Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, February 15-17, 2006 (pp254-258)

Towards Multi-Layer Perceptron as an Evaluator Through Randomly Generated Training Patterns

JANIS ZUTERS
Department of Computer Science
University of Latvia
Raina bulv. 19, Riga, LV-1050
LATVIA
janis.zuters@lu.lv

Abstract: - Multi-layer perceptron (MLP) is widely used, because many problems can be reduced to approximation of functions. Pattern evaluation, which is discussed in this article, belongs to this range of problems. MLP manages function approximation problems quit well, however an important prerequisite is a uniformly distributed set of training patterns. Unfortunately, such a set is not always available. In this article, the use of randomly generated additional training patterns is examined to see whether this improves the training result in cases, when just "positive" patterns are available.

Key-Words: - Learning process, Multi-layer perceptron, Randomly generated patterns, Pattern evaluation

1 Introduction

The problem of pattern evaluation can be simply solved with a multi-layer perceptron. We just have to design a network that approximates the unknown evaluation function *f(·)* [1]:

$$d = f(x), \tag{1}$$

where the vector x is the input and the vector d is the output.

We are given the set of labeled examples (training patterns) T:

$$\mathrm{T} = \{(x_i, d_i)\}_{i=1}^{n} \tag{2}$$

The function *F(·)* describing the input-output mapping actually realized by the network should be close enough to *f(·)* in a Euclidean sense over all inputs:

$$\forall x \left\| F(x) - f(x) \right\| < \varepsilon, \tag{3}$$

where ε is a small positive number.

Such problems are perfect candidates for supervised learning, int. al. MLPs.

A problem arises if we have no set of patterns adequately representing *f(·)*.

[2] describes such a case with solving a timetabling problem via a genetic algorithm (GA), where a neural network is intended to be a part of the fitness function within GA in order to support the evaluation (rating) of solutions (timetables). The idea is to train the network on existing (i.e., valid) school-timetables, thus obtaining the network, which is able to evaluate timetables. Unfortunately for this approach we have no "bad" or "not very good" examples of timetables, so the available training set is incomplete, and the evaluation should be done as some kind of similarity computation between the candidate timetable and valid timetables. Figure 1 shows the schema of GA-based timetabling, where the timetable evaluation should be partially committed to a neural network.

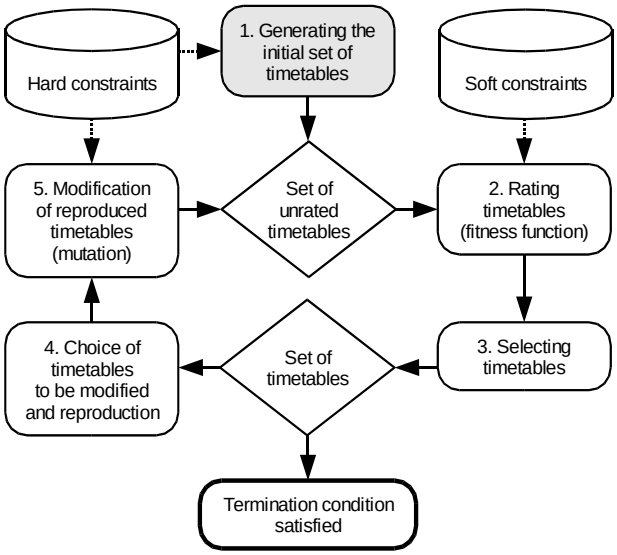

Fig. 1. The schema for GA in timetabling [2]

In this article the author examines a method of organizing a learning process (LP) through adding randomly generated training patterns (TPs) to the training set. The method is proposed in order to compensate for the lack of a complete training set.

2 Learning Process of a Neural Network

The ability of the network to learn from its environment and to improve its performance through learning is the property of primary significance for neural networks. [1]

The two aspects of the learning process are to be distinguished:

1. Learning algorithm (LA). LAs differ significantly among various neural network models.
2. Organization of the LP.

The latter encompasses the way in which TPs are passed to the network and correspondingly, to the LA. Usually the LP is arranged in epochs. An epoch is one sweep through all the patterns in the training set, presenting them to the network, i.e. the LA.

