Neural Network Weight Space Symmetries Can Speed up Genetic Learning

ROMAN NERUDA[*]
Institue of Computer Science
Academy of Sciences of the Czech Republic
P.O. Box 5, 18207 Prague, Czech Republic
tel: (4202) 6605 3750, fax:(4202) 858 5789

Abstract: A functional equivalence of feed-forward networks has been proposed to reduce the search space of learning algorithms. A novel genetic learning algorithm for RBF networks and perceptrons with one hidden layer that makes use of this theoretical property is proposed. Experimental results show that our procedure outperforms the standard genetic learning.

Key-Words: Feedforward neural networks, genetic learning algoritms.

1 Introduction

We consider a feedforward network as a device for computing a real function of several real variables which depend on a set of network parameters (weights). A function realized by the network is referred to as an *input/output (or I/O) function* of the network. The logical question which functions can be approximated by a class of networks of a given type has been answered in recent years. A so called *universal approximation property* (the possibility to approximate any reasonable, e.g. continuous, function arbitrarily well) was examined. It has been proven that many common network architectures, including multilayer perceptrons and RBF networks which satisfy certain mild conditions on the activation or radial function, posses this property.

Thus, theoretically, for any reasonable function we can find a network of a given type that computes an arbitrarily close approximation of this function as its I/O function. It means that during the learning phase, the parameters of the network can be assigned in such a way that the desired function (usually described by a set of examples of input/output values) is approximated with arbitrary precision. In practice this typically requires to solve a non-linear optimization problem in the high-dimensional space of network parameters. This motivates one to search for possibilities to simplify this task. One of the approaches is to reduce the search space by identifying the classes of functionally equivalent networks and by selecting a single representative of each class. An algorithm which is able to restrict the learning only to these representatives operates on much smaller search space and thus may perform faster.

Hecht-Nielsen [3] pointed out that characterization of functionally equivalent network parameterizations might speed up some learning algorithms. Several authors studied functionally equivalent weight vectors for one hidden layer perceptron-type networks with various activation functions ([1], [5], [10]). In [6], [8] we have characterized the form of functional equivalence of Radial Basis Function networks with Gaussian ra-

[*]This work has been partially supported by the Grant Agency of the Czech Republic under grants no. 201/99/P057, 201/99/0092, and 201/01/1192.

dial function. The notion of unique parameterization property has been proposed by Kůrková to summarize results common to important non-trivial network architectures.

2 Functional equivalence

From now on we consider two types of feed-forward network architectures. By an *RBF network* we mean the feed-forward network with one hidden layer containing radial-basis-function (RBF) units with a radial function $\psi : \mathcal{R}_+ \to \mathcal{R}$ and a metrics ρ on $\mathcal{R}^n$ (n is the number of input units) and with a single linear output unit. Such a network computes the function:

$$f(\mathbf{x}) = \sum_{i=1}^{k} w_i \psi\left(\frac{\rho(\mathbf{x}, \mathbf{c}_i)}{b_i}\right). \tag{1}$$

Here, the *perceptron network* means a feed-forward network with n inputs, one hidden layer containing perceptron units and one linear output unit. This network computes the function $f : \mathcal{R}^n \to \mathcal{R}$ of the form:

$$f(\mathbf{x}) = \sum_{i=1}^{k} w_i \psi(\mathbf{v}_i \mathbf{x} + b_i), \tag{2}$$

where $k \in \mathcal{N}$ is the number of hidden units, $w_i, b_i \in \mathcal{R}$, $\mathbf{v}_i \in \mathcal{R}^n$ and $\psi : \mathcal{R} \to \mathcal{R}$ is an activation function.

