International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:7, 2007

An Extension of Multi-Layer Perceptron Based on Layer-Topology

Jānis Zuters

***Abstract*—**There are a lot of extensions made to the classic model of multi-layer perceptron (MLP). A notable amount of them has been designed to hasten the learning process without considering the quality of generalization. The paper proposes a new MLP extension based on exploiting topology of the input layer of the network. Experimental results show the extended model to improve upon generalization capability in certain cases. The new model requires additional computational resources to compare to the classic model, nevertheless the loss in efficiency isn't regarded to be significant.

***Keywords*—**Learning algorithm, multi-layer perceptron, topology.

I. Introduction

GENERALIZATION is one of the main capabilities of artificial neural networks. Generalization refers to the neural network producing reasonable outputs for inputs not encountered during training [1].

A network is said to generalize well when the input-output mapping computed by the network is correct (or nearly so) for test data never used in creating or training the network. Generalization is influenced by three factors: (i) the size of the training set, and how representative it is of the environment of interest, (ii) the architecture of the neural network, and (iii) the physical complexity of the problem at hand [1].

Let's focus on the second factor starting up by the following example (Fig. 1).

(a) (b) (c) (d)

Fig. 1. Four sample patterns to demonstrate human ability to differentiate

Here four bitmaps of similar proportions of black and white are depicted. It's obvious for first two bitmaps to be differentiated by humans more easily than second two. Simply speaking, that's because of concentration of similar pixels together. Anyway the easier differentiation process is attained through observing topological character of the patterns.

The goal of the paper is to propose an extension to MLP model that observes topology of input patterns to improve upon generalization capability of a neural network.

Manuscript received July 15, 2005.

J. Z. is with the Department of Computer Science, University of Latvia, Riga, Latvia (e-mail: janis.zuters@lu.lv).

II. The Classic MLP with Back-Propagation

MLP is regarded to be a reference point for neural networks, and back-propagation is the basis for training supervised neural networks. The core MLP operating and training algorithm is briefly shown in (1), (2), and (3) [1], [2].

Propagation function:

$$NET_j = \sum_{i=0}^{n} w_{ji} o_i \,, \tag{1}$$

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 (i=0: $o_i = 1$).

Activation function:

$$o_j = \varphi(NET_j) \,, \tag{2}$$

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

The general weight correction rule:

$$\Delta w_{ji} = \eta \delta_j o_i \,, \tag{3}$$

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.

(1) and (3) show the independence among input signals, so topology of neurons plays no role in operation of a neural network.

Since the plain back-propagation algorithm is regarded as very slow, there have been designed various enchantments to it in order to speed up the training process. The several of the most famous ones are: Rprop, Quickprop, and Cascade Correlation [1], [2]. Unfortunately the accelerated algorithms not always bring satisfactory results, thus in many cases people would rather prefer slower algorithms.

III. Extending MLP Model by Introducing Topology

The main idea of extending the MLP model: considering the input signals have certain relations (i.e., are not independent) – to make them to cooperate in order to yield better results. The cooperation is performed in a manner close to that of Kohonen network. Topology means that for every two neurons in a

layer the distance function d is known:

$$\forall i, j \exists d(i, j), \tag{4}$$

where d(·,·) is a distance function, that represents layer topology.

Similarly, the neighbourhood factor h can be defined between every two neurons, and a typical choice for that is the Gaussian function:

$$h(i, j) = \exp\left(-\frac{d^2(i, j)}{2\sigma^2}\right), \tag{5}$$

where h(·,·) is the neighbourhood factor between two neurons (or inputs); σ – the "effective width", a parameter for the Gaussian function.

Having defined the neighbourhood factor, the **extended propagation function** can be stated as:

$$NET_j = \sum_{i=0}^{n} w_{ji} o_i^*, \tag{6}$$

where o_i^* is the i-th **extended input** of the neuron j, defined by (7):

$$o_i^* = \left\{ \begin{array}{c} \sum_{k=1}^{n} h(i,k) o_k ; i \neq 0 \\ 1; i = 0 \end{array} \right\}, \tag{7}$$

where o_k is the k-th input of the neuron j.

Similarly the **extended weight correction rule** is obtained:

$$\Delta w_{ji} = \eta \delta_j o_i^*. \tag{8}$$

(7) shows all the inputs to theoretically cooperate; still in practice the neighbourhood function covers just the nearest neighbourhood.

IV. Experimental Results on Testing the Extended MLP Model

The goal of the experimentation was to show the new topology-based MLP model to improve upon generalization capability to compare to the classic model in certain circumstances.