Provided that the set of training patterns T is available (see (2)), the LP can be generally described as follows (Fig. 2):

```
% T - set of training patterns
do
    p := fetch the first pattern from T
    do
        operate the learning algorithm on p
        p := fetch the next pattern from T
    while p is available
while the stopping criterion not met
```

Fig. 2. A learning process

Fetching of patterns can differ in terms of sequence – some LAs require random order, others do not.

3 Organization of the Learning Process with Incomplete Available Training Set

3.1 The Problem

Let's explore the following problem: Compute the extent of similarity between a given alphanumeric symbol and a fixed set of alphanumeric symbols T.

In order to solve this problem, we need to design a similarity function for the fixed set T:

$$h^{\mathrm{T}} : \mathrm{A} \to \{0..1\} \qquad (4)$$

where T⊂A is a set of alphanumeric symbols.

This problem is the same as the one discussed in Section 1, just replacing patterns of symbols with timetables.

To solve this problem we could face the 2 following sub-problems.

Sub-problem 1 (extremely important with timetable evaluation within GA). Determine the criterion of similarity between 2 patterns (e.g., how close are symbols 'B' and '8').

Sub-problem 2 (assuming that the problem is being solved using a neural network trained just on positive patterns). Overcome the lack of a complete training set.

The author proposes adding randomly generated patterns to the training set to try to solve such problems.

3.2 Unsuccessful Attempts

The first idea was to examine the Kohonen network. The competition principle based on comparison between neurons seemed very promising – just to replace "winner takes it all" to "everyone takes as much as deserved according to gained evaluation". Unfortunately various attempts crashed – the Kohonen network did the clusterization well, still it was unable to find out the evaluation of assigning a pattern to some cluster.

3.3 The Proposed Method – Training MLPs with Randomly Generated Patterns

The second idea, which yielded results, was to use additional training patterns along with available ones. By this approach we have to choose the random rate $\tau \in \{0..1\}$, which denotes the proportion of the use of randomly generated patterns. The proposed supervised learning method can be described as follows (see Figure 3):

```
% T - set of positive training patterns
% τ - the random rate {0..1}
% s - size of the training set
% determining the size of an epoch (s2 ≥ s):
s2 := s / (1 - τ)
do
    for i := 1 to s2 do
        rnd := get random value from interval 0..1
        if rnd < τ then
            p := generate random pattern
            d := 0
        else
            p := choose a pattern from T in random
            d := 1
        operate the learning algorithm on [p, d]
while the stopping criterion not met
```

Fig. 3. A supervised learning process with additional randomly generated training patterns

The proposed method was tested through the experimentation described below.

4 Description of the Experimentation

The goal of the experimentation was to examine a method proposed by the author for organizing the LP by adding randomly generated TPs (solving problems like the problem, described in Section 3.1). The goal of each experiment is to obtain a neural network that could be used for evaluation of patterns, i.e., one, which realizes the similarity function (4).

4.1 Describing the Learning Process of the Experimentation

The classic MLP with the backpropagation learning algorithm was used in the experimentation along with the author's proposed learning method (See Fig. 3).

The core MLP operating and training algorithm is briefly shown as follows [3].

Propagation function:

$$NET_j = \sum_{i=0}^{n} w_{ji} o_i \,, \qquad (5)$$

where NET_j is propagation value of the neuron j; w_{ij} – the i^{th} weight of the neuron j (w_{j0} – bias of the neuron j); o_i – the i^{th} input signal of the neuron j (o_0 = 1).

Activation function:

$$o_j = \varphi(NET_j) \,, \qquad (6)$$

where o_j – output value of the neuron j; $\varphi(\cdot)$ – logistic activation function.

The general weight correction rule:

$$\Delta w_{ji} = \eta \delta_j o_i \,, \qquad (7)$$

where Δw_{ji} – correction of the i^{th} weight of the neuron j; η – learning rate; δ_j – local gradient of the neuron j (not specified in this paper); o_i – the i^{th} input signal of the neuron j.

4.2 Architecture of Neural Networks Used in Experiments

In computer experiments neural networks of the classic MLP architecture were used:

- Size of the input signal – 1280.
- 1 hidden layer with 3, 5, or 7 neurons.
- 1 neuron in the output layer.
- Networks operate as described above: see (5), (6), (7).

Logistic function was used as an activation function with the gain $\gamma \in \left\{\frac{m}{10}, \frac{m}{15}, \frac{m}{20}\right\}$, where m – number of neuron inputs.

