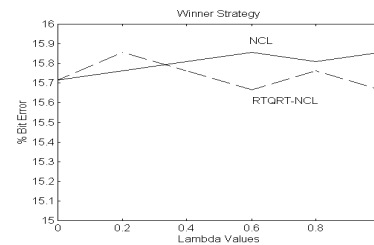
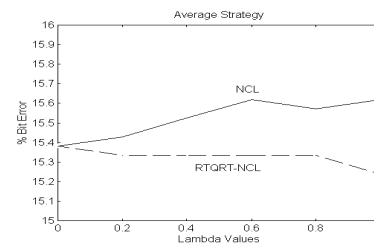
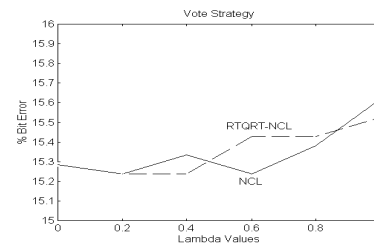


Figure 1: The average test error for the Australian Credit Card using NCL (top) and RTQRT-NCL (bottom) obtained by λ value using an ensemble size of 2. Top: vote strategy; Middle: average strategy; and Bottom: winner strategy.

the “vote” strategy after a λ value of 0.4.

In Figure 2, the picture improved for RTQRT-NCL although it did not for NCL. The accuracy decreases in the three graphs of NCL in approximately a linear trend. This is quite surprising and differs from the results of Liu and Yao (Liu and Yao 1999b). It is important to mention that the trend in NCL for the “average” and “winner-take-all” strategies with populations size 2 and 4 is almost the same. Even though the values are lower for population size 4,

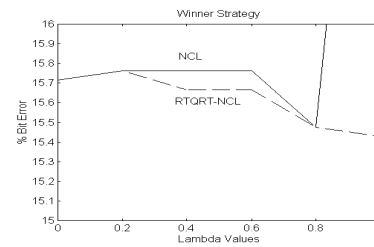
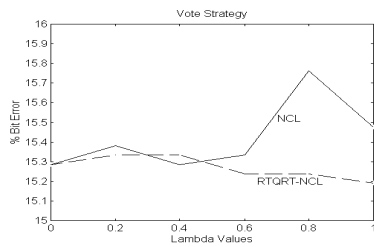
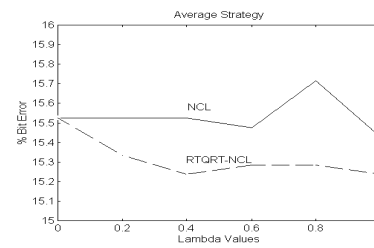
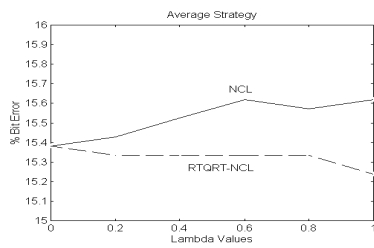
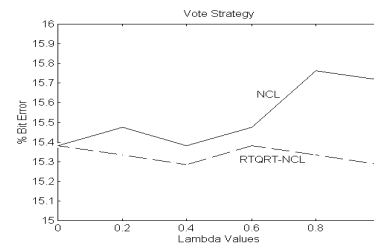
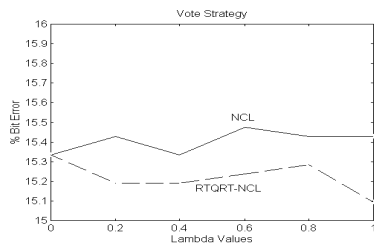


Figure 2: The average test error for the Australian Credit Card using NCL (top) and RTQRT-NCL (bottom) obtained by λ value using an ensemble size of 4. Top: vote strategy, Middle: average strategy, and Bottom: winner strategy.

