

Capabilities and Limitations of Feedforward Neural Networks with Multilevel Neurons

Aleksander Malinowski Tomasz J. Cholewo Jacek M. Żurada

Computer Science and Engineering Program
University of Louisville, Louisville, KY 40292, USA
phone: (502) 852-6289, fax: (502) 852-6807
e-mail: jmzura02@starbase.spd.louisville.edu

ABSTRACT

This paper proposes a multilevel logic approach to output coding using multilevel neurons in the output layer. Training convergence for a single multilevel perceptron is considered. It has been found that a multilevel neural network classifier with a reduced number of outputs is often able to learn faster and requires fewer weights. Concepts are illustrated with an example of a digit classifier.

I. INTRODUCTION

Most neural networks are composed of neurons using bilevel activation function with two regions of saturation of output. Activation functions are typically defined as sigmoidal (continuous), signum (piecewise continuous) or Gaussian. It is possible, however, to use an activation function with multiple saturation levels in a multilevel neuron (MN). Such neurons in the output layer of a binary classifier would allow coding of more states while using the same number of neural network outputs. This feature could facilitate an extensive reduction in the size of neural network models.

One of the most common applications of neural networks is pattern classification. The correct behavior of neural classifiers is possible due to the generalization ability of neural networks and the existence of saturation regions of the activation functions [1, 2]. Properly trained classifiers are characterized by a very low sensitivity of outputs with respect to inputs [3, 4, 5]. This feature allows the interpretation of responses as integer class indices even for imprecise (fuzzy) inputs. The expected benefit of a MN would be in providing higher coding density of output values while maintaining the property of saturation at those values.

A MN was first proposed for a Hopfield-type neural network in [6] and then pursued in [7]. This paper extends the use of a MN for multilayer perceptron network architectures. The conditions of training convergence and a

training algorithm are proposed for the multilevel perceptron. Examples of single- and multi-layer neural network classifiers are discussed. The possibility of neural network size reduction is reviewed.

II. MULTILEVEL NEURON

Let us consider a continuous neuron with a multiple level activation function as proposed in [7]. The activation function of a simple bilevel continuous neuron is given by

$$f(net) = 2f_s(net) - 1$$

with the commonly used elementary component f_s defined as

$$f_s(x) = \frac{1}{1 + e^{-\lambda x}}.$$

One of the possible forms of the activation function of the three-level neuron results from combining two elementary components:

$$f(net) = f_s(net + 1/2) + f_s(net - 1/2) - 1 \quad (1)$$

This function, shown in Fig. 1a, has two inflection points at $net = -0.5$ and $net = 0.5$ for λ sufficiently high.

From the point of view of function approximation there is no difference between two- and multi-level neurons because in this case neurons do not work in their saturation region and have continuous outputs. For classifiers, however, it is important that the output values of output layer neurons lie in the saturation region. Therefore, the outputs of the neural network classifier do not change significantly when input patterns become fuzzy. The neuron activation function proposed in formula (1) possesses this feature for the three values of the neuron's output as is shown in Fig. 1a. Therefore, it is potentially useful for classifiers, and allows multilevel logic coding of network outputs.

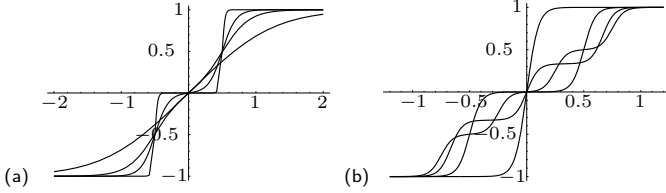


Figure 1: Activation function of a multilevel neuron: (a) three-level neuron activation functions for $\lambda = 2, 5, 10$ and 50 ; (b) multilevel neuron activation functions for $L = 1, \dots, 4$ and $\lambda = 20$.

The idea of a MN can be extended to any number of levels. The general formula for inflection points of an activation function is given by

$$\Theta_l = \frac{1}{L} + 2\frac{l-1}{L} - 1, \quad l = 1, \dots, L$$

where L is the number of components. The MN activation function is then defined as

$$f(\text{net}, L) = \frac{2}{L} \sum_{l=1}^L f_s(\text{net} + \Theta_l) - 1 \quad (2)$$

The family of such functions is shown in Fig. 1b. It should be noted that the computation cost for evaluating the output of the MN having L levels is L times larger than for a simple neuron. A potential advantage could be a reduction of the number of neurons in the hidden layer which could lead to a smaller number of weights.