4.3 Available Set of Training and Testing Patterns

The networks were tested on black and white images of a size 32 × 40 pixels with black hand-written digits and capital Latin letters depicted on a white background (Fig. 4). The number of pattern (symbol) types was n=36 (10 digits + 26 letters). Let's denote the set of pattern types as Γ^0**.** The number of variants for each pattern type was v=4. So, the total size of the set of available patterns T^0 were n×v = 144. As we have 4 different variants of each symbol (i.e., four rows of patterns), then for each experiment, one of the rows served as the test set, the other three – as the training set.

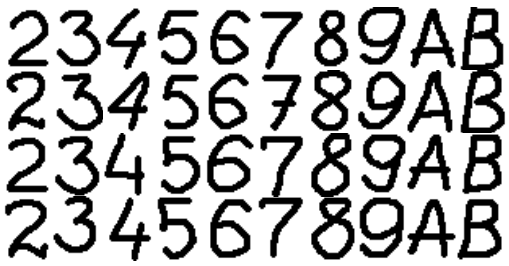

Fig. 4. A subset of the set of available patterns T^0

4.4 Course of a single experiment

A total of 1,250 experiments were conducted. Each experiment involves the designation, training, and testing one MLP.

1. Create a new MLP according to the description in Section 4.2.
2. Choose from T^0, at random, three rows of available patterns (Fig. 4) as a candidate set of training patterns $T^{0A} \subset T^0$, so those of the remaining row would be a test set $T^{0B} \subset T^0$.
3. Choose at random the "random rate" τ among values {0, 0.2, 0.5, 0.7, 0.9}. τ=0 means that the network is trained just on selected patterns without use of randomly generated ones.
4. Choose at random the amount of pattern types c among values {1, 2, 4} in the training set.
5. Select pattern types at random for the training set from 36 available digits and letters: $\Gamma = \{q_1, q_2, ..., q_c\} \subset \Gamma^0$. As three rows of four available figures serve for training, the chosen pattern types represent s = c × 3 patterns, so the training set

would consist of s patterns: $T=\{p_1, p_2, ..., p_s\}\subset T^{0A}$.

6. Train the network using the author's proposed method (Section. 3.3) until 100 epochs are passed or the total error of output neurons decreases under $\varepsilon=0.1$.
7. Random patterns were generated as black and white images of a size of 32 × 40 pixels, with proportion of black at 0.17-0.35 (also chosen at random).
8. Test the trained network on the set of test patterns T^{0B} and record the outputs, thus forming the experimental results.

The goal of an experiment – to obtain a network that is able to evaluate input patterns T^{0B} (represents Γ^0) with respect to a fixed set Γ, i.e. realizes a version of the similarity function h_Γ.

5 Analysis of the Proposed Training Method

Sub-problem 2 (Section 3.1) is stated to determine the criterion of similarity between two patterns. In this case (analyzing the training method), such a criterion should be obtained by external means, and the similarity function that is based on it would then serve as a benchmark to measure the quality of the solution.

5.1 A Questionnaire Method to Obtain the Similarity Criterion

As there exist no manifest formal criteria in terms of similarity between two patterns, human opinion was chosen as a criterion. A survey was arranged, and eight respondents were asked to evaluate pairs of symbols (a total of 630 pairs – each of 36 symbols with each of the rest) with the mark between 0 and 1 (with a minimum step of 0.1), where 1 means – "very similar", but 0 – "absolutely different" (Fig. 5). All the respondents rated most of pairs as 0.

As the average result of all respondents, the **similarity measure** for two patterns $g(\cdot,\cdot)$ was obtained:

$$g:\Gamma^0\times\Gamma^0\to\{0..1\} \tag{8}$$

To reduce the computation, the function $g(\cdot,\cdot)$ was simplified in a manner so that the following 2 equations hold true:

$$\forall i\in\Gamma^0 : g(i,i)=1 \tag{9}$$

$$\forall i,j\in\Gamma^0 : g(i,j)=g(j,i) \tag{10}$$

Code	Pat1	Eval.	Pat2	/	Code	Pat1	Eval.	Pat2
2920	T		K	/	2901	T		1
2902	T		2	/	2912	T		C
2913	T		D	/	2923	T		N
2924	T		O	/	2905	T		5
2906	T		6	/	2916	T		G
2917	T		H	/	2927	T		R
2928	T		S	/	2909	T		9

Fig. 5. An excerpt of an inquiry form used in the survey

Now we can define the **similarity function** with respect to Γ:

$$h_\Gamma(i)=\max_{j\in\Gamma} g(j,i), \tag{11}$$

where $\Gamma\subset\Gamma^0$ – the set of pattern types, representing the training set, and $i\in\Gamma^0$ – the pattern type. The introduced similarity function represents the average evaluation of all respondents with respect to Γ.

