A Supervised Learning Process Using Randomly Generated Training Patterns to Obtain an Evaluative Neural Network

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

Abstract: - Supervised neural networks are widely used, because many problems can be reduced to the approximation of functions. Pattern evaluation, which is discussed in this article, belongs to this range of problems. Function approximation problems are perfect candidates for networks with supervised learning, but 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. The effect of the proposed method is shown analyzing results of experiments with multi-layer perceptrons (MLPs) and radial-basis function networks (RBF networks).

Key-Words: - Supervised learning process, 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. Multi-layer perceptrons and RBF networks.

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.

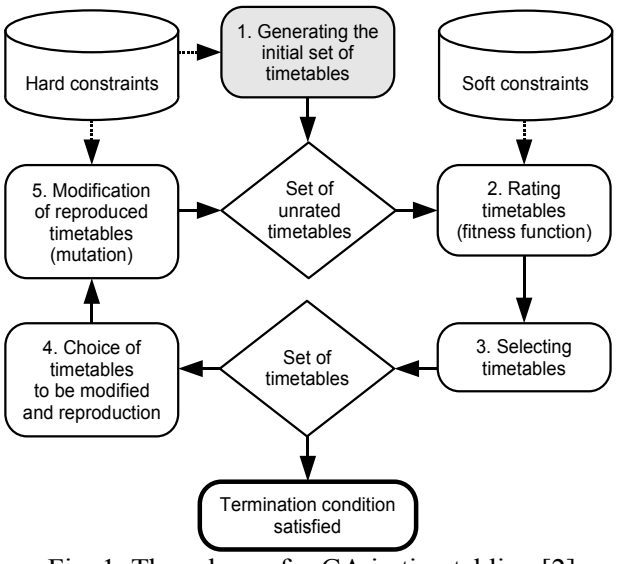

Fig. 1. The schema for GA in timetabling [2]

In this article the author examines a method of organizing a supervised 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 Problems

Let's explore the following two problems.

Problem 1. Compute the extent of similarity between a given alphanumeric symbol and some digit (the extent to which the symbol resembles some digit). In order to solve this problem, we need to design a similarity function:

$$h_{digits} : \mathrm{A} \rightarrow \{0..1\} \tag{4}$$

where A is set of alphanumeric symbols.

Examples. For every digit symbol, this function should return 1, or at least to a value close to 1. For the letter 'B' the return value should also probably be large enough, because it is by appearance similar to the digit '8'.

Problem 2. Compute the extent of similarity between a given alphanumeric symbol and a fixed set of alphanumeric symbols T.

In order to solve Problem 2, we need to design a similarity function for the fixed set T:

$$h_{\mathrm{T}} : \mathrm{A} \rightarrow \{0..1\} \tag{5}$$

where T⊂A is a set of alphanumeric symbols.

The Problem 1 is a special case of Problem 2, assuming the fixed set B is a set of digit symbols.

Problem 2 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 handled 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 Supervised Networks 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 depicted in Figure 3.

