Non-Redundant Genetic Coding of Neural Networks

Dirk Thierens
Department of Computer Science
Utrecht University
PO Box 80089, 3508 TB Utrecht
The Netherlands
dirk.thierens@cs.ruu.nl

Abstract

Feedforward neural networks have a number of functional equivalent symmetries that make them difficult to optimise with genetic recombination operators. Although this problem has received considerable attention in the past, the proposed solutions have all a heuristic nature. Here we discuss a neural network genotype representation that completely eliminates the functional redundancies by transforming each neural network to its canonical form. This transformation is computationally extremely simple since it only requires flipping the sign of some of the weights, followed by sorting the hidden neurons according to their bias. We have compared the redundant and the non-redundant representation on the basis of their crossover correlation coefficient. As expected the redundancy elimination results in a much higher crossover correlation coefficient, which shows that more information is now transmitted from the parents to the children. Finally, experimental results are given for the two spirals classification problem.

1 Introduction

Several authors have shown that single hidden layer feedforward neural networks are universal approximators for any continuous mapping. Unfortunately these existence proofs give very little guidance on how to construct a good network for a specific problem. The network design problem can be decomposed into two subtasks. First we have the problem of network architecture: how many hidden layers do we need, how many neurons should we use in each hidden layer, and what connectivity will give us an optimal performance? Second we have the problem of weight determination: how do we determine the connection weights once we have chosen a particular network topology?

The network design problem can easily be defined as an optimisation problem, and therefore one can use a genetic algorithm to try to solve it. Using a GA requires to define a genotype coding of neural networks: unfortunately the existence of a structural-functional redundancy in the neural network representation hampers the application of the genetic recombination operators. This problem has also been called the *competing conventions* problem (Schaffer, Whitley & Eshelman, 1992).

Several authors have recognised this redundancy problem but the proposed solutions have either a heuristic nature (Hancock, 1992; Radcliffe, 1993; Thierens et al., 1993), or they have severely reduced and even eliminated the role of crossover in the genetic algorithm (de Garis, 1990; Parisi, Cecconi & Nolfi, 1990; Whitley et al., 1990), or the problem is just ignored (Maniezzzo, 1994), or finally claims have been made that the genetic algorithm is not suited for searching the weight space and one should use evolutionary programming techniques since they only use mutation (Fogel, Fogel & Porto, 1990; Angeline, Saunders & Pollack, 1994).

In this paper we give a simple non-heuristic procedure to transform multi-layer perceptrons such that the structural-functional redundancy problem is eliminated. Section 2 reviews the representational redundancy issue. Section 3 discusses the actual neural network transformation. Section 4 compares the redundant and non-redundant coding by computing their crossover correlation coefficients and their performance on the two spirals classification problem.

2 Background

The functional mapping implemented by a multi-layer perceptron is not unique to one specific set of weights. For a network with a single hidden layer, the mapping from the m-dimensional input vector X to the p-dimensional output vector Y is defined as (using the hyperbolic tangent $tanh$ as transfer function):

$$Y = tanh(Wtanh(VX))$$

where V is the (n,m)-dimensional matrix of weights from the input layer to the hidden layer, and W the (p,n)-dimensional matrix of weights from the hidden layer to the layer.

Obviously we can generate a number of structurally different neural nets that have the same input-output mapping. What characterises these networks is that they all are a member of a finite group of symmetries defined by two transformations. (Sussmann,1992; Chen, Lu, & Hecht-Nielsen, 1993). Any member of this group can be constructed from any other member by a sequence of these transformations. The first transformation is a permutation of hidden neurons. Interchanging the hidden neurons including their incoming and outgoing connection weights does not change the functional mapping of the network. The second transformation is obtained by flipping the weight signs of the incoming and outgoing connection weights of a hidden neuron. Since *tanh* is an odd symmetric function this sign flipping leaves the overall network mapping unchanged.

