Neural Networks that Teach Themselves through Genetic Discovery of Novel Examples

Byoung-Tak ZHANG & Gerd VEENKER

Division I – Artificial Intelligence
Institute for Computer Science
University of Bonn, 5300 Bonn, Germany
Email: zhang,veenker@dbninf5.bitnet

Abstract. Conventional learning paradigms for neural networks are passive in the sense that they only try to learn from the training patterns presented by their environment or teacher. In this *passive learning paradigm* there is no learning when the environment generates no training examples. To be more useful, a learning system can learn by actively interacting with its enviroment.

In this paper we introduce an *active learning paradigm* for neural networks. In contrast to the passive paradigm, the learning in the active paradigm is initiated by the machine learner instead of its environment or teacher. We present a learning algorithm that uses a genetic algorithm for creating novel examples to teach multilayer feedforward networks. The creative learning networks, based on their own knowledge, discover new examples, criticize and select useful ones, train themselves, and thereby extend their existing knowledge. The experiments on function extrapolation show that the self-teaching neural networks not only reduce the teaching efforts of the human, but the genetically created examples also contribute robustly to the improvement of generalization performance and the interpretation of the connectionist knowledge.

1 Introduction

One of the most outstanding features of neural networks is their ability to learn from examples. There are currently three main paradigms of learning in neural networks: supervised, semisupervised and unsupervised learning. (see [2] and references therein). In *supervised learning* the training example ψ_p consists of an input pattern X_p, from the input space X and the corresponding output pattern Y_p in the output space Y of a problem domain. The goal of supervised learning is to approximate the mapping $f : X \rightarrow Y$ from the given training set $\Psi = \{\psi_p \mid p = 1, 2, ..., P\}$ as precisely as possible. In *semisupervised leraning*, the teacher presents input patterns X_p but does not give the corresponding correct outputs. Instead the teacher observes first the computed output pattern Y_p' of the learner and then evaluates the output by giving a scalar-valued *reinforcement* signal r_p (therefore this paradigm is also called reinforcement learning). In *unsupervised learning*, the teacher also gives input stimulus patterns X_p but no information ϕ about the output pattern. The task of the unsupervised learning systems is to discover and generate the regularities in the training set of input patterns. Although these three learning paradigms receive different forms of information about the ouput patterns, they all learn only when the input (or stimulus) patterns are presented from their teacher or environment. In this sense they are *passive learning paradigms* (see Figure 1).

To be more useful, a learning system should be able to actively interact with its environment to acquire new knowledge by generating hypotheses based on its own knowledge and testing them on its environment. Such research has been done in symbolic machine learning under the framework of artificial intelligence [4], but is still missing in neural networks.

CH 3065-0/91/0000-0690 $1.00 © IEEE

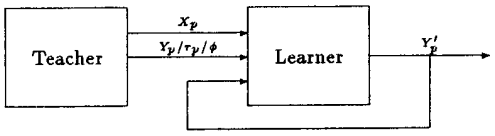

Figure 1: The passive learning paradigm

The aim of this paper is to intoroduce a new *active learning paradigm* and present a learning procedure by which neural networks adapt themselves actively to the environment by self-generating training examples during learning. The basic idea is to use genetic algorithms [1,3] for generating new examples. Section 2 introduces the active learning paradigm. Section 3 describes the genetic algorithm for discovering novel examples. Section 4 presents the learning algorithm for self-teaching neural networks. Section 5 shows the experimental results on extrapolating binary-vector functions. Finally in Section 6, some possible extentions to the current implementation are discussed.

2 Active Learning Paradigm

Figure 2 shows a schematic diagram of the active learning paradigm for neural networks. In correspondence with the three types of learning in the passive paradigm, we can divide three learning types in the active paradigm. We call them supervised, semisupervised and unsupervised creative learning, respectively. For all types of the active learning, a set of training examples, called *seed examples*, are given initially by the teacher in order to allow the learning system to set up its knowledge base. After setting up the initial knowledge, the learning system learns by generating novel examples and interacting with its environment.