The proposed method was tested through the experimentation described below.

```
% 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

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 Problem 2, 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 (5).

4.1 Describing the Learning Process of the Experimentation

The classic MLP with the backpropagation learning algorithm and the basic form of the RBF-network were used in the experimentation along with the author's proposed learning method (See Fig. 3).

4.1.1 Operation of a MLP

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

Propagation function:

$$NET_j = \sum_{i=0}^{n} w_{ji} o_i , \quad (6)$$

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:

$$o_j = \varphi(NET_j) , \quad (7)$$

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

The general weight correction rule:

$$\Delta w_{ji} = \eta \delta_j o_i , \quad (8)$$

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.

4.1.2 Operation of an RBF-Network

The algorithm for operating and training RBF-networks used in experiments is depicted as follows [1].

Activation function for the RBF-layer:

$$o_j = \varphi_j(x) = G\left(\left\|x - t_j\right\|\right) , \quad (9)$$

where G – the Gauss function; t_j – the center of the neuron j.

A linear activation function is used for the output layer:

$$o_j = NET_j , \quad (10)$$

where NET_j – value of the propagation function, computed by (6).

The simplest training method for the RBF-layer is selected: All the available TPs are chosen as centers.

The output layer is trained according to (8) with respect to (10).

4.2 Architecture of Neural Networks Used in Experiments

4.2.1 Architecture of a MLP

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 in Section 4.1.1.

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.

4.2.2 Architecture of an RBF-network

In computer experiments, neural networks of the basic RBF-network architecture were used:

- Size of the input signal – 1280.
- RBF-layer with s neurons, where s – the number of available training patterns (3, 6, or 12).

- 1 linear neuron in the output layer.
- Networks operate as described in Section 4.1.2.

In the RBF-layer, the Gauss function was used with the slope coefficient $\sigma \in \{0.2, 0.3\}$.

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.

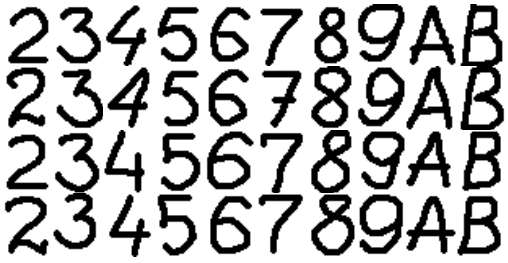

Fig. 4. A subset of the set of available patterns T^0

4.4 Course of a single experiment

A total of 1,250 experiments with MLPs and 900 experiments with RBF networks were conducted. Each experiment involves the designation, training, and testing one network.

1. Create a new neural network 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 $T^{0B} \subset T^0$ ($T^{0A} \cup T^{0B} = T^0$, $T^{0A} \cap T^{0B} = \varnothing$).
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: $T = \{p_1, p_2, ..., p_s\} \subset T^{0A}$. In case of an RBF-network, the patterns from T become the centers of the RBF layer.
6. Train the network using the author's proposed method (Section. 3.3) until the stopping criterion is met. The stopping criterion for MLPs: 100 epochs are passed, or the total error of output neurons decreases under ε=0.1. The stopping criterion for RBF-networks: 200 epochs are passed, or the maximum error of output neurons decreases under ε=0.05.
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 The Methodology for Analyzing 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). 204 pairs of symbols (of 630) were rated as more than 0 by at least one respondent.

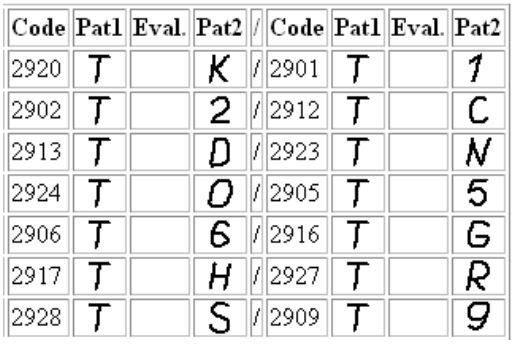

Fig. 5. An excerpt of an inquiry form used in the survey

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

$$g : \Gamma^0 \times \Gamma^0 \to \{0..1\} \tag{11}$$

Table 1. A portion of the results of the survey (pairs of patterns, the similarity of which was evaluated at least as 0.2 of 1).

Pat1	Pat2	Eval.
0	O	0.98
6	G	0.80
0	D	0.74
O	Q	0.71
0	Q	0.70
1	I	0.65
5	S	0.64
8	B	0.61
D	O	0.61
7	Z	0.60
C	G	0.54
P	R	0.50
E	F	0.48
U	V	0.46
1	7	0.44
K	X	0.44
8	S	0.39
V	Y	0.36
3	8	0.35
6	C	0.33

Pat1	Pat2	Eval.
1	J	0.31
6	9	0.31
7	T	0.31
B	R	0.31
X	Y	0.31
M	N	0.29
6	S	0.26
C	O	0.25
G	Q	0.25
1	T	0.24
B	E	0.24
I	J	0.24
N	V	0.23
9	O	0.21
D	Q	0.21
F	T	0.21
2	Z	0.20
3	B	0.20
H	M	0.20

The data in Table 1 represent the function g, defined by (11).

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