A *radial-basis-function network parameterization with respect to* (ψ, n, ρ) is a sequence $\mathbf{P} = (w_i, \mathbf{c}_i, b_i;\ i = 1, \ldots, k)$, where k is the number of hidden units and for the i-th hidden unit the vector $\mathbf{c}_i \in \mathcal{R}^n$ describes the centroid while the real numbers b_i and w_i are widths and output weights, respectively (see(1)). If additionally, for every $i \in \{1, \ldots, k\}$ $w_i \neq 0$, and for every $i, j \in \{1, \ldots, k\}$, such that $i \neq j$ either $\mathbf{c}_i \neq \mathbf{c}_j$ or $b_i \neq b_j$, it is called *a reduced parameterization*. Similarly, a *perceptron network parameterization with respect to* (ψ, n) is a sequence $\mathbf{Q} = (w_i, \mathbf{v}_i, b_i;\ i = 1, \ldots, k)$ with the meaning of symbols described by (2). Additionally, if for every $i \in \{1, \ldots, k\}$ $w_i \neq 0$, and for every $i, j \in \{1, \ldots, k\}$ $i \neq j$ implies that either $\mathbf{v}_i \neq \mathbf{v}_j$, or $b_i \neq b_j$ and there exists at least one i such that $\mathbf{v}_i = \mathbf{0}$, it is called *a reduced parameterization.*

It is clear that a parameterization $\mathbf{P}$ (or $\mathbf{Q}$) determines a unique I/O function of an RBF network according to the formula (1) (or a perceptron network according to (2)).

Two network parameterizations $\mathbf{P}$ and $\mathbf{P}'$ are *functionally equivalent* if they determine the same input/output function. Two network parameterizations are called *interchange equivalent,* if $k = k'$ and there exists a permutation π of the set $\{1, \ldots, k\}$, such that for each $i \in \{1, \ldots, k\}$ $w_i = w'_{\pi(i)}$ and $b_i = b'_{\pi(i)}$ and $\mathbf{c}_i = \mathbf{c}'_{\pi(i)}$ for RBF network parameterizations, or $\mathbf{v}_i = \mathbf{v}'_{\pi(i)}$ for perceptron network parameterizations.

We are interested in relationship between the functional equivalence and interchange equivalence. Clearly the later implies the former, so it is the non-trivial reverse implication that is in our focus.

Let $n \in \mathcal{N}$. Function ψ has a *unique parameterization property* with respect to n, if for every two reduced parameterizations of perceptron networks w.r.t. (ψ, n) (or RBF networks w.r.t. (ψ, n, ρ)) functional equivalence implies interchange equivalence.

The most general characterization of functions satisfying the unique parameterization property of perceptron networks is due to [5].

Theorem 1 *Let ψ be bounded, non-constant and asymptotically constant activation function, $n \in \mathcal{N}$. Then ψ has a unique parameterization property of perceptron networks with respect to n, if and only if it is neither self-affine, nor affinely recursive.*

Many popular activation functions, including logistic sigmoid or Gaussian, are not affinely recursive. On the contrary, polynomials are affinely recursive, so they do not posses the unique parameterization property. Self-affinity requires a finer analysis which is described in the original paper. Roughly speaking, trivial parameter changes such as sign flips also have to be taken into account.

In the case of RBF networks, the standard choice of a radial function is Gaussian and the most popular metrics are those induced by various inner products (such as Euclidean), or the maximum metrics. Our previous results [6] show that the unique parameterization property is satisfied in these cases.

Theorem 2 *Let n be a positive integer, ρ metrics on $\mathcal{R}^n$ induced by an inner product, or a maximum metrics. Then $\gamma(t) = \exp(-t^2)$ has a unique parameterization property of a corresponding RBF network.*

