# Induced Weights Artificial Neural Network

Slawomir Golak

Sielsian University of Technology, Electrotechnology Department,
Division of Informatics and Modeling of Technological Processes
sgolak@polsl.pl

**Abstract.** It is widely believed in the pattern recognition field that the number of examples needed to achieve an acceptable level of generalization ability depends on the number of independent parameters needed to specify the network configuration. The paper presents a neural network for classification of high-dimensional patterns. The network architecture proposed here uses a layer which extracts the global features of patterns. The layer contains neurons whose weights are induced by a neural subnetwork. The method reduces the number of independent parameters describing the layer to the parameters describing the inducing subnetwork.

## 1 Introduction

The great potential of the neural networks is most frequently used in pattern recognition. The most challenging problem here is achieving the proper generalization. Typical images and time-series are usually large, often with several hundred variables. Fully connected, unrestricted networks do not work well as far as recognizing such large patterns is concerned.

The number of examples needed to achieve an acceptable level of generalization ability is dependent on the intrinsic entropy of the chosen architecture, and can be decreased by reducing the number of independent parameters needed to specify the network configuration. One of the ways to improve generalization is a reduction of the network structure on the base of a pruning algorithm (e.g. the Optimal Brain Damage [7]).

Another deficiency of the fully-connected architectures is that the topology of the inputs is entirely ignored. In fact, images have a strong 2D structure, while time-series have a strong 1D structure. Pixels, or variables, spatially or temporally adjacent are correlated.

The application of a specialized network architecture, instead of a fully-connected net, can reduce the number of free parameters.

There are many papers that propose specialized network architectures for the recognition of large patterns. Convolutional networks, for instance, use the techniques of local receptive fields and shared weights. These networks extract and combine local features of pattern [4,5]. Principal component analysis transforms a number of correlated variables into a smaller number of uncorrelated variables called principal components and is frequently adopted for dimensionality reduction [1].

W. Duch et al. (Eds.): ICANN 2005, LNCS 3697, pp. 295 – 300, 2005.

The idea that has been followed in the IWANN is based on the invention of dynamically-calculated weights, which results in giving the individual neurons the ability to transform large patterns, using only a limited number of parameters describing their connection weights. The proposed network architecture extracts and transforms the global features of the patterns.

## 2 Network Architecture

The IWANN network makes use of dynamically-calculated connection weights. As a result, the number of parameters describing neural network connections is reduced.

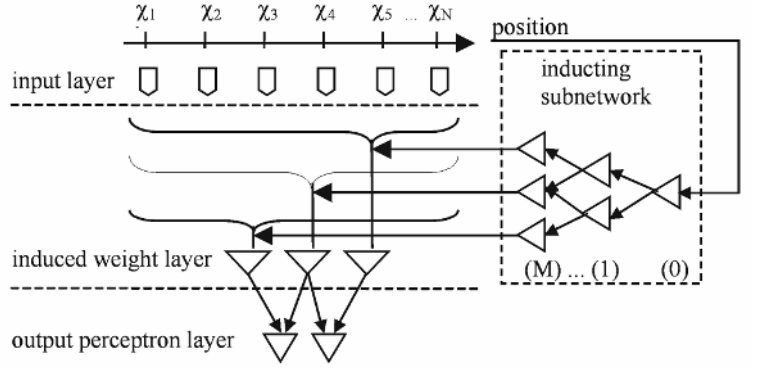


**Fig. 1.** Induced weights network scheme

The input layer of the network proposed can be one- or multidimensional, and every neuron in this layer is described by its geometric position.

The layer is a data source for the induced weights layer. It contains radial basis neurons which apply the Gaussian transformation function. The input of this radial basis transformation function is the Euclidean distance between the input vector and the vector of weights calculated by the inducing subnetwork, multiplied by the bias.

$$y_i^{(1)} = \varphi\left( b_i^{(1)} \sqrt{\frac{1}{2}\sum_{j=1}^{N^{(0)}} \left( \dot{y}_{ji}^{(\dot{M})} - y_j^{(0)} \right)^2} \right) \tag{1}$$

where, $y_i^{(1)}$ output of the i-th neuron in induced weights layer, $\varphi()$ Gaussian transformation function, $b_i^{(1)}$ bias of neuron, $\dot{y}_{ji}^{(\dot{M})}$ output of neuron in output (M-th) layer of inducing network equal to the weight of the j-th input of the i-th neuron of the induced layer, and $y_j^{(o)}$ output of j-th input neuron – network input.

The task of the inducing neural subnetwork consists in positioning of high-dimensional centers. The inducing network is a multilayer perceptron:

$$\dot{y}_{ji}^{(\dot{m})} = f\left( \sum_{k=1}^{\dot{N}^{(\dot{m}-1)}} \dot{y}_{jk}^{(\dot{m}-1)} \dot{w}_{ik}^{(\dot{m})} \right) \tag{2}$$

where, $\dot{y}_{ji}^{(\dot{m})}$ output of i-th neuron in the m-th layer for j-th network input, $\dot{w}_{ik}^{(\dot{m})}$ weight of the k-th input of the i-th neuron in m-th layer.