$$\forall\, i \in \Gamma^0 : g(i,i) = 1 \tag{12}$$

$$\forall\, i, j \in \Gamma^0 : g(i,j) = g(j,i) \tag{13}$$

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

$$h_\Gamma(i) = \max_{j \in \Gamma} g(j,i), \tag{14}$$

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 (14), to represent the similarity by the **similarity sequence** with respect to Γ:

$$\chi = \langle \chi_i \rangle_{i=1}^{n}, \tag{15}$$

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:

$$h(i) \le h(j) \to \chi_i \le \chi_j \tag{16}$$

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:

$$\psi = \langle \psi_i \rangle_{i=1}^{n}, \tag{17}$$

ordered according to the outputs of the network:

$$F(i) \le F(j) \to \psi_i \le \psi_j \tag{18}$$

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:

$$\lambda(\chi,\psi) = \sqrt{\frac{\sum_{i=1}^{n} (\chi_i - \psi_i)^2}{n}} \tag{19}$$

Using the similarity sequence (15) instead of the similarity function (14) 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 (19).

6 Analysis of Experimental Results

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 Tables 2 and 3, and Fig. 6. Only those experimental results of RBF-networks were included in the summary for which the training algorithm converged within less then 200 epochs.

Table 2. Experimental results on MLPs 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 Tables 2 and 3, the error λ is nearly 10-11, 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 error value of approximately 14.

The RBF-networks show better results than MLPs, and yet this doesn't guarantee the same effect for evaluating timetables.

To better recognize the background of the experimentation, the evaluations of separate respondents also were analyzed.

Assume that the evaluations of separate respondents are expressed as in (15). As most of evaluations yielded 0, the equation (19) was modified, and, to compare two evaluations, the following difference function was introduced:

$$\lambda^{+}(A,B) = \sqrt{\frac{\sum_{i=1}^{n} \begin{cases} 0; A_i = 0 \wedge B_i = 0 \\ (A_i - B_i)^2; otherwise \end{cases}}{n}}, \quad (20)$$

where A and B – evaluative similarity sequences (e.g., of two different respondents).

Table 3. Experimental results on RBF-networks 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)
9.959279	0.5	4
9.998273	0.9	4
10.30499	0.2	2
10.50103	0.2	4
10.59219	0.9	2
10.63132	0.7	4
10.74725	0.0	2
10.93429	0.5	2
10.93745	0.2	1
11.01699	0.0	4
11.02072	0.5	1
11.09412	0.7	2
11.10901	0.7	1
11.12824	0.9	1
11.14873	0.0	1

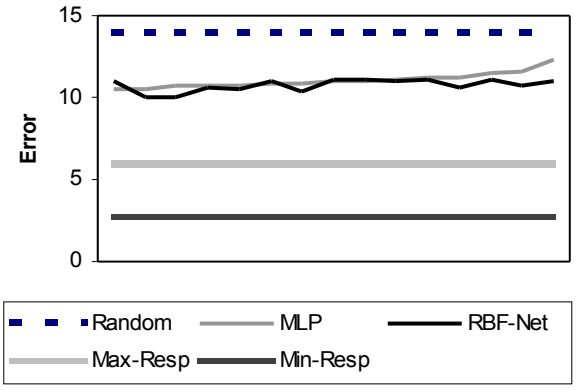

Fig. 6. Summary of experimental results for both network models, ordered by the results on MLPs

Explanation of the legend of Fig. 6:

- Random – the approximate error value of randomly generated sequences;
- MLP – the error of MLPs (Table 2);
- RBF-Net – the error of RBF-networks (Table 3);
- Max-Resp – the highest error of a respondent;
- Min-Resp – the lowest error of a respondent.

Applying the modified difference function λ^+, the evaluations, obtained from separate respondents and the average evaluation of all respondents (15) were compared, thus obtaining the average error for each of respondents. The "error" of respondents was between 2.82 and 5.99, i.e., also rather far away from the "ideal" value (see also Fig. 6).

This value shows the answers of respondents to be rather different, thus emphasizing the subjectivity of the criterion and softening the error level of the experiments. Against that background, the acquired results for the proposed training method look fairly good.

The summary of experimental results for both MLPs and RBF-networks is shown in Fig. 6.

7 Conclusion

The proposed training method yielded positive results.

The effect of using the method may vary on the type of the problem, and there's hope that the effect will be considerable enough with evaluating timetables.

Besides the limited capabilities of the method itself, there are several possible reasons of the high error level in the experiments:

- The choice of the criterion to evaluate the method: The similarity measure (11), which represents human opinion;
- The choice of the methodology to evaluate the method: The similarity function, based on the similarity measure (15);
- Specificity of the problem solved in the experiments: Evaluation of patterns of symbols.

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.