Figure 3: The average test error for the Australian Credit Card using NCL (top) and RTQRT-NCL (bottom) obtained by λ value using an ensemble size of 6. Top: vote strategy, Middle: average strategy, and Bottom: winner strategy.

it is easy to see from the graph that these values are the contribution of the population size (when $\lambda = 0$, the error decreased with population size 4; however, if we subtract the error when $\lambda = 0$, the resultant graphs are almost identical). For the RTQRT-NCL algorithm, it is clear that it performed rather better than NCL in the three strategies. This trend continued for population sizes 6, 8, and 10 as shown in Figures 3, 4, and 5 respectively.

value is limited to a maximum of 1.0, hence the results have been presented above only over a lambda range from 0.0 to 1.0. However these theoretical considerations do not apply to RTQRT-NCL, and indeed, it is clear from the figures that for larger committees, values of lambda exceeding 1.0 are likely to give improved results. Experiments were carried out in which the lambda values were multiplied by the square root of the committee size (since this seemed likely to bring the figures into approximately the same scale). These are presented in Figures 6, 7, 8, 9, and 10

For theoretical reasons discussed in (1999b), the lambda

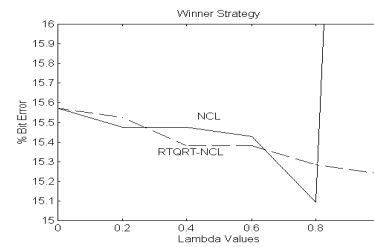
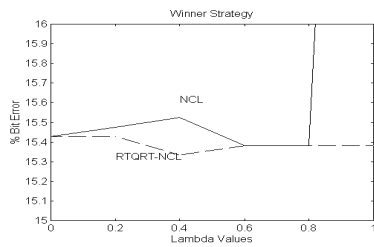
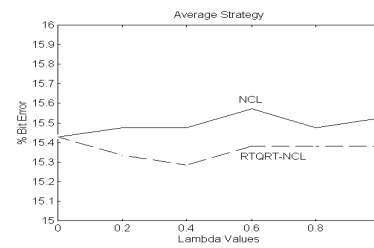
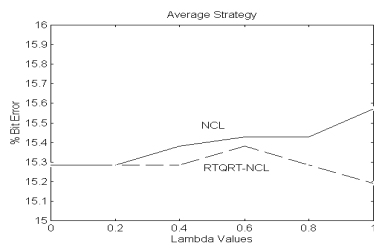
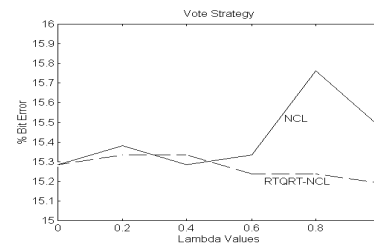
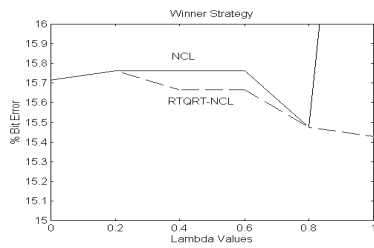


Figure 4: The average test error for the Australian Credit Card using NCL (top) and RTQRT-NCL (bottom) obtained by λ value using an ensemble size of 8. Top: vote strategy, Middle: average strategy, and Bottom: winner strategy.

Figure 5: The average test error for the Australian Credit Card using NCL (top) and RTQRT-NCL (bottom) obtained by λ value using an ensemble size of 10. Top: vote strategy, Middle: average strategy, and Bottom: winner strategy.

respectively.

For the vote strategies in particular, it is clear that even these increased lambda values do not exhaust the improvement available with RTQRT-NCL.

3.3 Comparisons

Table 1 summarizes these results. It shows the average and standard deviations of the error, averaged over the 6 different λ values. These are computed for each of the 5 different committee sizes used.