A. Architecture of Neural Networks Used in Experiments

In computer experiments neural networks of the classic MLP architecture were used (Fig. 2) applying topology to the input layer:

- Size of the input signal – 1280.
- 1 hidden layer with 2..7 neurons.
- 2 neurons in the output layer.
- The input layer has a regular rectangular topology (32 × 40 nodes).
- Networks operate as described above: see (6), (2), (5), (7), (8). On practical considerations, (5) and (7) are adjusted below.
- Logistic function was used as an activation function with the gain $g = \frac{n}{35}$, where n – amount of neuron inputs.
- To evaluate a trained network, (9) computes error for a single output neuron on a fixed test pattern:

$$err_j^p = (d_j^p - o_j^p)^2, \tag{9}$$

where err_j^p – error of the output neuron j on test pattern p; d_j^p – desired value for the output neuron j on test pattern p; o_j^p – actual output value of the output neuron j on test pattern p.

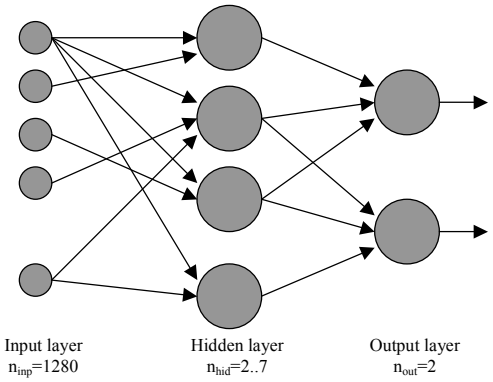

Fig. 2 Architecture of MLPs used in experiments

B. Adjusting the Model for Practical Considerations

To reduce the computational capacity of obtaining the extended input, the amount of addends in (7) was decreased eliminating those of a too small neighbourhood factor:

$$o_i^* = \left\{ \begin{array}{c} \sum_{k:\, h(i,k) \geq \Theta} h(i,k) o_k ; i \neq 0 \\ 1; i = 0 \end{array} \right\}, \tag{10}$$

where Θ – the minimum value of neighbourhood factor to be considered.

For normalization purpose, the computation of the neighbourhood factor h, defined in (5), was complemented by normalization factor:

$$h(i, j) = \tau \exp\left(-\frac{d^2(i, j)}{2\sigma^2}\right), \tag{11}$$

where τ – normalization factor (constant for a fixed σ).

C. Overview of the Experimentation

The model was tested on classification of black and white images of size 32 × 40 pixels with black digits depicted on a white background (Fig. 3).

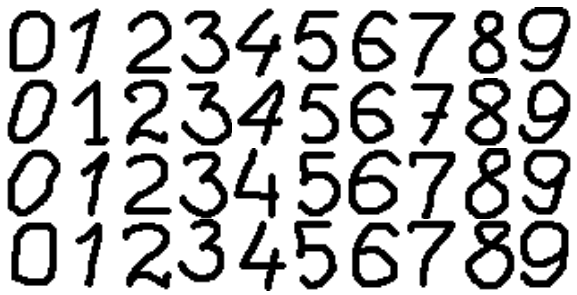

Fig. 3 Patterns available for training and testing

Description of the experimentation:

- Networks were trained to classify patterns (images of digits) into two categories.
- There were 3 types of experiments: (a) "1:9" (i.e. one digit into the first category, the nine remaining digits into the second category), (b) "3:7", (c) "5:5".
- 1000 experiments were made for each type, i.e. totally 3000 experiments.
- There were 2 neurons in output layer. The first one was trained to value 1 for the first category and to value 0 for the second category, the second one – vice versa.
- Each experiment gave 20 results (2 output neurons and 10 digits), so the total amount of units of experimental results was 60,000.

Course of an experiment:

- Choose at random 3 rows of available patterns (Fig. 3) as a training set, so those of the remaining row would be a test set.
- Choose at random parameter σ (11) among values {0="no topology"; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1.0}. When "no topology" is chosen, the classic algorithm is applied ((1), (3)) instead of the extended one.
- Choose at random the amount of neurons in the hidden layer (2 to 7).
- Choose at random patterns for the both of categories to train for. Note, that for simplicity digits for the both categories are chosen in consecutive order, e.g. {2,3,4,5,6} and {7,8,9,0,1}.
- Train the network until 100 epochs are passed or the average error of output neurons (11) decreases under $\varepsilon = 0.01$.
- The learning rate η depends from epoch u and is changed by the following rule (12):

$$\left\{ \begin{array}{c} \eta(0) = \eta^0 \\ \eta(u+1) = \eta(u) + \eta^- \cdot (\eta^* - \eta(u)) \end{array} \right\}, \quad (12)$$

where $\eta(\cdot)$ – learning rate function, η^0=0.1 – the initial rate, η^*=0.001 – the target rate, η^-=0.05 – the decay rate, u=0..99 – epoch number.

- Test the trained network on the set of test patterns (10 digits) and record the outputs of both output neurons.

