PRINCIPLE OF TRAINING ARTIFICIAL NEURAL NETWORKS USING BACKPROPAGATION ALGORITHM

Tomáš Ondrák
Doctoral Degree Programme, FEEC BUT
E-mail: xondra11@stud.feec.vutbr.cz

Supervised by: Jiří Skalický
E-mail: skalicky@feec.vutbr.cz

ABSTRACT

This paper describes learning process of artificial multi-layer neural network using *backpropagation* algorithm. To illustrate this process the three layer artificial neural network with two inputs, one hidden layer and one output is used.

1. INTRODUCTION

Backpropagation training algorithm was created by generalizing the Widrow-Hoff learning rule to multi-layer networks (fig.1) and nonlinear differentiable transfer functions. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as defined by you. Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities.

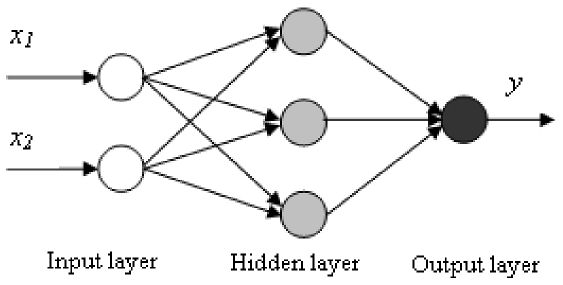

Figure 1: Multi-layer neural network

2. TRAINING BY BACKPROPAGATION ALGORITHM

Training of artificial neural network using backpropagation algorithm is established on training with teacher. This means, it’s necessary to make a set, which contain reference input values and reference output values. This set is called „**training set**“. Input value and corresponding output value shape pairs, which are called „**templates**“.

Real training sets usually has a hundreds to thousands training templates. It means that training set that training set don’t describe a problem absolutely but only in part. In this case it’s not a fault, because generalization of problems is the main attribute of artificial neural networks. This attribute secure, that similar input vectors gets from the neural networks same value. It’s necessary to templates represents all mains parts of image, but that can be a problem.

Backpropagation algorithm of neural network works on principles of training with teacher. This form is similar to way of teaching students in the school. If teacher established that student has a problem, than teacher have to press the student to learn the material. Neural network represents a student and the check is the mechanism which we can used to probe that output of network is the same like a require input. If the network don’t make a correct response, it’s necessary change values of weights so long, that the network will make a correct response. This principle is displayed on the figure 2.

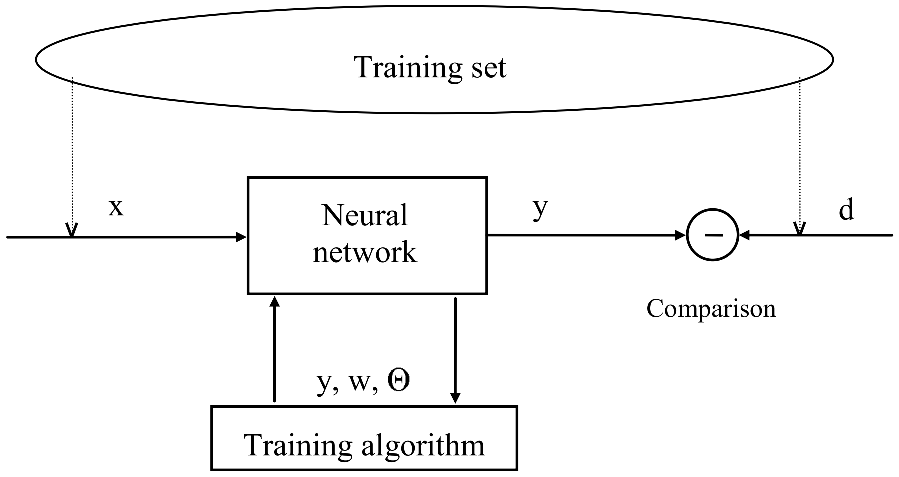

Figure 2: Principle of training backpropagation algorithm

Divergention between real and require values of neural network outputs for training set express rate of network precept – when the training is better, then divergention is lower. This rate of precept is called network energy. Training backpropagation algorithm minimalize this energy. This energy should be written:

$$E = \frac{1}{2}\sum_{i=1}^{n} \left(y_i - d_i \right)^2 \qquad (1)$$

where n is number of outputs, y_i at output i, d_i is required i output.

