Normalization for Neural Network in Genetic Search

Jung-Hwan Kim
School of Computer Science and Engineering
Seoul National University
Shilim-dong, Kwanak-gu, Seoul, 151-742 Korea
aram@soar.snu.ac.kr

Sung-Soon Choi
School of Computer Science and Engineering
Seoul National University
Shilim-dong, Kwanak-gu, Seoul, 151-742 Korea
sschoi@soar.snu.ac.kr

Byung-Ro Moon
School of Computer Science and Engineering
Seoul National University
Shilim-dong, Kwanak-gu, Seoul, 151-742 Korea
moon@soar.snu.ac.kr

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Genetic Algorithms

Keywords

Neural network, Isomorphism, Transformation, Normalization, Distance measure

1. INTRODUCTION

Genetic algorithms (GAs) are heavily used in neural network (NN) modeling for the evolution of connection weights, network architectures, learning rules, and control parameters of networks. The effectiveness of an evolutionary method for the neural network optimization highly depends on the genotype coding of neural networks.

An arbitrary neural network has a number of functionally equivalent other networks. This causes redundancy in genetic representation of neural networks, which considerably undermines the merit of crossover in GAs. This problem has received considerable attention in the past and has also been called the "competing conventions" problem [1, 2]. If the functional equivalence or redundant topological representation of the network is not considered in the genetic framework, it is hard to overcome the poor search capability of the neural network.

A typical multilayer neural network consists of input, output, and hidden layers. The neurons in the hidden layer enable the network to learn complex tasks by progressively extracting more meaningful features from the input patterns and to form the rules classifying the input patterns. Using a GA requires to define a genotype coding of neural networks: The weights of a neural network can be represented by a 2D matrix and thus they are intrinsically suitable for two-dimensional encoding. In this paper, we use a two-dimensional encoding for the genetic representation and employ 2D *geographic crossover* which demonstrated good performance for the neural network optimization problem and the graph partitioning problem. Once the structure of the neural network is fixed and a neural network is represented by a 2D encoding, the genotype of a neural network depends on the order of neurons in the structure. All the hidden neurons are identical when only the connections (without weights) are considered. They become different only after they start having weights in the connections.

Some hidden neurons have stronger relationships than those between ordinary pairs of hidden neurons. However, the relationship among the hidden neurons has little to do with neurons' physical placements. Since the indices of hidden neurons are assigned before they start having weights, there is no guarantee that strongly related neurons are assigned close indices one another. This may cause high-quality schemata to have scattered specific symbols and, consequently, not to survive well.

In this paper, we transform each neural network to an isomorphic neural network to maximize the genotypic consistency between two parents. We will show that the transformation can be done in polynomial time or is NP-hard depending on the distance measures between neural networks. We aim to develop a better genetic algorithm for neural network optimization by helping crossover better inherit common functional characteristics of the two parents. This is achieved by protecting "phenotypic" consistency and, consequently, preserving building blocks with promising schemata. We will show the experimental results with well known medical datasets for classification.

GECCO'05, June 25–29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

2. ISOMORPHISM AND NORMALIZATION

Given a neural network, let N_i, N_h, and N_o be the numbers of input neurons, hidden neurons, and output neurons in the network, respectively. And let $N = N_i + N_h + N_o$ be the total number of the neurons in the neural network. We assume that N_h is not too small relative to N, more formally, $N_h = \Theta(N)$. We denote a neural network $\mathfrak{N}$ by $\mathfrak{N} = \{\mathbf{h}_1, \ldots, \mathbf{h}_J\}$, where $\mathbf{h}_j = [w^i_{j1}, \ldots, w^i_{jN_i}, w^h_{1j}, \ldots, w^h_{N_h j}, w^o_{1j}, \ldots, w^o_{N_o j}]^T$ and w^i_{jk}, w^h_{kj}, and w^o_{kj} are the synaptic weights from input neuron k to hidden neuron j, from hidden neuron j to hidden neuron k, and from hidden neuron j to output neuron k, respectively. Let S_J be the set of all the permutations of the set $\{1, \ldots, J\}$ where J is the number of hidden neurons. We define the neural network isomorphism as follows:

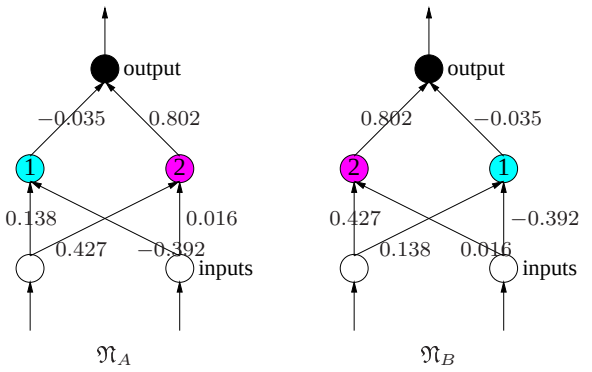

Figure 1: Two isomorphic neural networks

```
Normalize(𝔑,𝔑′)
{
    do {
        Q ← ∅;
        𝔑′_0 ← 𝔑′;
        for i ← 1 to N_h/2 {
            choose a, b ∈ {1, 2, ..., N_h} − Q
                such that 𝔡(𝔑, 𝔑′_{i−1}[a, b]) is minimal;
            Q ← Q ∪ {a, b};
            𝔑′_i ← 𝔑′_{i−1}[a, b];
        }
        choose k ∈ {0, 1, ..., N_h/2}
            such that 𝔡(𝔑, 𝔑′_k) is minimal;
        𝔑′ ← 𝔑′_k;
    } until (there is no improvement);
    return 𝔑′;
}
```