III. TRAINING OF SINGLE MULTILEVEL PERCEPTRON

Consider one multilevel neuron having three levels ($L = 2$). It can be trained using a simple perceptron learning rule with some restrictions. Let us compare the learning of a binary and a three-level perceptron.

Fig. 2 shows the input pattern (Fig. 2a) and the weight solution region (WSR) in the weight space. In case of an ordinary neuron tuned to recognize one input pattern, WSR is bounded by a line in two-dimensional space (Fig. 2b), or by a hyperplane in the case of more dimensions. Neurons with 3 or more levels possess more boundaries (Fig. 2c). There is a boundary line between each two neighboring output levels for a particular input pattern. States -1 and 1 are privileged because they correspond to half-planes in the weight space while other levels correspond only to stripes. Learning is not possible for some other labeling of the decision regions, e.g., as in Fig. 2d.

The standard perceptron training rule can be applied to the proposed MN. Let $\mathbf{x} = [x_1, x_2, \dots, x_I]^T$ be an input vector, $\mathbf{w} = [w_1, w_2, \dots, w_I]^T$ a neuron weight vector, o a neuron output and d its desired output for a given input pattern. The neuron output can be evaluated as

$$o = f(\mathbf{w}^T \mathbf{x}, L).$$

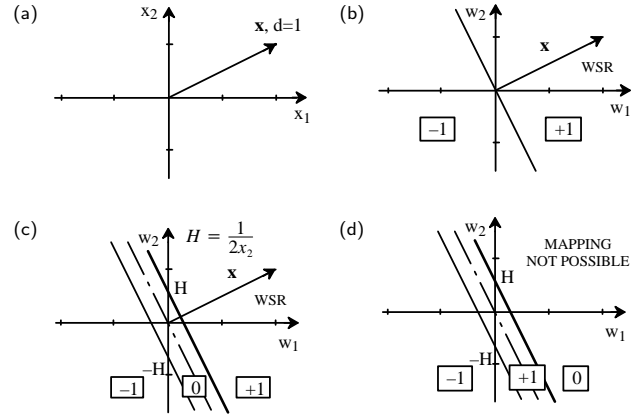


Figure 2: Input pattern for perceptron training and weight solution region for perceptron: (a) input pattern; (b) weight solution region for a simple perceptron; (c) weight solution region for a three-level perceptron; (d) labeling for which WSR does not exist. Width H of class 0 strip depends on weights' values and the width of the middle saturation region of activation function.

The neuron should respond with output o close to the desired value d . This response can be achieved through an iterative process of weight modification, starting from any initial weight value using formula

$$\mathbf{w}' = \mathbf{w} + \frac{1}{2} c \mathbf{x} (d - o) \quad (3)$$

where c is a learning constant. The above rule modifies the weights of the neuron according to classification or misclassification.

Let us determine whether for a given multilevel neuron (multilevel perceptron) using the activation function (2) whether there exists a learning constant c allowing the perceptron to be trained to recognize an input pattern as belonging to an arbitrary set class using formula (3).

For simplicity, consider a MN with two inputs, $L = 2$ (three levels), and a pattern \mathbf{x} to be classified. Assume the initial weights are outside the solution region. Fig. 3 shows the weights' movements during the neuron training for a binary perceptron and for a MN for $d = 1$, $o = -1$ (or $d = -1$, $o = 0$ or $o = 1$). For correct classification no weight modification is performed. When the neuron output is erroneous, the weights are adjusted according to formula (3).

The proof of learning convergence for a binary perceptron has been published in [1, 2]. While analyzing perceptron learning, several cases of desired (d) and obtained (o) output values can be considered. For the classical perceptron there are four possibilities: $o = \pm 1$, $d = \pm 1$. For a MN there are additional combinations such as $o = 0$, $d = \pm 1$, and $o = \pm 1$, $d = 0$.

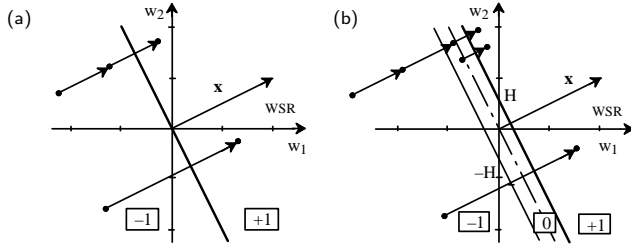


Figure 3: Weight changes during perceptron training and weight solution regions: (a) classical perceptron, $d = 1$ and $o = -1$; (b) three-level perceptron, $d = 1$ and $o = 0$ or 1 .