The number of neurons in the inducing network output layer is equal to the number of neurons in the induced weights layer. Geometrical positions of neurons in the input layer are introduced to the input layer of the inducing neural network:

$$\dot{y}_{jk}^{(0)} = \chi_{jk} \tag{3}$$

where, $\dot{y}_{jk}^{(0)}$ value of inducing network k-th input, $\chi_{jk}$ - k-th coordinate of j-th input neuron

The inducing network calculates connection weights between every neuron in the induced and input layer by using coordinates' values of the input neurons.

Remaining output layers of the IWANN network are perceptron layers:

$$y_i^{(m)} = f\left( \sum_{k=1}^{N^{(m-1)}} y_k^{(m-1)} w_{ik}^{(m)} \right) \tag{4}$$

where, $y_i^{(m)}$ output i-th neuron from m-th layer (m>1), $\dot{w}_{ik}^{(\dot{m})}$ weight of the k-th input of the i-th neuron from m-th layer.

The similar idea is used in the mixture of experts model, were weights of the gating function depend on the output of the gating network [2,3]. However, in the ME algorithm, the input of gating network are the global network inputs' values and not geometrical coordinates of these inputs. Furthermore, the gating function is a linear or non-linear weight function, while the induced layer uses a distance function.

## 3 Learning Algorithm

We can express the training error E as a function (5). If the transformation functions used in the network are continuous and differentiable, it is possible to calculate derivatives of this error function. Therefore, the proposed network can trained with the use of gradient learning methods.

$$E = \frac{1}{2} \sum_{i=1}^{N^{(M)}} \left( y_i^{(M)} - d_i \right)^2 \tag{5}$$

where, $d_i$ and $y_i^{(M)}$ are the target and output values for training example.

The gradient of the error function specifies the vector in whose direction the greatest increase in E can be obtained. Our aim is to calculate the partial derivatives of error function for each weight of the network. The algorithm to calculate the error derivatives for perceptron output layers is well-known:

$$\delta_i^{(m)} = \sum_{k=0}^{N^{(m+1)}} \left(\delta_k^{(m+1)} w_{ki}^{(m+1)}\right) f'\left(\sum_{k=1}^{N^{(m-1)}} y_k^{(m-1)} w_{ik}^{(m)}\right) \tag{6}$$

$$\frac{\partial E}{\partial w_{ik}^{(m)}} = \delta_i^{(m)} y_k^{(m-1)} \tag{7}$$

where, $\delta_i^{(m)}$ signal error of i-th neuron in m-th perceptron layer, $\delta_k^{(m+1)}$ signal error of k-th neuron in next layer, $w_{ki}^{(m+1)}$ connection weight between i-th neuron in current layer and k-th neuron in the next layer, $w_{ik}^{(m)}$ weight of the i-th input of the k-th neuron in the current layer, $y_k^{(m-1)}$ output of neuron in the previous layer.

Applying the algorithm, we can obtain signal errors for the neurons of the induced layer, and the derivatives for biases of these neurons:

$$\delta_i^{(1)} = \sum_{k=0}^{N^{(2)}} \left(\delta_k^{(2)} w_{ki}^{(2)}\right) \varphi'\left(b_i^{(1)} \sqrt{\frac{1}{2}\sum_{j=1}^{N^{(0)}} \left(\dot{y}_{ji}^{(\dot{M})} - y_j^{(0)}\right)^2}\right) \tag{8}$$

$$\frac{\partial E}{\partial b_i^{(1)}} = \delta_i^{(1)} \sqrt{\frac{1}{2}\sum_{j=1}^{N^{(1)}} \left(\dot{y}_{ji}^{(\dot{M})} - y_j^{(0)}\right)^2} \tag{9}$$

The values of signal error may be used to calculate errors of neurons in the output layer of the inducing network. They have to be calculated individually for every j-th input of the induced layer:

$$\dot{\delta}_{ji}^{(\dot{M})} = \delta_i^{(1)} b_i^{(1)} \frac{1}{2\sqrt{\frac{1}{2}\sum_{j=1}^{N^{(0)}} \left(\dot{y}_{ji}^{(\dot{M})} - y_j^{(0)}\right)^2}} \left(\dot{y}_{ji}^{(\dot{M})} - y_j^{(0)}\right) f'\left(\sum_{k=1}^{\dot{N}^{(\dot{M}-1)}} \dot{y}_{jk}^{(\dot{M}-1)} \dot{w}_{ik}^{(\dot{M})}\right) \tag{10}$$

Signal errors of neurons in the remaining layers are calculated by using the same backpropagation method as in (6).