Preceding theorems enables to describe a canonical representation of a network computing a particular function easily. One of the possible choices is to impose a condition on a parameterization that weight vectors corresponding to hidden units are increasing in a lexicographic ordering on a parameterization. Represent a parameterization $\{w_i, k_i, \mathbf{c}_i; i = 1, \ldots, k\}$ or $\{w_i, k_i, \mathbf{v}_i; i = 1, \ldots, k\}$ as a vector $\mathbf{p} = \{\mathbf{p}_i, \ldots, \mathbf{p}_k\} \in \mathcal{R}^{k(n+2)}$, where $\mathbf{p}_i = \{w_i, b_i, c_{i1}, \ldots, c_{in}\} \in \mathcal{R}^{n+2}$, or $\mathbf{p}_i = \{w_i, b_i, v_{i1}, \ldots, v_{in}\} \in \mathcal{R}^{n+2}$, are weight vectors corresponding to the i-th hidden RBF, or perceptron, unit. Let $\prec$ denotes the lexicographic ordering on $\mathcal{R}^{n+2}$, i.e. for $\mathbf{p}, \mathbf{q} \in \mathcal{R}^{d+2}$ $\mathbf{p} \prec \mathbf{q}$ if there exists an index $m \in \{1, \ldots, d+2\}$ such that $p_j = q_j$ for $j < m$, and $p_m < q_m$. We call a network parameterization $\mathbf{P}$ *canonical* if $\mathbf{p}_1 \prec \mathbf{p}_2 \prec \ldots \prec \mathbf{p}_k$.

In this terminology, theorems 1 and 2 guarantee that for every network parameterization a canonical parameterization determining the same input-output function exists. Thus, the set of canonical parameterizations corresponds to a minimal search set—weight space subset containing exactly one representative of each class of functionally equivalent weight vectors—proposed in [3].

An important extension of our previous results concerning the approximate version of functional equivalence exists. The strict functional equivalence, as was introduced, may not be a sufficient answer in many problems. Especially when dealing with approximation problems we are interested in the case where two functions realized by networks are not exactly the same but their difference is small. This case is also of a big importance in practical problems when we, by nature, operate with inexact values.

Let us focus on the case of RBF networks first. Define a function $\mathcal{F}_c^R$ operating on a set of canonical parameterizations and assigning each parameterization the corresponding network I/O function. Moreover, we restrict the space by bounding all the parameters by an arbitrary but fixed constant $B \in \mathcal{R}$. We will also suppose that the number of hidden units is bounded by a number $K \in \mathcal{N}$. Thus, without a loss of generality, we can consider all the parameterizations to be elements of a compact hypercube $[-B; B]^{K(n+2)}$. Shorter parameterizations whose $k \leq K$ are simply padded by a sufficient number of zeros.

The factor space $\mathcal{S}$ defined as: $\mathcal{S} = [-B; B]^{K(n+2)} / \sim$ can be considered identical to the space of all canonical parameterizations.

$\mathcal{F}_c^R$ is then defined as: $\mathcal{F}_c^R : \mathcal{S} \to \mathcal{C}([-B; B])$.

$$\mathcal{F}_c^R(\mathbf{P}) = \sum_{i=1}^{K} w_i \psi\left(\frac{\rho(\mathbf{x}, \mathbf{c}_i)}{b_i}\right) \qquad (3)$$

The following theorem states that the inverse of $\mathcal{F}_c^R$ is continuous, which means that a small change of the function introduces only a small change in the parameterization.

Theorem 3 *Let B be a positive real constant, n, K positive integers, $\mathcal{F}_c^R$ be defined by (3) where $\psi(t)$ is a continuous and bounded radial function, and ρ is ρ_{E}. Then $(\mathcal{F}_c^R)^{-1}$ is continuous.*

Analogously, we can define the function $\mathcal{F}_c^P : \mathcal{S} \to \mathcal{C}([-B; B])$.for perceptron networks and derive the following counterpart of theorem 3.

$$\mathcal{F}_c^P(\mathbf{P}) = \sum_{i=1}^{k} w_i \psi(\mathbf{v}_i \mathbf{x} + b_i) \qquad (4)$$

Theorem 4 *Let B be a positive real constant, n, K positive integers, $\mathcal{F}_c^P$ be defined by (4) where $\psi(t)$ is a continuous and bounded function. Then $(\mathcal{F}_c^P)^{-1}$ is continuous.*

3 Geometrical View

Kůrková and Kainen in [5] have shown an interesting property that stresses the importance of (perceptron) networks with one input. Roughly speaking, they proved that—considering perceptron networks with an asymptotically constant activation function—if functional equivalence implies interchange equivalence for one-input networks, the same holds for a network with any number of inputs. This means that we can limit our investigations to a one-dimensional case only.

