Neural Network application Pattern Classification Problems

Bohumir Jelinek

Institute for Signal and Information Processing
Mississippi State University
Mississippi State, MS 39762 USA
email: jelinek@isip.mstate.edu

ABSTRACT

There are two main reasons why Neural Networks (NN) are favorite pattern classification tool. First reason is that is has been proven that they can form arbitrarily complex decision regions. The second reason is that the their parameters can be trained to form the required decision region by the simple gradient descent method called back-propagation algorithm. This paper introduces the basic NN components focusing on multilayer NN, describes the feed-forward NN operation and back-propagation training algorithm and finally presents NN application to the pattern classification problems.

1. Introduction

Neural networks are built from simple units called *neurons* by analogy with the human brain. These units are connected by a set of weighted *connections*. Neural network learning is usually performed by a modification of the connection weights between units. The units are organized in layers. The first layer is called the *input layer*, the last one the *output layer*. Intermediate layers are called the *hidden layers* and resulting network is called *multi-layer perceptron* (Figure 1).

This article will present feed-forward neural networks, particularly multi-layer perceptrons. Their basic operation mode, called *feedforward operation*, is described in section 2. The review of the back-propagation algorithm is provided in section 3. The basic neural network training protocols are described in section 4. The experiments performed and results are explained in section 5. The summary is contained in section 6.

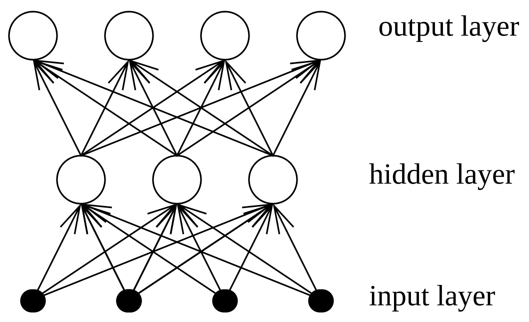

Figure 1: Multi layer perceptron with one hidden layer.

2. Feedforward Operation

The information to be analyzed is first presented to the neurons in the input layer and then propagated to the second layer for further processing. The result of this processing is propagated to the next layer and so on until the last - output layer. Each unit receives some information from the units in the previous layer, processes this information, and processing result is

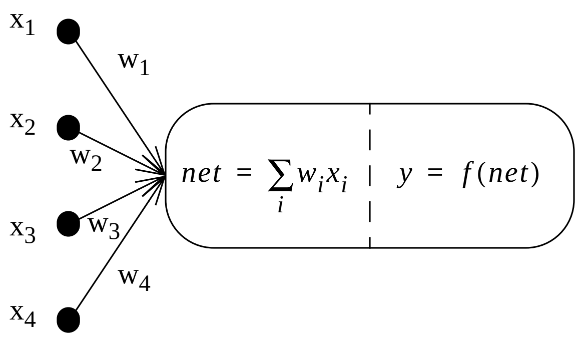

Figure 1: Basic processing unit

then propagated to the next layer. The standard processing unit first sums its weighted inputs to get so called *net activation* (denoted by *net*) and then computes a nonlinear function of this sum, as Figure 2 shows.

Most often used activation function are binary threshold

$$y = \begin{cases} 0 \ if \ x < 0 \\ 1 \ if \ x > 0 \end{cases} \tag{1}$$

sigmoid

$$y = sigmoid(x) = \frac{1}{1 - e^{-x}} \tag{2}$$

and *tanh* function

$$y = tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3}$$

Two last functions are related through

$$sigmoid(x) = \frac{tanh\left(\frac{x}{2}\right) + 1}{2} \tag{4}$$

3. Back-propagation Training Algorithm

The back-propagation algorithm is a gradient descent method minimizing the mean square error between the actual and target output of a multilayer perceptron. Assuming sigmoidal nonlinearity

$$f(net_i) = \frac{1}{1 - e^{-net}}, \tag{5}$$

the back-propagation algorithm consists of the following steps:

1. Initialize Weights and Offsets

Initialize all weights and node offsets to small random values.

2. Present Input and Desired Output Vector

Present continuous input vector **x** and specify the desired output **d**. Output vector elements are set to zero values except for that corresponding to the class of the current input.

3. Calculate Actual Outputs

Calculate the actual output vector **y** using the sigmoidal nonlinearity.

4. Adapt weights

Adjust weights by

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j x_i' \tag{6}$$

where x_i' is the output of the node i and δ_j is the sensitivity of the node j.

If node j is an output node, then

$$\delta_j = f'(net_j)(d_j - y_j) \tag{7}$$

where d_j is the desired output of the node j, y_j is the actual output and $f'(net_j)$ is the derivation of the activation function calculated at net_j.

If the node j is an internal node, then the sensitivity is defined as

$$\delta_j = f'(net_j) \sum_k \delta_k w_{jk} \tag{8}$$

where k sums over all nodes in the layer above the node j. Update equations are derived using the chain derivation rule applied to the LMS training criterion function.

Convergence can be faster if a momentum term is added and weight changes are smoothed by

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j x_i' + \alpha(w_{ij}(t) - w_{ij}(t-1)) \tag{9}$$

where $0 < \alpha < 1$.

5. Repeat by Going to Step 2

The simplest stopping criterion is to end when the change in the training criterion function is smaller then some preset value θ. A better approach is a cross-validation technique, stopping training when the error on a separate validation set reaches a minimum.