Figure 2: The KL-style heuristic for the normalization based on the measure $\mathfrak{d}$

Definition 1. For two neural networks $\mathfrak{N} = \{\mathbf{h}_1, \ldots, \mathbf{h}_J\}$ and $\mathfrak{N}' = \{\mathbf{h}'_1, \ldots, \mathbf{h}'_J\}$, $\mathfrak{N}$ is *isomorphic* to $\mathfrak{N}'$ ($\mathfrak{N} \simeq \mathfrak{N}'$) if and only if there exists a permutation $p \in S_J$ such that $\mathbf{h}_{p(j)} = \mathbf{h}'_j \ \forall j = 1, \ldots, J$.

From the definition, two isomorphic neural networks are constructed essentially in the same way. In other words, a neural network can be transformed to another isomorphic neural network by appropriate permutation of the hidden neurons.

In Figure 1, two neural networks $\mathfrak{N}_A$ and $\mathfrak{N}_B$ look different from each other. In other words, they have different representations. However, they are isomorphic because of their equivalent functionality; they output the exactly same value with respect to the same input vector.

If we perform crossover with two functionally similar solutions with significantly different shapes, the offspring will be significantly different from both parents. If we crossover the two solutions as they stand in appearance, the common semantic characteristics of the two neural networks are prone to be broken in the process of crossover. From a parent's point of view, the crossover is like too strong a mutation. We minimize this "visual inconsistency" before crossover.

We transform one of the parents in relation to the other so that high-quality schemata are well preserved and combined. We call such a transformation *normalization*. More formally, let $\mathcal{N}$ be the set of the neural networks and $\mathcal{N}_{\mathfrak{N}}$ be the set of the networks that are isomorphic to a network $\mathfrak{N} \in \mathcal{N}$. Suppose a distance measure $\mathfrak{d} : \mathcal{N} \times \mathcal{N} \to \mathbb{R}$ defined on a pair of networks that measures the genotypic distance of the two networks. Given parents $\mathfrak{N}, \mathfrak{M} \in \mathcal{N}$, the normalization operator transforms $\mathfrak{M}$ to $\mathfrak{M}' \in \mathcal{N}_{\mathfrak{M}}$ such that $\mathfrak{d}(\mathfrak{N}, \mathfrak{M}')$ is minimal among all the networks in $\mathcal{N}_{\mathfrak{M}}$. Selecting a suitable measure is crucial in the normalization operator design since the difficulty of the problem with respect to genetic algorithms considerably depends on the distance measure. It is important to design the measure so that high-quality schemata are well preserved and combined.

Figure 2 describes the heuristic for the normalization. In general, evaluating the distance $\mathfrak{d}(\mathfrak{N}, \mathfrak{N}')$ between given two networks $\mathfrak{N}$ and $\mathfrak{N}'$ consumes $\Theta(N^2)$ time from the definitions of the measures. By considering only the gains in the distance and updating the gains deliberately in the search process, we have the time complexity $O(N_h^3)$ for a pass of the outermost loop. Here we assumed $N_i + N_o = O(N_h)$. We observed that the number of iterations of the outermost loop is bounded by a small constant. Thus the typical complexity of the heuristic is $O(N_h^3)$.

Table 1 shows the classification results. In the table, "Base" represents the results of the neuro-genetic hybrid without normalization. $\mathfrak{d}_i$_B denotes the neuro-genetic hybrid with normalization by B on distance measure $\mathfrak{d}_i$. The two values in each experiment show the mean and the best classification results, respectively, from 50 trials. The normalization methods for the neural network overall showed improvement over the one without normalization. Among the five methods of normalization, the method $\mathfrak{d}_3$_KL showed the best results. This tendency was consistent in all of the three test problems. The results also showed that the learning degrees are more informative than the weights themselves.

Table 1: Comparison of Normalization Methods

	WBCD	CHDD	TDD
Base	96.47 (96.91)	80.61 (82.74)	94.54 (94.72)
$\mathfrak{d}_1$_2Opt	96.78 (97.42)	82.03 (85.79)	95.19 (96.44)
$\mathfrak{d}_1$_KL	96.72 (97.42)	82.16 (84.26)	95.16 (96.47)
$\mathfrak{d}_2$_HM	96.89 (97.42)	82.07 (83.82)	94.83 (94.97)
$\mathfrak{d}_3$_2Opt	97.03 (97.43)	83.36 (85.15)	95.65 (96.53)
$\mathfrak{d}_3$_KL	**97.21** (**97.60**)	**84.26** (**85.83**)	**95.77** (**96.85**)

Acknowledgment

This work was supported by grant No. (R01-2003-000-10879-0) from the Basic Research Program of the Korea Science and Engineering Foundation. This was also partly supported by the Brain Korea 21 Project. The ICT at Seoul National University provided research facilities for this study.

3. REFERENCES

[1] N. J. Radcliffe. Genetic set recombination and its application to neural network topology optimization. *Neural Computing and Applications*, 1:67–90, 1993.

[2] D. Thierens. Non-redundant genetic coding of neural networks. In *IEEE International Conference on Evolutionary Computation*, pages 571–575, 1996.