This result is in interesting accordance with a dimension reduction theorem used in some proofs of the universal approximation property, e.g. in [7]. No such result has been proven in the case of RBF networks, so far, and it is an open problem whether such a relationship holds.

Let us illustrate the problem from the geometrical point of view. If a function ψ of a single argument $x \in \mathcal{R}$ does not satisfy the property of the functional equivalence implying the interchange equivalence it can be expressed as:

$$\psi(t) = \sum_{i=1}^{k} w_i \psi(v_i t + b_i) + c\,, \tag{5}$$

where $w_i \neq 0$, $v_i \neq 0$ and if $i \neq j$, then either $v_i \neq v_j$, or $b_i \neq b_j$, and moreover, if $b_i = 0$, then $v_i \neq 1$. If k in the formula (5) equals 1, we call ψ *self-affine*. If (5) is satisfied for $k \geq 2$, ψ is called *affinely recursive*.

Functions which are odd or even represent simple examples of self-affine functions. Affinely recursive functions can be represented as a linear combination of their shifted and/or translated copies. Imagine for a moment the most simple RBF and perceptron networks with piecewise linear activation function and single input. Figure 1 shows two networks that are functionally equivalent. These networks differ not only in parameters values, but also in the number of hidden units. Geometrically we compose a graph of one hat function by means of its three copies.

Similar example for the case of two different functionally equivalent perceptron networks and their corresponding functions is shown in Figure 2. A shift to a higher dimensional input space is quite straightforward for this model. So, in the perceptron case we are able to compose one step out of several smaller steps independently of the input dimension. On the other hand, in the RBF case, the situation is more complicated. First, the form of a function in 2-D differs depending on the metrics chosen. For Euclidean metrics we get cones, while maximum or Manhattan metrics give square pyramids. Nevertheless, it is not clear for any of those metrices how to compose one pyramid out of larger number of its copies.