For a network with a single hidden layer of n neurons there are a total of $n!$ permutations. Similarly we can choose any combination of the n hidden neurons to flip their weight signs so there are $\sum_{i=0}^{n} \binom{n}{i} = 2^n$ structurally different but functionally identical networks generated by this transformation. Since the two transformations are independent of each other, there are a total of $2^n n!$ functional equivalent but structurally different networks. In (Sussmann, 1992; Chen, Lu, & Hecht-Nielsen, 1993) it is proven that all the functionally equivalent neural networks are compositions of hidden node permutations and sign flips.

For the traditional local weight optimisation algorithms this redundancy poses no problem since they only look in the immediate neighbourhood of the current point of the search space. Global optimisation algorithms however will try to explore the whole connection weight search space and this is a factor $2^n n!$ bigger than it really ought to be. For the genetic algorithm the problem is not only one of scale but also of crossover efficiency: functional equivalent near optimal networks often give rise to totally inappropriate networks after straightforward recombination because their weight structure is only equivalent up to a certain amount of transformations.

In the next section we give a simple non-heuristic procedure to transform multi-layer perceptrons such that the structural-functional redundancy problem is eliminated.

3 Non-redundant genetic coding

In the previous section we have seen that all functional equivalent multi-layer perceptrons form a finite group of networks that can be transformed into each other by a composition of hidden node permutations and sign flips. The functional redundancies can thus be eliminated if we transform each neural network to a canonical form that has a unique representation in each functional equivalence class. This non-redundant representation can be achieved in a number of ways. For instance to eliminate the hidden node redundancy we can simply flip the signs of the bias weight, the incoming and the outgoing weights of each hidden neuron whenever the number of positive incoming and outgoing weights of a neuron in the hidden layer is less than the number of negative incoming and outgoing weights. Alternatively, we might flip the signs whenever the bias weight is negative, so only hidden neurons with a positive bias are allowed in the non-redundant neural network representation, thus reducing the 2^n functional equivalent neural networks to just one representative of the group. Because of its extreme simplicity we will use the last approach here.

The second redundancy can be found at the level of each hidden layer and is caused by the permutation of the hidden nodes in each layer. To eliminate this redundancy we rearrange all hidden neurons in each layer such that the bias weights are sorted in ascending order. This way the $n!$ functional equivalent networks are eliminated and since the sort process does not interfere with the previous sign flipping all the $2^n n!$ equivalent networks are transformed to a single group representative.

The genotype-phenotype mapping is now a one-to-one mapping: each neural network within a functional equivalence class has now one unique genotype. These unique genotypes can now be recombined by the crossover operator since the competing conventions problem has been eliminated. To summarise, the following transformations are made to each neural network before they are recombined:

Neural network transformation:

1. $\forall$ hidden neurons:
   if (bias < 0)
   flip signs of each node weight
2. $\forall$ hidden layer:
   sort neurons in increasing bias order

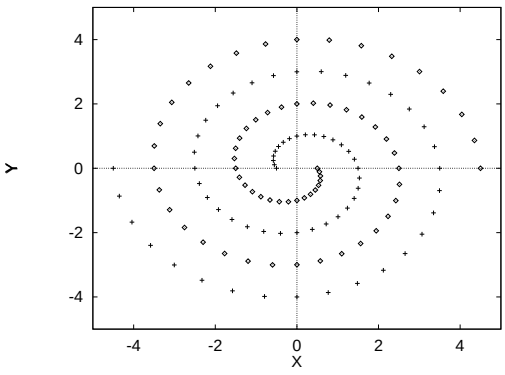

Figure 1: *Training data for the two spirals classification problem.*

4 Computational results

4.1 Two spirals classification problem

Since the number of functional equivalent neural networks is equal to $2^n n!$ for n hidden neurons, the effect of the representational redundancy on the genetic search will be more pronounced when the number of hidden neurons is high. We have tested the non-redundant genotype on the well known two spirals classification problem. The task is to learn to discriminate between two sets of training points which lie on two intertwined spirals in the x-y plane (see Figure 1). This is especially demanding for neural networks with non-local transfer functions such as multi-layer perceptrons, and requires a relatively large number of hidden neurons.