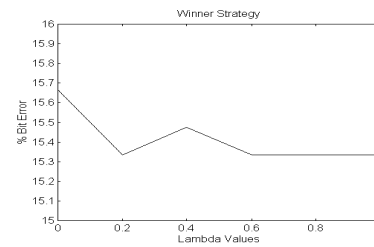
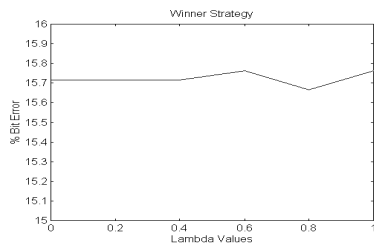
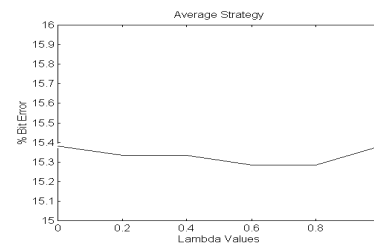
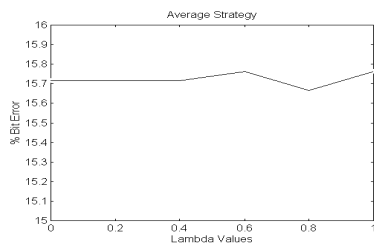
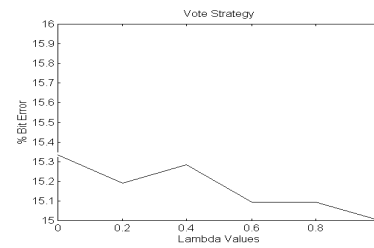
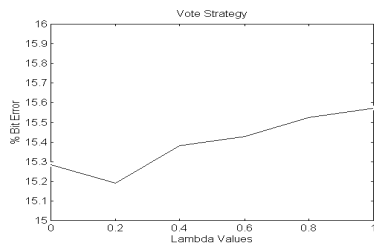


Figure 6: The average test error for the Australian Credit Card using RTQRT–NCL obtained by $m * \lambda$ value using an ensemble size of 2. Top: vote strategy; Middle: average strategy; and Bottom: winner strategy.

Figure 7: The average test error for the Australian Credit Card using RTQRT–NCL obtained by $m * \lambda$ value using an ensemble size of 4. Top: vote strategy; Middle: average strategy; and Bottom: winner strategy.

3.4 Discussion

On the Australian Credit Card dataset used in these experiments, as in the genetic programming examples reported below, the RTQRT–NCL penalty function outperformed the previous NCL penalty function. This is particularly the case with the 'voted' strategy, where except for the extreme case of a committee of 2, the individual differences are significant at better than the 80% confidence level (the differences being comparable with the sum of the standard

deviations). For the 'voted' strategy, the overall hypothesis, that for committee sizes greater than 2, RTQRT–NCL gives improved results compared with NCL, is significant at better than the 99% confidence level. Possible reasons for this improved performance will be discussed in Section 5

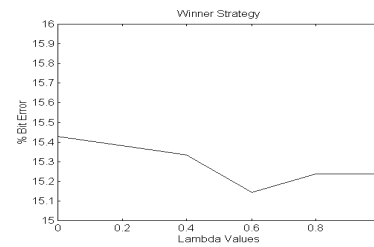
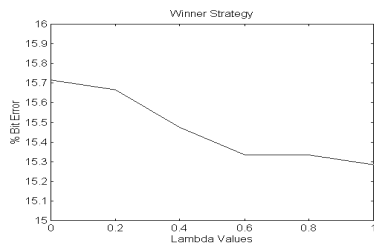
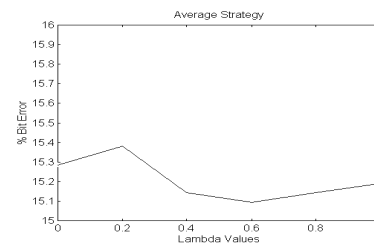
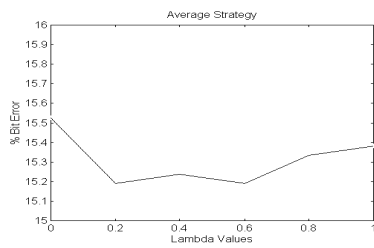
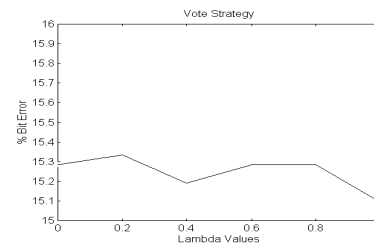
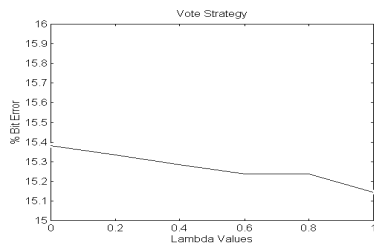


Figure 8: The average test error for the Australian Credit Card using RTQRT–NCL obtained by $m * \lambda$ value using an ensemble size of 6. Top: vote strategy; Middle: average strategy; and Bottom: winner strategy.