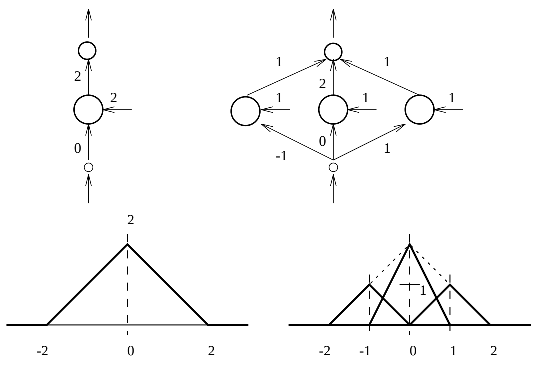

Figure 1: Functionally equivalent RBF networks and corresponding hat functions

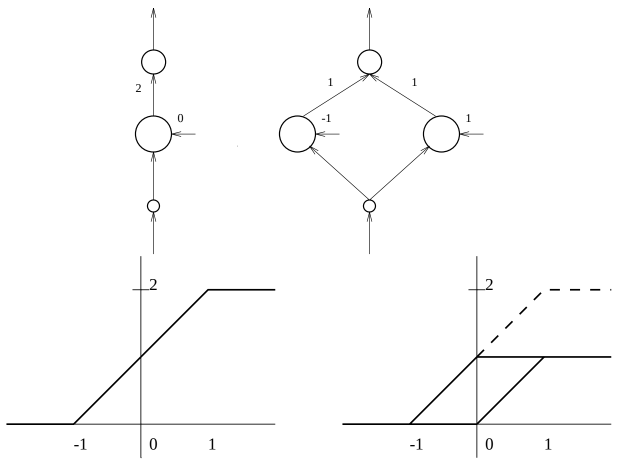

Figure 2: Functionally equivalent MLP networks and corresponding step functions

4 Canonical genetic algorithm

In order to take advantage of the previous results a learning algorithm that can operate only on canonical parameterizations is needed. Unfortunately, neither back propagation nor more complicated three step learning algorithms of RBF networks (cf. [4]) are suitable, since the analytical solution obtained in each step of the iterative process cannot in principle be limited to a certain weight space subset. This is not so with genetic algorithm whose operations can be changed to preserve the property of being canonical.

The core of canonical GA is the same as usual: In the beginning a population of m canonical parameterizations $\mathcal{P}_0 = \{\mathbf{P}_1, \ldots, \mathbf{P}_m\}$ is generated at random. Having population $\mathcal{P}_i$, the successive population $\mathcal{P}_{i+1}$ is generated by means of three basic genetic operations: *reproduction*, *crossover* and *mutation* that are proposed such that they generate only canonical parameterizations.

To generate the *initial population* of canonical parameterizations at random, one has to preserve the property that for each parameterization $\mathbf{P}$ it holds: $\mathbf{p}_s \prec \mathbf{p}_{s+1}$.

The *reproduction* operator represents a probabilistic selection of parameterization $\mathbf{P_l} \in \mathcal{P}_i$ according to the values of objective function $\mathcal{G}(\mathbf{P_l})$ which is computed by means of the error function (i.e. the sum of distances between the actual and desired output of the network over all of the patterns from the training set). We use the roulette wheel selection together with the elitist mechanism.

The *mutation* operates on two levels—first an element $\mathbf{p}_s$ is chosen randomly as a candidate for mutation. Its neighbors $\mathbf{p}_{s-1}$ and $\mathbf{p}_{s+1}$ then determine the lower and upper border of the range in which the $\mathbf{p}_s$ is changed at random.

The *crossover* operator chooses two parameterizations $\mathbf{P} = (\mathbf{p}_1, \ldots, \mathbf{p}_k)$ and $\mathbf{Q} = (\mathbf{q}_1, \ldots, \mathbf{q}_k)$ in $\mathcal{P}_i$ and generates a new offspring $\mathbf{P'} \in \mathcal{P}_{i+1}$. A position s is found at random such that the parameterization $\mathbf{P'} = (\mathbf{p}_1, \ldots, \mathbf{p}_s, \mathbf{q}_{s+1}, \ldots, \mathbf{q}_k)$ still satisfies the condition: $\mathbf{p}_s \prec \mathbf{q}_{s+1}$.

5 Experiments

In the following we describe our experiments testing the performance of canonical and standard learning algorithms of RBF networks on two problems: the XOR problem and the approximation of $\sin(x) \cdot \sin(y)$ function. The experiments were made on the PC with 400Mhz Pentium and 128MB RAM running Linux. The software called GARB used in these tests is written in C++ by the author and is publicly available [9].

The first task was a XOR problem defined by four training examples. We used 50 networks in the population and elitist selection for the two best networks. Error values for the first 500 iterations are plotted on Figure 3. Both algorithms were able to successfully learn the given task quite fast; the canonical algorithm was about two times faster in terms of error decrease. Running times of both algorithms were roughly identical—about 2 seconds per 1000 iterations.

The second experiment was an approximation of the function $f(x, y) = \sin(x) \cdot \sin(y)$ given by a 10×10 mesh of points regularly taken from a $[0; 2\pi] \times [0; 2\pi]$ square. Again, both algorithms with 50 networks in population were used with the same elitist rate as in the previous experiment. The learning speed is shown on Figure 4. The performance was similar to the previous experiment: the canonical GA was again about twice faster. The average speed of 1000 iterations was 30 seconds.