The derivatives for the weights of a neuron of the inducing network are the sums of the derivatives calculated for every j-th input of the network:

$$\frac{\partial E}{\partial \dot{w}_{ik}^{(\dot{m})}} = \sum_{j=1}^{N^{(0)}} \dot{\delta}_{ji}^{(\dot{m})} \dot{y}_{jk}^{(\dot{m}-1)} \tag{11}$$

Owing to the evaluation of these derivatives, various gradient-based optimization methods can be utilized (e.g. QuickProp, RPROP).

## 4 Experiments

The proposed neural network was applied to classification of one- and two-dimensional patterns. The first dataset consisted of 200 artificially generated one-dimensional patterns of size 100. These data were evenly divided into the training and test sets. The dataset contained four classes of patterns. Figure 2a shows examples of patterns in the dataset.

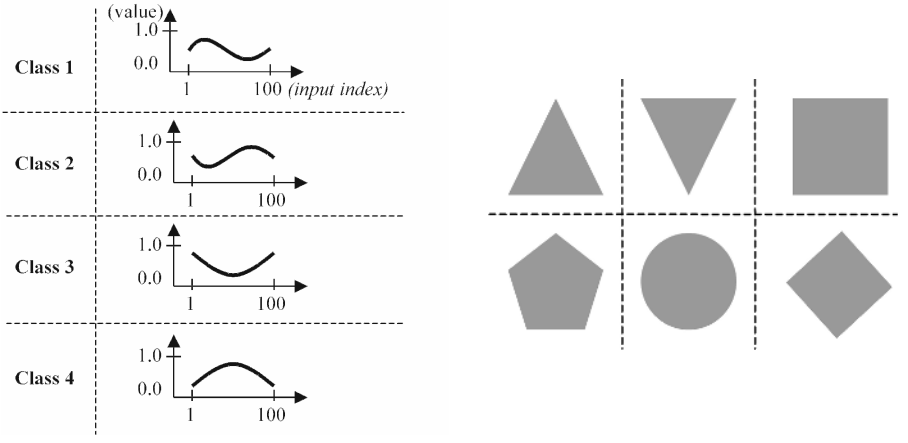


**Fig. 2.** Classes of patterns in dataset 1 (a) and dataset 2 (b)

The base patterns were randomly scaled (60÷140%), translated horizontally (±25) and vertically (±0.25).

The architecture of the neural network consisted of an induced layer with 4 neurons and an output perceptron layer with 4 neurons. The inducing subnetwork was a multilayer perceptron. This neural network consisted of three layers and the number of neurons in the hidden layer was 10. The error function was minimized by the RPROP method [6]. The training was stopped after 340 epochs. At the end of the training, the mean square error was 0.0022 and the percentage of wrong classifications was below 1%. Results of a classic multilayer perceptron with the optimized structure (100x20x4) were 0.0352 (MSE) and 16% (wrong classifications).

The second dataset consisted of 200 two-dimensional patterns of size 50x50. The patterns were evenly divided into the six classes and into the training and test datasets.

Figure 4b contains examples of patterns in the six classes. The base patterns were randomly transformed by rotation (±5°) and translation of polygons vertexes (±5 pixels). Uniformly distributed random noise was introduced to patterns at the level of 20%.

The network consisted of the induced layer with 10 neurons, the output layer with 6 neurons and the inducing subnetwork with 25 neurons in the hidden layer.

The network was trained by the RPROP method [6]. After 1230 epochs the mean square error was 0.0034 and the percentage of wrong classifications was equal to 3%. Results of the multilayer perceptron (2500x40x6) were 0.1134 and 41%.

## 5 Conclusions

The experiments described in this paper show that the IWANN network is a suitable model for the classification of large patterns. The presented algorithm may be of great help to the network designers in their time-consuming task of preprocessing the patterns. The method reduces the number of independent parameters of the induced layer to the parameters describing the biases of neurons in this layer and parameters describing the inducing subnetwork. Owing to this, it is possible to obtain satisfactory results of classification for high-dimensional patterns with the help of a limited number of learning examples.

## References

1. Diamantaras K. I., Kung S. Y.: Principal Component Neural Networks: Theory and Applications, Wiley, 1996
2. Jacobs R. A., Jordan M. I.: Adaptative Mixture of Local Expert, Neural Computation, v. 3, pp. 79-87, 1991
3. Jordan M. I., Jacobs R.A.: Herarchical mixtures of experts and the EM algorithm, Neural Computation, v. 6, pp. 181-214, 1994
4. LeCun Y., Bengio Y. Convolutional networks for images, speech, and time series, The Handbook of Brain Theory and Neural Networks, pp. 255-258. MIT Press, Cambridge, Massachusetts, 1995
5. LeCun Y., Bottou L., Benigo Y., Haffner P.: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, v. 86, pp. 2278-2324, 1998
6. Riedmiller M. and Braun H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In Proceedings of the IEEE International Conference on Neural Networks 1993 (ICNN 93), 1993
7. Solla S., LeCun Y., Denker J.: Optimal Brain Damage, Advances in Neural Information Processings Systems 2, pp. 598-605, San Mateo, Morgan Kaufmann Publishers Inc., 1990