Figure 9: The average test error for the Australian Credit Card using RTQRT–NCL obtained by $m * \lambda$ value using an ensemble size of 8. Top: vote strategy; Middle: average strategy; and Bottom: winner strategy.

4 Experiment 2: Genetic Programming

4.1 Experimental Design

Our GP experiments use the 6-multiplexer problem:

```

EXPR → BOOL
BOOL → TERM
BOOL → and BOOL BOOL
BOOL → or  BOOL BOOL
BOOL → not BOOL
BOOL → if BOOL BOOL BOOL
TERM → a0
TERM → a1
TERM → d0
TERM → d1
TERM → d2
TERM → d3

```

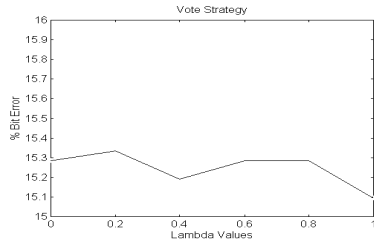
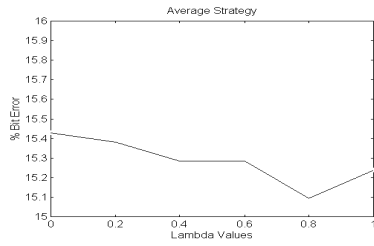
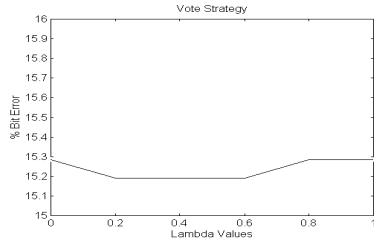


Figure 10: The average test error for the Australian Credit Card using RTQRT–NCL obtained by $m * \lambda$ value using an ensemble size of 10. Top: vote strategy; Middle: average strategy; and Bottom: winner strategy.

The 6-multiplexer problem is to predict, from the inputs, the outputs of a multiplexer having two address and four data lines. The search space is the set of boolean combinations of the address and data values using 'and', 'or', 'not' and three-way 'if' combinators. The GP parameters are listed in Table 2

The raw fitness was the proportion of the 64 cases correctly predicted. Runs were terminated at 200 generations. 20 experiments were conducted. Nine experiments used the

Table 1: The average and standard deviations $\times 10^{-3}$ of the error over different λ values for the 5 committee sizes used in the experiments

Committee size	Strategy	NCL	RTQRT–NCL
2	vote	153.49 ± 1.43	153.96 ± 1.43
	average	158.01 ± 0.55	157.22 ± 0.35
	winner	158.01 ± 0.55	157.22 ± 0.35
4	vote	154.04 ± 0.58	151.66 ± 1.26
	average	155.23 ± 0.99	153.33 ± 0.42
	winner	157.06 ± 1.06	154.12 ± 1.36
6	vote	155.31 ± 1.66	152.69 ± 0.83
	average	155.31 ± 0.97	153.09 ± 1.30
	winner	161.58 ± 11.40	154.68 ± 1.84
8	vote	154.28 ± 1.44	152.46 ± 0.87
	average	153.96 ± 1.07	152.06 ± 1.07
	winner	164.52 ± 24.85	152.93 ± 1.06
10	vote	154.20 ± 1.81	152.38 ± 0.52
	average	154.92 ± 0.49	152.85 ± 1.16
	winner	165.39 ± 27.73	152.93 ± 1.51

Table 2: Genetic Programming Parameters

Parameter	Value
Number of Runs	100
Generations/Run	100
Population Size (1st/later)	300/150
Max depth (initial pop)	8
Max depth (subsequent)	10
Tournament size	5
Crossover Probability	0.9
Mutation Probability	0.1

NCL measure in the fitness function, with λ ranging from 0.1 to 0.9. Nine equivalent experiments were performed with the RTQRT–NCL measure. One experiment used raw fitness; it is included in the graphs for NCL and RTQRT–NCL since it is equivalent to using these measures with a λ value of zero. Finally, the experiment was repeated using implicit fitness sharing as the fitness measure, since this is a common alternative diversity preserving mechanism in evolutionary computation.

4.2 Results

The population fitness by generation for NCL and RTQRT–NCL are shown in Figures 11 and 12.