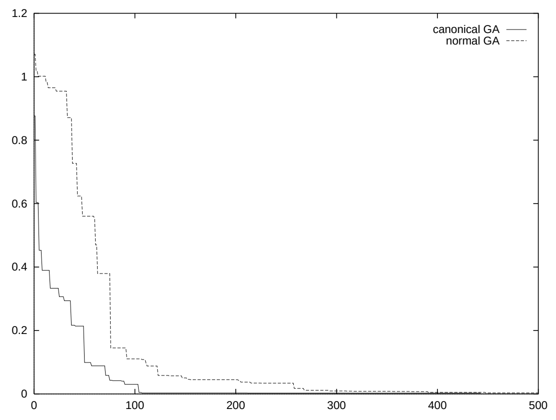

Figure 3: Comparison of error decrease for the XOR experiment.

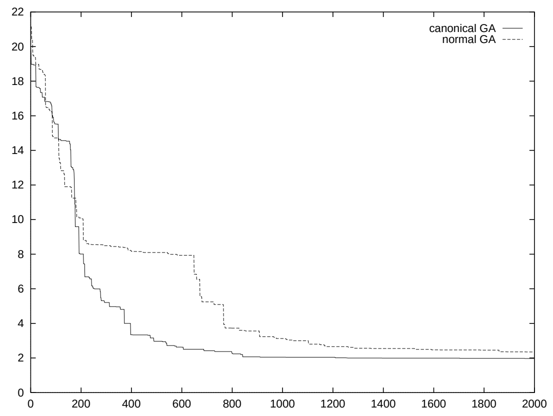

Figure 4: Comparison of error decrease for the $\sin(x) \cdot \sin(y)$ experiment.

6 Conclusions

We have presented results concerning functional equivalence property for the case of Gaussian RBF networks and one hidden layer perceptron networks. Based on these we have proposed the canonical genetic algorithm for learning feed-forward neural networks. The proposed algorithm has been realized and tested for the case of RBF networks. It has been shown that for small/middle-size tasks the canonical GA is about twice faster in reaching the same error threshold. Moreover, the canonical GA does not show any relevant increase in time for one iteration in comparison to standard GA. Thus, the twice better times hold also in real time. An interesting comparison would be against the standard learning algorithms of feed-forward networks, such as back propagation. This is one of the directions of our further work.

The results presented as theorems 3 and 4 assure that our approach is valid in the case of approximation and can be used in practical problems working with inexact values and limited precission machine arithmetics. Moreover, we hope that these results can be further extended to derive estimates of approximation error.

References

[1] F. Albertini and E.D. Sontag. For neural networks, function determines form. *Neural Networks*, 6:975–990, 1993.

[2] R. Engelking. *General Topology*. Polish Scientific Publishers, Warsaw, 1977.

[3] R. Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pages 129–135. Elsevier, 1990.

[4] K. Hlaváčková and R. Neruda. Radial basis function networks. *Neural Network World*, 3(1):93–101, 1993.

[5] V. Kůrková and P. Kainen. Functionally equivalent feedforward networks. *Neural Computation*, 6:543–558, 1993.

[6] V. Kůrková and R. Neruda. Uniqueness of the functional representations for the Gaussian basis functions. In *Proceedings of the ICANN'94*, pages 474–477. Springer Verlag: London, 1994.

[7] M. Leshno, V. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a non-polynomial activation function can approximate any function. *Neural Networks*, 6:861–867, 1993.

[8] R. Neruda. Functional equivalence and genetic learning of RBF networks. In *Proceedings of the ICANNGA'95*, pages 53–56. Springer Verlag: Vienna, 1995.

[9] R. Neruda. Genetic learning of RBF networks with GARB. Technical Report V–713, Institute of Computer Science, Prague, 1997.

[10] H.J. Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural Networks*, 5(4):589–594, 1992.