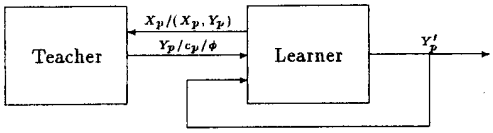

Figure 2: The active learning paradigm

In *supervised creative learning*, the learner generates only the input part of the training example, i.e. X_p. The input pattern is presented to the environment or teacher who is to answer with the correct output pattern Y_p that satisfies $Y_p = f(X_p)$ for the mapping f in the problem domain. From the newly acquired training example (X_p, Y_p), the learner refines his knowledge base through internal supervised lerning. In *semisupervised creative learning*, the learner generates a complete training example, i.e. a pair (X_p, Y_p) of input-output patterns. The task of the teacher in this type of learning is to evaluate the novel example by giving a scalar value c_p, what we call the *certainty* of the example. The certainty c_p is distinguished from the conventional reinforcement r_p in that c_p is related to the learner-generated example (input-output pair) while r_p is only to the learner-generated output pattern. In the third type, *unsupervised creative learning*, the learner generates (X_p, Y_p) as in the semisupervised creative learning, but there is no interaction between the learner and the environment during learning. The learning in this type is a completely autonomous process. In this paper we will focus on the supervised creative learning of associations between binary vector spaces with multilayer feedforward networks.

3 Creating Novel Examples by Genetic Experimentation

New examples are generated by genetic recombination of two *parent examples* of the already existing training set, Ψ. The parent examples are chosen on the basis of the reproductivity v_p of the examples (i.e. the capability of examples to mate and generate *child examples*). v_p is defined as

$$v_p = \frac{V_{Y_p}}{P_{Y_p}} = \frac{P - P_{Y_p}}{P} \cdot \frac{1}{P_{Y_p}} = \frac{P - P_{Y_p}}{P \cdot P_{Y_p}} = \frac{1}{P_{Y_p}} - \frac{1}{P} \tag{1}$$

where $P = |\Psi|$ and P_{Y_p} is the number of examples in Ψ of the category for the example p, and $V_{Y_p} = \frac{P - P_{Y_p}}{P} = 1 - \frac{P_{Y_p}}{P}$ is the reproductivity of the category Y_p. By this definition, the examples of categories which have fewer training examples have a larger probability of mating. So the novel examples of classes containing fewer examples will be more often generated than those of classes containing more.

For the creation of novel examples, three basic genetic operators and their combinations are used. The first operator is a two-point crossover operator $\otimes$. Given two string patterns $X_p = (x_1^p, ..., x_n^p)$ and $X_q = (x_1^q, ..., x_n^q)$, and two crossover points $a, b = 1, ..., n (a \leq b)$, the operator $\otimes$ generates two child patterns $X_{p\prime}$ and $X_{q\prime}$ by

$$\otimes(X_p, X_q) = \begin{cases} X_{p\prime} = (x_1^p, ..., x_{a-1}^p, x_a^q, ..., x_b^q, x_{b+1}^p, ..., x_n^p) \text{ and} \\ X_{q\prime} = (x_1^q, ..., x_{a-1}^q, x_a^p, ..., x_b^p, x_{b+1}^q, ..., x_n^q). \end{cases}$$

The second is the one-point crossover operator $\oslash$, which is defined as

$$\oslash(X_p, X_q) = \begin{cases} X_{p\prime} = (x_1^p, ..., x_{a-1}^p, x_a^q, x_{a+1}^q, ..., x_n^q) \text{ and} \\ X_{q\prime} = (x_1^q, ..., x_{a-1}^q, x_a^p, x_{a+1}^p, ..., x_n^p). \end{cases}$$

The third operator is the mutation operator $\odot$:

$$\odot(X_p) = \odot(x_1^p, ..., x_n^p) = (\odot x_1^p, ..., \odot x_n^p) = (x_1^{p\prime}, ..., x_n^{p\prime}) = X_{p\prime}$$

where the bit x_i^p is inverted by $\odot x_i^p$, subject to the mutation probability μ.