It is clear even from a cursory glance that RTQRT–NCL error rates are significantly lower than NCL error rates over the range of values of λ , and that for the best values of λ for each, RTQRT–NCL gives a better performance than

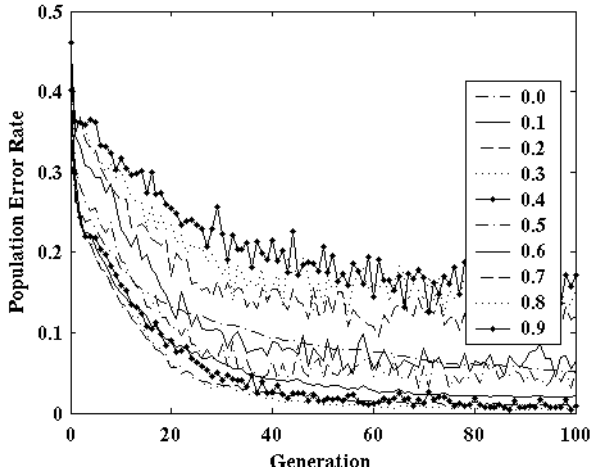


Figure 11: *Fitness vs Generation, NCL, for λ ranging from 0.0 to 0.9*

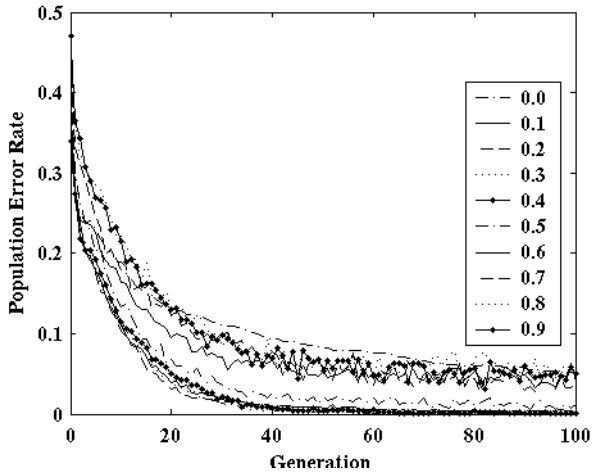


Figure 12: *Fitness vs Generation, RTQRT-NCL, for λ ranging from 0.0 to 0.9*

NCL. This is confirmed by Table 3, which shows, for each value of λ , the mean number of generations, out of the 100 generations in each run, where perfect accuracy on the 6 multiplexer was not achieved. It is thus a proxy measure of the expected number of generations before perfect accuracy is achieved.

These results are shown graphically in Figure 13. The equivalent value for implicit fitness sharing (IFS) is 31.11. Thus at least on this measure, RTQRT-NCL with a λ value between 0.6 and 0.9, outperforms IFS, though at the cost of an extra parameter (λ) which must be appropriately set. Figure 14 shows the error of raw fitness and IFS against generation. In comparison with figure 12, it appears to confirm that the optimal λ values for RTQRT-NCL out-perform IFS.

Table 3: Mean Proportion of Generations without Perfect Accuracy

λ	NCL	RTQRT-NCL
0	74.31	74.31
0.1	56.24	62.38
0.2	40.37	56.57
0.3	34.36	57.63
0.4	37.48	56.89
0.5	48.78	37.63
0.6	57.96	30
0.7	73.36	24.7
0.8	78.91	26.02
0.9	82.51	30.42