Elimination of the structural redundancies from the genotype representation ensures that the crossover operator transmits more information from the parent strings to the offspring. This information preservation can be quantified by comparing the crossover correlation coefficient for the redundant and non-redundant genotype coding.

In the next paragraph we compute the crossover correlation coefficient on the two spirals problem, and in Section 4.3 we look at the effect of the redundancy elimination on the actual neural network training process.

4.2 Crossover correlation coefficient

The crossover correlation coefficient is a statistical feature expressing how correlated the fitness landscape appears to the crossover operator (Manderick, de Weger, & Spiessens, 1991). The fitness landscape is defined by the combination of the fitness function and the specific genotype coding. The idea is that the more correlated a landscape appears to be for a specific operator the more efficient the GA search will be because the higher the correlation coefficient the more information is transmitted from the parents to the children.

To calculate the crossover correlation coefficient ρ_X one randomly generates a large number of parents, applies the crossover operator to obtain the offspring and computes the correlation coefficient as:

$$\rho_X(F_p, F_c) = \frac{cov(F_p, F_c)}{\sigma(F_p)\sigma(F_c)}$$

where F_p and F_c are respectively the random variables representing the mean fitness of each parent and offspring pair, $cov(F_p, F_c)$ is the covariance between F_p and F_c, and $\sigma(F_p)$ and $\sigma(F_c)$ are the standard deviations.

Previous studies used the crossover correlation coefficient to compare different crossover operators for a given genotype coding (Manderick, de Weger & Spiessens, 1991; Mathias & Whitley, 1992). Here we take the opposite approach: instead of comparing different crossover operators, we fix the crossover operator and compare the genotype representation. We have used one-point crossover that is restricted to exchange complete hidden neurons (see below). Since the fitness landscape is determined by the combination of the fitness function and the specific genotype coding, the crossover correlation coefficient ρ_X can also be used to see how well the crossover operator can transmit information from parents to offspring for different genotype codings.

Table 1 shows the crossover correlation coefficient ρ_X of the redundant and the non-redundant genotype coding for two different neural network topologies: the first has one hidden layer with 15 hidden neurons, while the second has two hidden layers, respectively with 15 and 5 hidden layers.

The crossover correlation coefficient is computed by recombining 2500 randomly generated parent pairs. The same parents and the same crosspoint is taken for the two network representations, the only difference is that for the non-redundant genotype the parents are first transformed into non-redundant form before being crossed. Clearly the crossover correlation coefficient ρ_X for the non-redundant neural network representation is much higher, indicating that crossover transmits much more information from the parents to the children, and thus will lead to a more efficient GA search.

4.3 Hybrid GA+BPX algorithm

Comparing the crossover correlation coefficients of the redundant and non-redundant coding shows that by eliminating the structural redundancies from the genotype representation, the recombination of neural networks becomes more efficient.

NNs	*redundant*	*non-redundant*
2-15-1	0.456	0.892
2-15-5-1	0.598	0.903

Table 1: *Crossover correlation coefficient ρ_X for the redundant and non-redundant neural network genotype representation. Results are computed for networks with one hidden layer of 15 neurons, and for two hidden layers with respectively 15 and 10 hidden neurons.*

To see the effect on the actual learning process we have tested both genotype codings with a hybrid genetic-backpropagation algorithm. Genetic algorithms work well when meaningful substructures can be recombined by the crossover operator. For neural networks these meaningful substructures are formed by the hidden neurons, so we restrict crossover to exchange only the incoming weights and bias of each neuron as a whole. The incoming weights and bias of each neuron thus cannot be separated by the crossover operator. This way the hybrid GA+BPX algorithm decomposes the optimisation task into two subtasks: a combinatorial search of entire hidden neurons by the genetic algorithm and a local gradient search of the neuron's weights by the backpropagation algorithm.