Backpropagation algorithm can be written in this points:

1. `Initialisation` – Weight in network is set accidentally in range <-0.3; 0.3>.
2. `Template presentation` – We take template from training set and give them on network inputs. Then step by step by layers from inputs to outputs count input of individual neurons by using this patterns:

$$y = f\left(\sum_{i=}^{n} w_i \cdot x_i + \Theta\right) \tag{2}$$

$$f(\varphi) = \frac{1}{1+e^{-\varphi}} \tag{3}$$

Pattern (3) is the nonlinear function called sigmoid and γ is coefficient describes steepness of curve in paratactic system beginning.

3. `Comparison` – We count energy by using this pattern

$$E = \frac{1}{2}\sum_{i=1}^{n} (y_i - d_i)^2 \tag{4}$$

, which is used like increase to total energy counted above all templates. Then we count error for input layer:

$$\delta = (d_i - y_i^0)\cdot\gamma\cdot y_i^0 \cdot (1 - y_i^0) \tag{5}$$

4. `Backward error diffusion and modification of weights` – for all neurons in layer:

$$\Delta w_{ij}^l(t) = \eta\ \delta(t)\cdot y_j^{l-}(t) + \alpha\ \Delta w_{ij}^l(t-) \tag{6}$$

$$\Delta\Theta_i^l(t) = \eta\ \delta(t) + \alpha\ \Delta\Theta_i^l(t-) \tag{7}$$

, where η is coefficient describes training speed (step length), α is inertia – it means how long we maintain specific direction, than we change direction by using a new gradient. Gradient is describe by this pattern:

$$\delta^{-1} = y_i^{h-1}\cdot(1 - y_i^{h-1})\cdot\sum_{k=1}^{n} w_{ki}^h \cdot \delta \tag{8}$$

Then we backward diffuse error to layer, which is immediate to inputs. On conclusion of this step we modificate weight and biases:

$$\begin{aligned} w_{ij}^l(t+1) &= w_{ij}^l(t) + \Delta w_{ij}^l(t) \\ \Theta(t+1) &= \Theta(t) + \Delta\Theta(t) \end{aligned} \tag{9}$$

This step is repeat for all network layers. We begin with output layer (l=0), than follow hidden layers (l=h). In case that we process hidden layer, which is closer to input layer, than replace y_j^{l-1} in pattern (6) to input value x_j.

5. `Termination of template range from training set` – If we show to network all templates from training set, then we can continue to step 6. Otherwise we go back to step 2.

6. `Termination of training process` – If the neural network energy is for last epoch smaller than our criterion, then we terminate training. Otherwise we go back to step 2.

As the algorithm's name implies, the errors (and therefore the learning) propagate backwards from the output nodes to the inner nodes. So technically speaking, backpropagation is used to calculate the gradient of the network error with respect to the network's modifiable weights. This gradient is almost always then used in a simple stochastic gradient descent algorithm to find weights that minimize the error. Often the term "backpropagation" is used in a more general sense, to refer to the entire procedure encompassing both the calculation of the gradient and its use in stochastic gradient descent. Backpropagation usually allows quick convergence on satisfactory local minima for error in the kind of networks to which it is suited.