4 Learning Algorithm for Self-teaching Neural Networks

The creative learning paradigm can be easily implemented by incorporating the example generation component in the selective incremental learning algorithm that we developed earlier [7]. The creative learning network distinguishes three kinds of representation structures: an *exempel base* Φ, an *exemplar base* Ψ, and a *knowledge base* Ω. At the start of learning, $\Phi = \phi$ and Ψ contains a small number of *seed examples* given by the teacher. $\Omega = \{w_{ji}\}$ has randomly initialized connection weights of the fully-connected feedforward networks with one hidden layer (n-h-m networks). There are three learning mechanisms in the system: creation, selection and adaption (see the box below for a summary of the algorithm).

The adaption algorithm trains the network using only the examples in Ψ and updates Ω. We use an online backpropagation algorithm [5] with the following weight modification rule

$$\Delta w_{ji}(t) = \epsilon(1 + e_p)\frac{\partial E_p}{\partial w_{ji}(t-1)} + \eta \Delta w_{ji}(t-1) \tag{2}$$

where ϵ and η are the adjustment rate and momentum factor in the standard backpropagation, respectively. e_p is called the *interestingness* of the example p and defined as Equation 3 below. (This

interestingness term gives a second momentum to the interesting examples for quicker adaption to the existing knowledge structure Ω [6].)

After training of the network to the required precision, the creator generates new examples through genetic experimentation on existing examples, as was described in the last section. The generation of novel patterns and the communication with the environment are dependent on the types of creative learning; in supervised creative learning the learner generates X_p and the teacher Y_p, in semisupervised and unsupervised creative learning the learner creates (X_p, Y_p) and the teacher c_p and ϕ (no information), respectively. The newly generated examples are added to Φ.

Following the creation, the selector chooses the most useful (maximal π) examples in Φ. In selecting the useful examples, the so-called *interestingness* is computed by

$$e_p = \frac{1}{m}\sqrt{(Y_p - \Gamma(\Omega, X_p))^T(Y_p - \Gamma(\Omega, X_p))} = \frac{1}{m} \parallel Y_p - \Gamma(\Omega, X_p) \parallel \tag{3}$$

where m, X_p, Y_p and $\Gamma(\Omega, X_p)$ are the number of output units, the input, the desired and the actual output vectors of the network, respectively. If an example has sufficient interestingness ($e_p > \beta$), then it is selected and moved to the exemplar base Ψ. The uninteresting examples ($e_p < \alpha$) are deleted from the exempel base Φ. Through this incremental selection of critical examples, the network achieves improved generalization using possibly reduced sizes of training sets which contributes to the better interpretation of acquired knowledge (see [7] for more details).