4.3.1 Elitist recombination GA

The specific GA implementation used is the elitist recombination GA because it is ideally suited for hybrid GA algorithms and it has the same selective pressure as the well known tournament selection scheme (Thierens & Goldberg, 1994). The elitist recombination GA intertwines the selection and recombination phase by holding a competition between the mating parents and their offspring. This local elitist mechanism ensures that population members can only be replaced by better solutions, and since strings are not copied by a separate selection process there is no danger of premature convergence as is the case with some of the more traditional elitist strategies.

Elitist Recombination GA:

1. initialise population
2. $\forall$ generation
   (a) randomly shuffle population
   (b) $\forall$ mating pair:
   - generate two offspring
   - keep best two of the four individuals

The elitist recombination GA allows us to integrate the local search algorithm in an elegant way. Every generation the population members are locally optimised for a few steps, and are then recombined to generate new offspring. Before the children compete with the parents to enter the population they also are locally optimised. Due to the elitism the local search is automatically focused on the current best solutions: good solutions are generally harder to replace by the children and will remain in the population for a number of generations thus receiving more local optimisation steps. Bad solutions are replaced very quickly so no expensive gradient search is wasted on them. In the worst case the problem, its coding and the crossover operator are totally incompatible, so recombination and the subsequently local search never produce any children better than their parents. As a result no new solutions will ever enter the population and the hybrid algorithm reduces to a multistarted local search algorithm where the starting points are simply the initial random population.

4.3.2 Accelerated backpropagation

The weight gradient optimisation is done using the accelerated backpropagation algorithm (BPX) which uses a momentum and adaptive learning rate (Vogl, Mangis, Rigler, Zink & Alkon, 1988). The parameters were set to their default value (momentum = 0.9, initial learning rate = 0.01, learning rate increase factor = 1.05, learning rate decrease factor = 0.7 and maximum error ratio = 1.04). The neurons had a hyperbolic tangent transfer function and were initialised by the Nguyen-Widrow procedure which places the linear part of the hyperbolic tangent function within the input space region where training data are present (Nguyen & Widrow, 1990).

4.3.3 Experimental results

To test the non-redundant neural network representation we compare it with the redundant representation starting from the same initial population of 30 neural networks. At each generation the parents are optimised by the accelerated backpropagation algorithm for 100 epochs and the children for 200 epochs.

Figure 2 plots the sum-squared error (SSE) of the best and mean performing neural network in the population at each generation. As expected from the crossover correlation measurements training is faster when using the non-redundant genotype representation: for instance after 25 generations the error of the best network in the non-redundant population is more than 10 times lower than the error of the best redundant neural net. A similar observation is made for the population average error.

These experimental results are of course very limited.

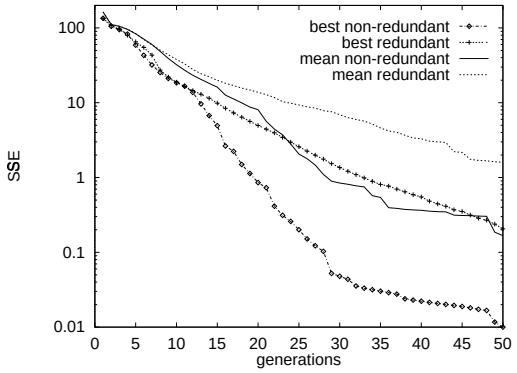

Figure 2: *Sum-squared error of the best and the mean performing neural network in the population at each generation when solving the two spirals classification problem with the hybrid GA+BPX algorithm.*

In future work we plan to do a more fully experimental analysis for the two-spirals problem and other benchmark problems. In addition it is important to study the effect that the local gradient search has in the overall optimisation task.