Case 1: $d = 1$, $o = -1$ or 0 . The perceptron weights \mathbf{w} are initially placed outside the WSR as shown in Fig. 3b. Expression $d-o$ from the formula (3) is positive and the vector \mathbf{w} will be moved toward WSR. If the learning constant c is large enough, \mathbf{w} can be moved directly into WSR, or otherwise through the area $o = 0$. In this second case the movement toward WSR will be with stopovers. The proof is therefore analogous to the case of the classical perceptron shown in Fig. 3a.

Case 2: $d = -1$, $o = 1$ or 0 . This case is similar to the previous one. The WSR in Fig. 3b, however, will be in the area labeled -1 instead of 1 , and the weights \mathbf{w} will move from the area of 1 to -1 through the area of 0 .

Case 3: $d = 0$, $o = \pm 1$. This case is illustrated in Fig. 4a. Let $o = -1$. The value of $d - o$ is positive. Vector \mathbf{w} will move toward the WSR with the step size controlled by the learning coefficient c . If this constant is too large, however, the weights \mathbf{w} will go directly into region 1 . The next iteration will bring the weights back to the starting point since the distance between -1 and 0 is equal to the distance between 0 and 1 . This will initiate weight oscillation and training will not succeed.

There is a solution to this oscillation problem; however, it is beyond the capability of a discrete multilevel perceptron with a fixed learning constant c . If the MN activation function is assumed continuous, the error decreases with decreasing distance to the WSR and the oscillation will decay as training progresses (Fig. 4b).

Another limitation of a MN involves linearly separating of the classes. In the case of a regular bipolar perceptron, the only restriction on the learning convergence was the ability to dichotomize the input space by a hyperplane into areas corresponding to different classes. In the case

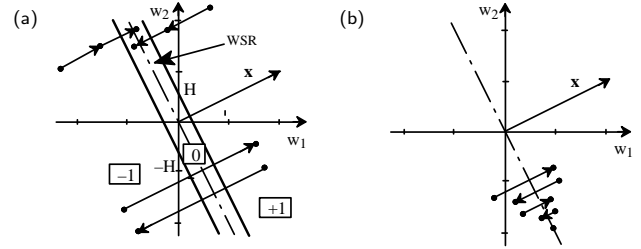


Figure 4: Weight changes during perceptron training and weight solution regions for $d = 0$ and $o = \pm 1$: (a) discrete three-level perceptron; (b) continuous three-level perceptron.

of MN, there is an additional restriction on class labeling. The two half-planes have to be assigned to classes -1 and 1 , and the area between them to class 0 (Fig. 2d). Otherwise, a single MN is not able to learn and training will not converge.

If weights leading to the perceptron are modified according to the formula (3), the final weights placed in WSR are given by

$$\mathbf{w}^* = \mathbf{w}_0 + \frac{1}{2}c \sum_{p=1}^P \sum_{n=1}^N (d_p - o_p^{(n)}) \mathbf{x}_p$$

where P is the number of input patterns, N is the number of iterations, \mathbf{w}_0 are the initial weights, \mathbf{w}^* are the weights in WSR, and $o_p^{(n)}$ is a neuron's output in the n -th iteration for the p -th pattern. For a binary perceptron this expression can be simplified to

$$\mathbf{w}^* = \mathbf{w}_0 + c \sum_{p=1}^P k_p d_p \mathbf{x}_p$$

where k_p is the number of misclassifications for the p -th pattern during perceptron training.

IV. MULTILAYER FEEDFORWARD NEURAL NETWORK AND MULTILEVEL NEURONS

A MN can also be applied also to multilayer feedforward neural networks (MFNN), resulting in a multilayer multilevel feedforward neural networks (MMFNN). As mentioned before, not all neurons in such a network have to be multilevel. MFNN classifiers often do not operate internally in a saturation mode [1]. Several experiments were performed to demonstrate that hidden layer neurons often remain nonsaturated (their results are omitted as being beyond the scope of this paper). Given such behavior of hidden neurons, there is no justification for using a MN in the hidden layers. The main advantage of a MN is then in reducing the size of the output vector of the classifier.

V. NUMERICAL EXAMPLES

An MMFNN was constructed and applied to digit image classification. 81-dimensional patterns (including an augmented input) of 10 digits from Fig. 5 were presented to the classifiers with one hidden layer and different types of output neurons. Neural networks with 10 binary output neurons coded as one of ten, 7 binary output neurons coded with ASCII codes, and one multilevel neuron with 10 levels ($L = 9$) have all been trained successfully. In the first two cases the minimum number of hidden layer neurons was about 10. In the last case, two continuous bilevel neurons were sufficient for correct classification. Fig. 6 shows the learning profiles for “one of ten” and MN classifiers.