1. **Adaption**
   - (a) (*Termination criterion*) Select λ. $t \leftarrow 0$.
   - (b) (*Training*) $t \leftarrow t + 1$. $E \leftarrow 0$. $\forall \psi_p = (X_p, Y_p, c_p) \in \Psi$:
     - i. $a_j = 1/(1 + e^{-(\sum_i w_{ji} a_i + \theta_j)})$
     - ii. $E_p = (Y_p - \Gamma(\Omega, X_p))^T(Y_p - \Gamma(\Omega, X_p))$, $e_p = \frac{1}{m}\sqrt{E_p}$, $E \leftarrow E + E_p$.
     - iii. $w_{ji}(t) = w_{ji}(t-1) + c_p \Delta w_{ji}(t)$ (see Eqn. 2).
   - (c) (*Termination test*) If $E < \lambda$, terminate *adaption*. Otherwise go to *training*.
2. **Creation**
   - (a) (*Reproduction*) $\forall \psi_p \in \Psi : v_p = \frac{1}{P_{Y_p}} - \frac{1}{P}$. Choose ψ_p and ψ_q with $v_p, v_q \geq v_r, r \neq p, q$.
   - (b) (*Recombination*) Generate $\varphi_{p'} = (X_{p'}, Y_{p'}, c_{p'})$ and $\varphi_{q'} = (X_{q'}, Y_{q'}, c_{q'})$ using $\otimes$, $\oslash$, $\odot$ and the environment (see text).
   - (c) (*Expansion*) $\Phi \leftarrow \Phi \cup \{\varphi_{p'}, \varphi_{q'}\}$.
3. **Selection**
   - (a) (*Criticism*) $\forall \varphi_p = (X_p, Y_p, c_p) \in \Phi : e_p = \frac{1}{m} \parallel Y_p - \Gamma(\Omega, X_p) \parallel$
   - (b) (*Survival*) $\forall \varphi_p$ with $e_p \geq \beta$: $\Psi \leftarrow \Psi \cup \{\varphi_p\}$, $\Phi \leftarrow \Phi - \{\varphi_p\}$.
   - (c) (*Expulsion*) $\forall \varphi_p$ with $e_p < \alpha$: $\Phi \leftarrow \Phi - \{\varphi_p\}$. Go to *adaption*.

Then the extended Ψ is used by the adaption algorithm to train the network, and the following cycles of the creation-selection-adaption steps are repeated until the network generalizes perfectly into the test set Ξ. At the termination of learning, Ψ contains the examples for training the neural networks, and Ω contains the connectionist knowledge.

5 Extrapolating Functions Starting with Two Seed Examples

The self-teaching capability of the creative learning networks was tested on extrapolation of binaray-vector functions. Figures 3 and 4 show the results of two extreme cases of problems of input size $n = 15$: the majority function (2^1 outputs; return a 1 if more than half of the input units are 1) and the autoassociation problem (2^n output vectors; the output vector the same as the input vector). Each figure shows the generalization performance of the networks in the course of creation-selection-adaption cycles (g), using the supervised creative learning scheme. In each creation-selection cycle, 20 examples were generated and 10 of them were selected. The mutation rate was $\mu = 0.3$. To compare the perfomances we have also shown the result of a random generation method (no reproduction, no crossover, no selection and mutation with $\mu = 0.5$). The graphs are based on the average values of fifty runs. The networks and seed examples used in the experiments are shown in Table 1. The parameter values for training the network were $\epsilon = 0.1$, $\eta = 0.9$.

Task	Network	Seed examples
Majority function	15-1-1	(11...11 → 1) and (00...00 → 0)
Autoassociation	15-15-15	(11...11 → 11...11) and (00...00 → 00...00)

Table 1: The networks and seed examples used in the experiments

In the figures we can see that the generalization performances G of self-teaching networks are better than those (G_{random}) of the networks trained with randomly generated examples. This confirms that the reproduction, crossover and selection components of the algorithm are useful to guide the search for discovering critical training examples. The results can be summarized as follows. First, the human teacher did not have to make efforts to prepare the training examples. He only needed to give two seed examples and correct output patterns for the machine-generated input patterns. For the case of 15-input majority problem, the teacher is required only to generate a 1-bit output pattern, Y_p, instead of a 16-bit input-output-pattern (X_p, Y_p). Second, for 100% generalization only a small number of the possible training examples were actually used, while the same number of random examples could not always achieve 100% generalization. This is the result of the genetic creation and focused selction capabilities of the learning algorithm. Third, the reduced training sets were representative in that they contained many *critical patterns* [7]. This helps to interpret the knowledge hidden in the connectionist network.