The performance of the back-propagation algorithm can be corrupted by finding a local minimum of the least mean square error function instead of the desired global minimum. Suggestions to improve the performance and avoid the occurrence of the local minima include allowing extra hidden units, lowering the gain term used to adapt weights, and making many training runs starting with different training sets of random weights.

4. Training Protocols

The three main training protocols are batch, stochastic and on-line training. In case of *on-line training* network weights are updated immediately after each new pattern presentation. *Stochastic training* is the same as on-line training, but patterns are chosen randomly from the training set. In *batch training* all patterns are presented to the network and then all weights are updated in one step. The advantage of on-line training is that there is no need to store large number of patterns in memory. The overall amount of pattern presentations is described as an *epoch* - one epoch corresponds to the presentation of all patterns from the training set. The number of training epochs generally indicates the relative amount of training.

5. Interpretation of Output Activation

It can be shown that in case of the infinite training data outputs of the neural network trained on 0-1 targets approximate the true *a posteriori* probabilities of the classes associated with the output units. Another key assumption of such output interpretation is that neural network have enough hidden units to be able to exactly represent posterior probability function. If, however, these conditions are not hold - in particular we have only finite amount of the training data - then the outputs will not represent a probabilities, it is not even guarantied that they sum to 1.0. If the sum of the output activations differs significantly from 1.0 in some range of the input space, it is an indication that the network is not accurately modeling the posteriors.

One approach toward approximating probabilities is to choose the output activation to be exponential rather than sigmoidal, and for each pattern to normalize the outputs to sum to 1.0 using the following formula:

$$z_k = \frac{e^{net_k}}{\sum_{m=1}^{c} e^{net_m}} \tag{10}$$

while using 0-1 target signals. This is a *softmax* method - a smoothed continuous version of a *winner-take-all* nonlinearity in which the maximum output is transformed to 1.0 and all others are reduced to 0.0. If such a trained network is going to be used on the data where the priors have been changed, it can lead to the performance degradation. Possible solution is to rescale each network output by the new prior class probabilities or their ratio. This can be accomplished only if the prior class probabilities in the test set are known in advance. It can be one of the reason for the not very good neural network performance on the presented tasks.

6. Experiments and Results

The goal of the first experiment is to train a classifier on a provided training set containing 528 ten-dimensional input vectors assigned to 11 categories. The development training set contained 379 testing vectors. The final evaluation set has 83 vectors.

The second experiment assumes that vectors provided as a training and testing set have a temporal dependence. Training set consist of 925 vectors belonging to 5 classes. Development set has a 350 vectors and final evaluation set has 225 test vectors. The training and test data are summarized in Table 1.

Table 1:

	Experiment 1 static classification	Experiment 2 temporal modeling
vector dimension	10	39
number of classes	11	5
training set	528	925
development set	379	350

Table 1:

	Experiment 1 static classification	Experiment 2 temporal modeling
evaluation set	83	225

Several techniques have been applied to improve the recognition performance, including data normalization, on-line, batch and stochastic training, sigmoidal and tanh activation function, different number of hidden units.

The performance on Experiment 1 is shown in the Table 2.

Table 2:

# of hidden units	# of iterations	activation	classification error [%]
50	25	tanh	47.23
50	50	tanh	50.13
24	25	sigmoid	47.76
24	75	sigmoid	46.97
10	50	sigmoid	45.65
10	75	sigmoid	46.70

The experiments shown that the neural networks are not performing well on the tasks with insufficient training data available. We can also see that neural network with the more hidden units tends to overtrain faster. Results of the experiments are sensitive to the random initialization, the relative difference in error rate is in 20% range

It is necessary to perform many experiments until the optimal neural network structure is found. The usual way of finding an optimal network structure is experimental. The correct parameters are found by finding an optimal value in a large searching space. SNNS neural network tool provides many different neural network configurations. The tool could be fully exploited after the theory behind the different approaches provided by the tool is learned - it requires a time. SNNS 300 pages user manual is a reference of the required theory. The output of the classification are posteriori probabilities of the class, and it is necessary to find the highest probability class.

7. Acknowledgments

I would like to thank to my wife Veronika for encouraging me to work and for bearing all of my problems together with me.

I give many thanks to my supervisor Dr. Joseph Picone for forcing me to do painful steps to increase the quality of my personality and life.

References

[1] H. Abdi, D. Valentin, B. Edelman: *Neural Networks*, SAGE Publications, Thousand Oaks, California, USA, 1999.

[2] M. Anthony, P.L. Bartlett, *Neural Network Learning: Theoretical Foundations*, Cambridge University Press, 1999.

[3] R.E.Duda, P.E.Hart, D.G.Stork, *Pattern Classification*, Second Edition, John Wiley & Sons, 2001.

[4] F.Jelinek, S*tatistical Methods for Speech Recognition*, MITT Press, 1997.

[5] R. Lippmann, “An Introduction to computing with Neural Nets,” *IEEE Signal Processing Magazine*, pp. 4-22, April 1987.

[6] G.B. Orr, K.-R. Muller, *Neural Networks: Tricks of the trade, Lecture Notes* in Computer Science, Springer-Verlag, Berlin, Germany, 1998.

[7] S.M. Ross, Introduction to Probability Models, Seventh Edition, Academic Press, 2000.

[8] Tebelskis, J.: Speech recognition using neural networks, partial Ph.D. thesis, Carnrgie Mellon University Pittsburgh, Pennsylvania, USA, 1995

[9] SNNS - Stuttgart Neural Network Simulator, User Manual, Version 4.2, http://www-ra.informatik.uni-tuebingen.de/SNNS/