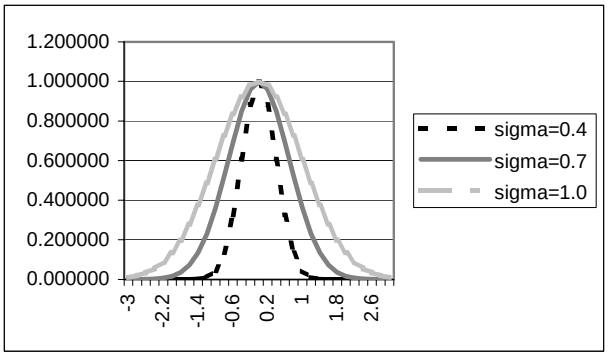

Fig. 4 A few of neighbourhood functions (5) used in experiments

TABLE I
AVERAGE OUTPUT ERRORS OF EXPERIMENTS GROUPED BY EXPERIMENT TYPE AND VALUE OF PARAMETER σ OF NEIGHBOURHOOD FACTOR

Experiment type		1:9	3:7	5:5
Classic MLP		0.036	0.090	0.101
MLP extended with input-layer topology	σ=0.4	0.035	0.092	0.110
	σ=0.5	0.028	0.075	0.085
	σ=0.6	0.028	0.076	0.100
	σ=0.7	0.027	0.092	0.115
	σ=0.8	0.037	0.068	0.091
	σ=0.9	0.031	0.068	0.088
	σ=1.0	0.028	0.091	0.098
	Average	0.031	0.080	0.098

D. Summary of Experimental Results

Acquired experimental results (error values according (9)) were grouped by the level of topology impact on the MLP model (classic model or a certain value of parameter σ) to make out the effect of exploiting topology in MLP model. The results are shown in Table I and in portions visualized in figures 5 and 6.

The experimental results show the topology-based approach to outmatch the classic one in most cases regarding output error level. It doesn't prove layer-topology based MLP to be better than classic MLP in terms of generalization capability. The experimental results just show the approach to be able to bring better results in certain circumstances.

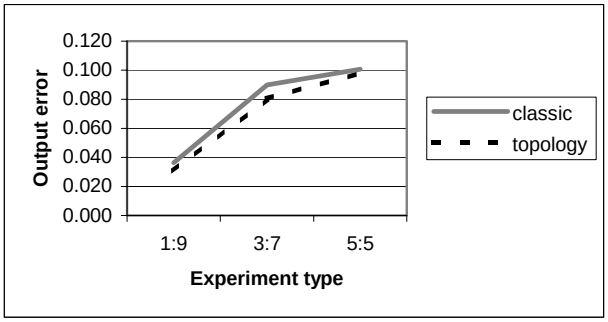

Fig. 5 Average output errors for classic MLP and MLP applying topology (rows 2 and 10 of Table I)

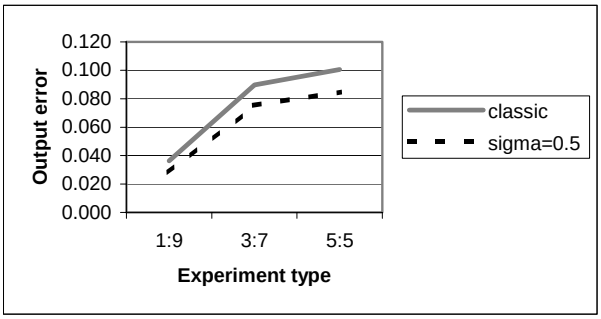

Fig. 6 Average output errors for classic MLP and MLP applying topology with σ=0.5 (rows 2 and 4 of Table I)

V. Considerations on Computational Complexity

Unfortunately the new topology-based approach requires extra computational resources to compare to the classic MLP with back-propagation.

Propagation function of the classic MLP (1) has the complexity of the degree O(n), where n – amount of weights in a neuron.

The pure extended propagation function ((6) and (7)) has $O(n^2)$ complexity, what is considered to be unsatisfactory.

Replacing (7) by (10) gives the complexity $O(n \cdot m)$, where m – size of neighbourhood. Considering (10), the size of neighbourhood is constant for a fixed neighbourhood parameter σ. If the minimum neighbourhood factor Θ=0.01 (10), then with σ=0.4 the size of neighbourhood is 5, but with σ=1.0 – 21. So the theoretical complexity is the same as with the classic MLP (O(n)).

In practice the new model works slower, still the slowdown with rather small value of parameter σ is regarded not to be significant.

VI. Conclusion

An extended model of MLP based on layer topology is proposed to improve upon generalization capability of a neural network. The effect of the model in a certain case is shown by the computer experiments described above.

The future researches would show the effect of the topology-based approach exploited with other conditions: (a) applying other topologies (e.g. hexagonal), (b) applying topology also for other layers (not only for input layer).

References

[1] Haykin, Simon, "Neural networks: a comprehensive foundation," 2nd ed. Prentice-Hall, Inc, 1999, pp. 2, 166–167, 205–206, 250.
[2] Scherer, Andreas, "Neuronale Netze: Grundlagen und Anwendungen," Vieweg, 1997, pp. 73–75, 81–87, 142–145.

Jānis Zuters has graduated from the University of Latvia, Department of Computer Science in 1999 (Mg. sc. comp.) and now is a PhD student here. He is a lecturer of information technologies at the University of Latvia and at the Vidzeme University College. The main field of his researches is artificial neural networks.