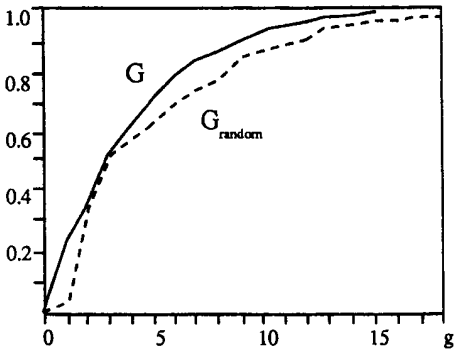

Figure 3: 15-1-1 Majority network

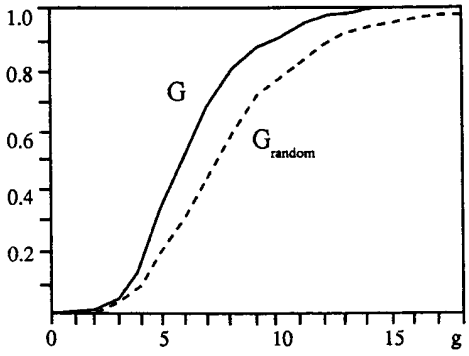

Figure 4: 15-15-15 Autoassociation net

6 Concluding Remarks

We introduced a new learning paradigm in which the learner, by discovering new examples, actively interacts with its environment or teacher, instead of passively adapting to the given stimulus patterns of its environment. A method for creating novel examples using genetic recombination of existing training examples are presented. The creative learning paradigm was implemented in the framework of selective incremental learning neural networks [7]. The creative learning process takes place incrementally by generating novel examples through genetic experimentation on the existing examples, criticizing and selecting useful ones, and adapting the network to the new examples.

The preliminary results on function extrapolation tasks show that creative self-teaching neural systems have advantages in the generalization performance, in the interpretation of the connectionist knowledge, as well as in the reduction of human teaching efforts. Since the training examples for neural computers correspond to the programs to be compiled for execution in conventional computers, and new examples in creative learning are created automatically by the network itself, this work can be seen as a step to the construction of self-programming and so more "intelligent" neural computers.

To make the method presented here more powerful, two points should be extended in the future. First, the incremental growth of the training set requires the network itself to grow. Without this ability the network must be initialized to an arbitraly large size, which entails in general bad generalization and so requires a large training set for the specified performance. Second, it would be useful for the creation component to have a more diverse set of genetic (or heuristic) operators whose use can also be adapted to the given problem structure during the incremental creation of examples.

Acknowledgements

We would like to thank Knut Möller, Jörg Kindermann, Heinz Mühlenbein, Michael J. Krueger and all the other members of the neural network research groups of the University of Bonn and GMD for useful discussions on the early version of this paper. This work was supported in part by the POSCO Scholarship Society.

References

[1] Goldberg DE (1989) *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley.

[2] Hinton GE (1989) Connectionist learning procedures, *Artificial Intelligence 40*, pp. 185-234.

[3] Holland JH, Holyoak KJ, Nisbett RE & Thagard PR (1986) *Induction: Processes of Inference, Learning, and Discovery*, MIT Press.

[4] Michalski RS, Carbonell JG & Mitchell TM (eds.) (1983,1986) *Machine Learning: An Artificial Intelligence Approach*, Volume I (Tioga, 1983) and Volume II (Morgan Kaufmann, 1986).

[5] Rumelhart DE, Hinton GE & Williams RJ (1986) Learning internal representations by error propagation, in: Rumelhart DE & McClelland JL (eds.) *Parallel Distributed Processing*, Vol. I, pp. 318-362, MIT Press.

[6] Zhang BT (1990) *GENIAL: A Genetic Neural Evolutionary Model for Autonomous Learning and Self-Development*, Inst. for Comp. Sci., Div.I – AI, University of Bonn (in German).

[7] Zhang BT & Veenker G (1991) Focused incremental learning for improved generalization and reduced training sets, in: *Proc. of Int'l Conf. on Artificial Neural Networks*, ICANN-91, North-Holland Press.