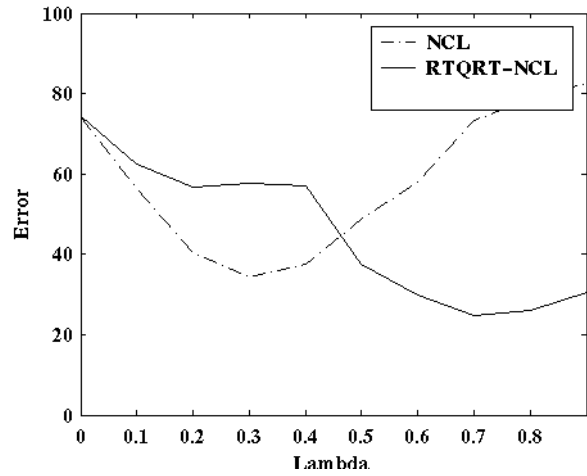


Figure 13: *Mean Proportion of Generations without Perfect Accuracy*

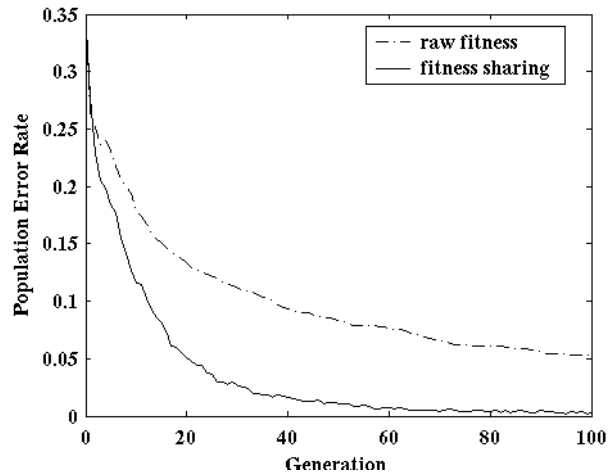


Figure 14: *Fitness vs Generation, raw fitness and implicit fitness sharing (IFS)*

4.3 Discussion

To summarize the results, with the appropriate choice of λ , root-quartic negative correlation learning out-performs

implicit fitness sharing, which in turn out-performs standard negative correlation learning. Both forms of negative correlation learning out-perform raw fitness except when extreme values of λ are specified. Once more, these results are highly attractive.

5 Theoretical Analysis

Our original motivation for pursuing this work was the unexpectedly poor performance of NCL with the larger committees used in genetic programming. The degree of dispersion of the committee was rather less than expected. As a result, we investigated a number of alternative measures of anti-correlation. The RTQRT-NCL function had similar advantages to NCL - that is, it is dimensionally consistent with mean square error, exerts a strong pressure for diversity, and its derivative, while more complex than NCL, is still sufficiently simple to be usable in the backpropagation algorithm.

However our choice of RTQRT-NCL was initially motivated by a further consideration about the penalty function used in NCL:

$$\Phi^p(m) = (\hat{Y}^p(m) - F^p) \sum_{l \neq m} (\hat{Y}^p(l) - F^p) \quad (8)$$

Noting that

$$\sum_{l \neq m} (\hat{Y}^p(l) - F^p) \quad (9)$$

can be re-written

$$\sum_l (\hat{Y}^p(l) - F^p) - ((\hat{Y}^p(m) - F^p)) \quad (10)$$

and

$$\sum_l (\hat{Y}^p(l) - F^p) = \sum_l ((\hat{Y}^p(m)) - M * F^p) = 0 \quad (11)$$

the equation is equivalent to:

$$\Phi^p(m) = -(\hat{Y}^p(m) - F^p)^2 \quad (12)$$

That is, the NCL penalty function acts to maximize the average difference between each network and the mean of the population. Yet the intended aim of anti-correlation mechanisms is to maximize the average difference between pairs of population members, which is not necessarily the same thing. Our aim with the RTQRT-NCL penalty was to derive a function which, while it was dimensionally equivalent to NCL, could not be simplified into a function of differences from the mean. We initially believed that, because of cross-terms, the RTQRT-NCL penalty function could not be so simplified. And we still believe that this is a desirable characteristic of a penalty function for anti-correlation. However after the experimental work

described above had been carried out, we discovered that the RTQRT-NCL penalty function could be so simplified. In fact,

$$\Phi^p(m) = \sqrt{\frac{(2M \sum_{l=1}^M (\hat{Y}^p(m) - F^p)^4 + 6(\sum_{l=1}^M (\hat{Y}^p(m) - F^p)^2)^2)}{M}} \quad (13)$$

Thus we were left without an acceptable explanation of the very good performance of RTQRT-NCL. In seeking such an explanation, we were led to consider the shape of the gradient of the RTQRT penalty function. While this function is highly complex, and difficult to analyze in general, consideration of a specific case may help to illuminate its behaviour.