5.2 Representing the Similarity by the Similarity Sequence

Unfortunately, it was impossible to examine experimental results through comparing them directly to the values yielded by the similarity function. That was because of disparate absolute output values among experiments. There was a need for a different notion of the similarity function.

The idea is, instead of the similarity function (11), to represent the similarity by the **similarity sequence** with respect to Γ:

$$\chi=\langle\chi_i\rangle_{i=1}^n, \tag{12}$$

where $i\in\Gamma^0$ determines the type of the pattern; χ_i determines the position of the pattern type in the sequence, ordered according to the similarity function:

$$h(i)\le h(j)\to\chi_i\le\chi_j \tag{13}$$

5.3 The Proposed Error Function to Evaluate The Training Method

Assume that the result of an experiment is also expressed as an ordered sequence of pattern types:

$$\psi=\langle\psi_i\rangle_{i=1}^n, \tag{14}$$

ordered according to the outputs of the network:

$$F(i)\le F(j)\to\psi_i\le\psi_j \tag{15}$$

where $F(\cdot)$ – the function realized by the neural network trained on Γ.

Then we can define the **sequential error** $\lambda(\cdot,\cdot)$ as difference between positions of two sequences:

$$\lambda(\chi,\psi)=\sqrt{\frac{\sum_{i=1}^{n}(\chi_i-\psi_i)^2}{n}} \qquad (16)$$

Using the similarity sequence (12) instead of the similarity function (11) is acceptable with evaluation (fitness) function within a GA, as the fitness function is involved in determining, which solutions are to be eliminated or to be chosen as parents, and the absolute values of the evaluation are not of great importance.

In the next section, the performed experiments are examined according to the sequential error λ (16).

5.4 Analysis of Experimental Results and Conclusion

All the experiments were examined according to the sequential error λ and grouped by the number of pattern types c in the training set and random rate τ (see Section 4.4). The summary of experimental results is shown in Table 1.

Table 1. Experimental results as average of the sequential error λ, grouped by the number of pattern types c and the random rate τ.

sequential error (λ)	random rate (τ)	number of pattern types in T (c)
10.51049	0.5	2
10.52552	0.9	4
10.73023	0.5	4
10.73762	0.7	4
10.76883	0.2	4
10.88012	0.5	1
10.90313	0.2	2
10.96310	0.7	2
10.99388	0.7	1
11.11132	0.2	1
11.23665	0.9	1
11.26268	0.9	2
11.41785	0.0	1
11.62686	0.0	2
12.28688	0.0	4

According to the results shown in Table 1, the value of λ is nearly 11, i.e., it is a great distance away from the "ideal" value of 0. It is just a little better than one, which can be acquired by randomly generated sequences, which yield the λ value of approximately 14.

To better evaluate the experimental results, the similarity sequences, obtained from separate respondents, and the total similarity sequence (12) also were examined. This was done by comparing them in terms of sequential error λ, and the results were between 2.82 and 5.99, i.e., also rather far away from the "ideal" value. Against that background, the acquired results for the proposed training method look fairly good.

Although the improvement is small, we still observe the following benefits:

- There is a noticeable effect of using just "positive" patterns in the training set – the improvement is small, but stable.
- The worst results were shown by random rate of value 0. This shows the effect of using randomly generated patterns in the learning process.

The results encourage the author to continue research in order to build a neural network that would serve as evaluator of school timetables, one, which would be included in a genetic algorithm as a part of the fitness function.

References:

[1] S. Haykin, Neural networks: a comprehensive foundation, 2nd ed. Prentice-Hall, Inc, 1999.

[2] J. Zuters, An Adaptable Computational Model for Scheduling Training Sessions, *Annual Proceedings of Vidzeme University College "ICTE in Regional Development"*, 2005, pp. 110-113.

[3] J. Zuters, An Extension of Multi-Layer Perceptron Based on Layer-Topology, *Proceedings of the 5th International Enformatika Conference'05*, 2005, pp. 178-181.