5 Conclusion

Feedforward neural networks have a number of functional equivalent symmetries that make them difficult to optimise with genetic recombination operators. We have shown how neural networks can be coded such that the functional redundancies are completely eliminated. The genotype coding is obtained by transforming the networks to their canonical form. This transformation is computationally extremely simple since it only requires flipping the sign of some of the weights, followed by sorting the hidden neurons according to their bias. We have compared the redundant and the non-redundant representation on the basis of their crossover correlation coefficient. As expected the redundancy elimination results in a much higher crossover correlation coefficient, which shows that more information is now transmitted from the parents to the children.

References

1. Angeline P., Saunders G. & Pollack J., An Evolutionary Algorithm that Constructs Recurrent Neural Networks. *IEEE Transactions on Neural Networks, Vol. 5, No. 1* pp.54-65, 1994.
2. Chen A., Lu H. & Hecht-Nielsen R., On the Geometry of Feedforward Neural Network Error Surfaces. *Neural Computation, Vol.5* pp.910-927, 1993.
3. de Garis H. Genetic Programming: modular neural evolution for Darwin machines. *Proceedings of International Joint Conference of Neural Networks.* 1990.
4. Fogel D.B, Fogel L.J. & Porto V.W., Evolving neural networks. *Biological Cybernetics*, Vol. 63, 1990.
5. Hancock P., Recombination Operators for the Design of Neural Nets by Genetic Algorithm. *Parallel Problem Solving from Nature PPSN-II.* eds. R. Männer & B. Manderick. pp.441-450. North-Holland, 1992.
6. Manderick B., de Weger M. & Spiessens P., The Genetic Algorithm and the Structure of the Fitness Landscape. *Proceedings of the Fourth International Conference on Genetic Algorithms.* eds. R. Belew & L. Booker. pp.143-150. Morgan Kaufmann, 1991.
7. Maniezzzo V. Genetic Evolution of the Topology and Weight Distribution of Neural Networks. *IEEE Transactions on Neural Networks* Vol. 5, No. 1, 1994.
8. Mathias K. & Whitley D., *Genetic Operators, the Fitness Landscape and the Traveling Salesman Problem.* Parallel Problem Solving from Nature PPSN-II. eds. R. Männer & B. Manderick. North-Holland, 1992.
9. Nguyen D. & Widrow B., Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *Proceedings of the International Joint Conference of Neural Networks, Vol3* 1990.
10. Parisi D., Cecconi F. & Nolfi S., Econets: neural networks that learn in an environment. *Network 1* pp.149-168, 1990.
11. Radcliffe N., Genetic set recombination and its application to neural network topology optimisation. *Neural Computing and Applications, Vol.1* pp.67-90, 1993.
12. Schaffer J.D., Whitley L.D. & Eshelman L.J, Combinations of genetic algorithms and neural networks: A survey of the state of the art. *Proceedings of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks* eds. L.D. Whitley and J.D. Schaffer. IEEE Computer Society Press, 1992.
13. Sussmann H.J., Uniqueness of the Weights for Minimal Feedforward Nets with a Given Input-Output Map. *Neural Networks, Vol.5* pp.589-593, 1992.
14. Thierens,D. & Goldberg,D.E., Elitist Recombination: an integrated selection recombination GA. *Proceedings of the IEEE World Congress on Computational Intelligence, Vol.1* pp.508-512, 1994.
15. Thierens D., Suykens J., Vandewalle J., & De Moor B., Genetic Weight Optimization of a Feedforward Neural Network Controller. *Proceedings International Conference on Artificial Neural Networks and Genetic Algorithms.* eds. Albrecht R., Reeves C., & Steele N. pp.658-663. Springer-Verlag, 1993.
16. Vogl T.P., Mangis J.K., Rigler A.K., Zink W.T. & Alkon D.L., Accelerating the Convergence of the Backpropagation Method. *Biological Cybernetics* Vol.59, 1988.
17. Whitley L.D., Starkweather T. & Bogart C., Genetic Algorithms and Neural Networks: optimizing connections and connectivity. *Parallel Computing* Vol.14, 1990.