0123456789

Figure 5: Patterns presented to the digit classifier. Size of single digit image is 8×10 pixels.

To achieve the same distance between classes in both cases, the MN neural network was trained to an error 10 times smaller than the binary neuron network. The MMFNN digit classifier required a smaller number of training steps than the classical MFNN. An MMFNN with one neuron having $L = 9$ needs as many computations of the simple neuron activation function f_s as 9 bipolar neurons. However, the resulting number of weights is 10 times smaller. Therefore, the computation effort for evaluating the output layer is smaller than in the bilevel approach. Furthermore, the number of weights decreases from 110 to 11 in the case of 10 hidden neurons. After applying the ultimate minimum number of hidden layer neurons, the weight number drops from 110 to 3. The number of weights in the hidden layer has also decreased from 810 to 162. However, this network reduction comes at a cost of increased number of learning cycles. In the case of digit classifier, the number of iterations increased by factor of 3.

Such spectacular reduction is not always easy to achieve because applying MN with a higher number of levels requires more accurate training. This sometimes causes convergence problems. The network then becomes more sensitive to the weight changes, and decreasing the learning constant has often been found necessary to implement learning.

VI. CONCLUSIONS

The multilevel neuron and its application to perceptron training was introduced with the proof of convergence for the simple three-level neuron case. The tested application of the digit classifier has been rather promising. Two important advantages of MMFNN are the reduction of computational effort and memory capacity needed to store

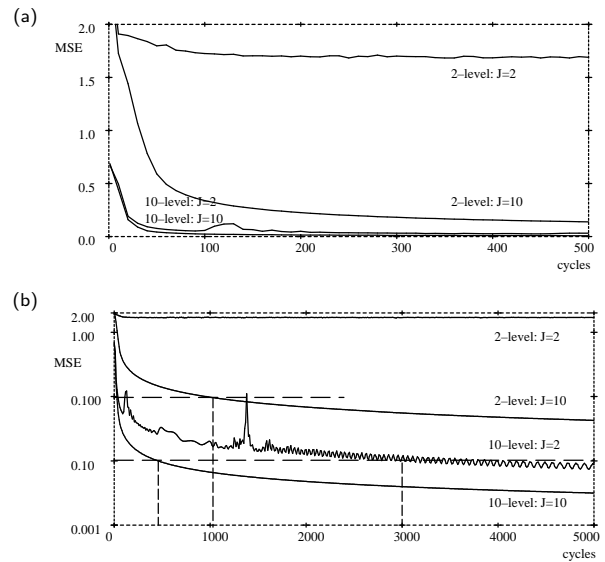


Figure 6: Learning profiles of the 10 digit classifiers for different numbers of hidden neurons: 1 of 10 classifier and one ten-level MN. In both cases $J = 2$ and $J = 10$ hidden layer neurons: (a) linear scale on the error axis; (b) logarithmic scale on the error axis; dashed lines show error levels for training termination conditions.

the weights. In addition, the method offers a possibility of VLSI implementation complexity reduction, especially leading to smaller size networks.

REFERENCES

- [1] J. M. Żurada, *Introduction to Artificial Neural Systems*. St. Paul, Minnesota: West Publishing Company, 1992.
- [2] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, 1990.
- [3] J. M. Żurada, A. Malinowski, and I. Cloete, “Sensitivity analysis for pruning of training data in feedforward neural networks,” in *Proc. of The First Australian and New Zealand Conference on Intelligent Information Systems*, (Perth, Western Australia), pp. 288–292, December 1993.
- [4] L. Fu and T. Chen, “Sensitivity analysis for input vector in multilayer feedforward networks,” in *Proc. of IEEE International Conference on Neural Networks*, vol. 1, (San Francisco, California), pp. 215–218, March 1993.
- [5] N. S. Orzechowski, S. R. T. Kumara, and C. R. Das, “Performance of multilayer neural networks in binary-to-binary mappings under weight errors,” in *Proc. of IEEE International Conference on Neural Networks*, vol. 3, (San Francisco, California), pp. 1684–1689, March 1993.
- [6] J. D. Yuh and R. W. Newcomb, “A multilevel neural network for A/D conversions,” *IEEE Transactions on Neural Networks*, vol. 4 (3), pp. 470–483, May 1993.
- [7] J. M. Żurada, I. Cloete, and E. van der Poel, “Neural associative memories with multiple stable states,” in *Proc. of the 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, (Iizuka, Japan), pp. 45–51, August 1994.