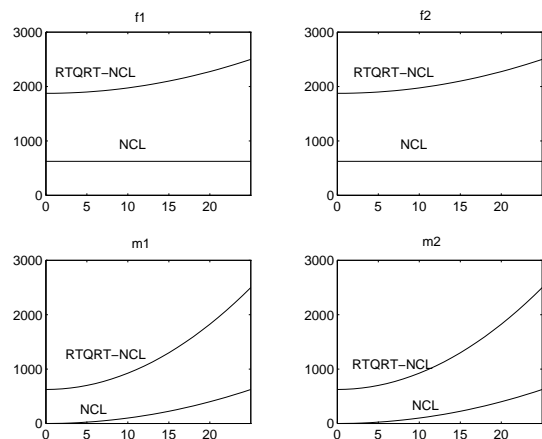
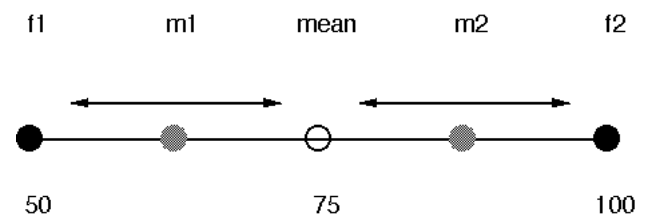


Figure 15: Points $f1$ and $f2$ are fixed, points $m1$ and $m2$ move symmetrically about the mean.

Consider the situation in Figure 15. In this situation, points $m1$ and $m2$ are free to move symmetrically about the fixed mean, while points $f1$ and $f2$ are fixed. Figure 15

6 Conclusion

For the particular case of the Australian credit card data set, the new RTQRT-NCL penalty function significantly outperforms the NCL penalty function for neural net committee learning, especially for larger committees. This is consistent with results obtained on the 6-multiplexer problem in genetic programming, where a similar result was obtained. We plan to extend the experiments to cover a wider range of standard datasets, and also to investigate a number of alternative promising penalty functions.

References

- Blake, C. and C. Merz (1998). UCI repository of machine learning databases, <http://www.ics.uci.edu/~mllearn/mlrepository.html>. University of California, Irvine, Dept. of Information and Computer Sciences.
- Liu, Y. and X. Yao (1999a). Ensemble learning via negative correlation. *Neural Networks* 12(10), 1399–1404.
- Liu, Y. and X. Yao (1999b). Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 29(6), 716–725.
- Liu, Y., X. Yao, and T. Higuchi (2000). Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation* 4(4), 380–387.
- Michie, D., D. Spiegelhalter, and C. Taylor (1994). *Machine learning, neural and statistical classification*. Ellis Horwood.

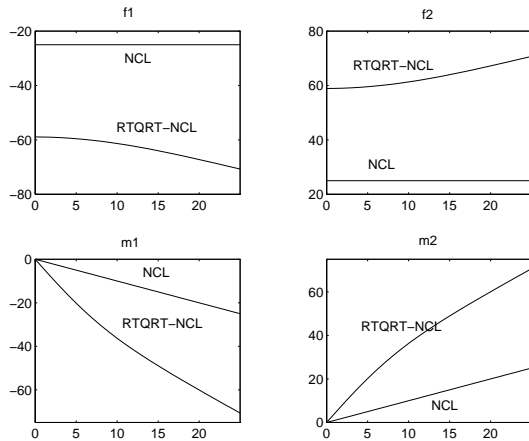


Figure 16: The gradient of f_1 , f_2 , m_1 , and m_2 for each position of m_1 and m_2 .

shows the value of the RTQRT and NCL penalty functions for each point, with respect to the position of points m_1 and m_2 .

Figure 16 shows the effect of movement of points m_1 and m_2 on the gradient of the penalty functions at the relevant points. The NCL gradients are constant for f_1 and f_2 , and linear for m_1 and m_2 , while those for the RTQRT-NCL penalty function are concave upwards in each case.

Thus while the learning pressure exerted by NCL increases linearly with distance from the mean, that exerted by RTQRT-NCL is smaller close to the mean, but increases superlinearly with distance from the mean. Thus RTQRT-NCL's penalty function serves to increase the pressure for already separated points to move apart, without also increasing the pressure for nearby points to separate. That is, when compared with NCL, RTQRT favours the creation of widely separated, but small, clusters. This, it is hypothesized, is a useful learning characteristic, at least for the Australian Credit Card dataset and the 6 multiplexer problem.