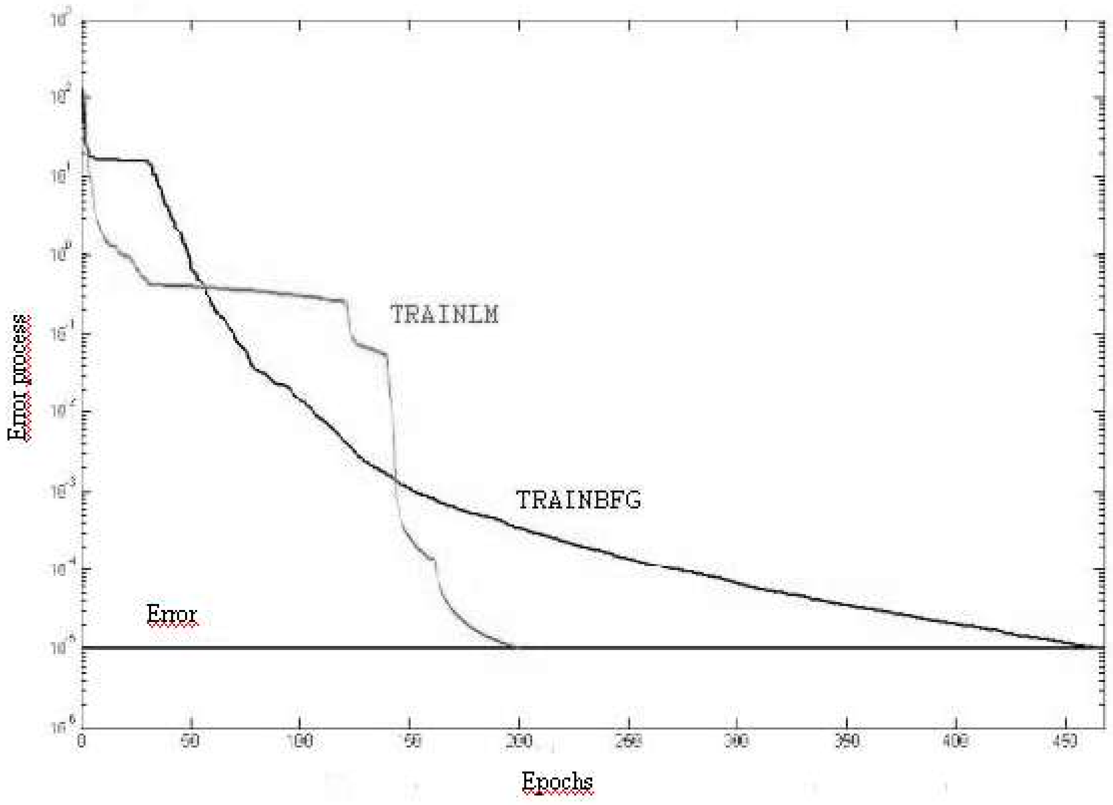

Figure 3: Process of training artificial neural network with Levenberg-Marquardt optimalization and quasi-Newton algorithms

3. CONCLUSION

Backpropagation algorithm is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). Backpropagation algorithm is name for group of algorithms (Levenberg-Marquardt optimalization, quasi-Newton, variable backward learning, adaptive learning rate…). The fastest of these algorithms are Levenberg-Marquardt optimalization and quasi-Newton. Training process of these algorithms is displayed on figure 3.

ACKNOWLEDGEMENT

This problem was solved with the support of the project MSM 0021630516 „Výzkum zdrojů, akumulace a optimalizace využití energie v podmínkách trvale udržitelného rozvoje“.

REFERENCES

[1] Demuth, H., Beale, M. Neural Network Toolbox User’s Guide Version 3. *The Mathworks Inc.*, USA 1998

[2] Klíma, J. Řízení synchronního motoru s permanentními magnety pomocí neuronové sítě. *Disertační práce*, Brno 2001

[3] Ondrák, T. Neuronové řízení stejnosměrného pohonu. *Diplomová práce*, Brno 2006

[4] Skalický, J.: Neuronové řízení v elektrických pohonech.

[5] http://www.speech.sri.com/people/ananand/771/html/node37.html

[6] http://www.cs.vsb.cz/jezek/vyuka/ns/backpropagation.html

[7] http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html