TWO-DIMENSIONAL MATRIX NEURAL NETWORKS AS ALTERNATIVE TO MODERN STATE-OF-THE ART ANN ARCHITECTURES

Албасова А.І.
Науковий керівник – д.т.н., Бодянський Є.В.
Харківський національний університет радіоелектроніки
(61166, Харків, пр. Науки,14, каф. ШІ, тел. (057) 702-13-06)

This work is devoted to the study of some issues related to the use of traditional vector-based neural networks to process non-vectorial inputs such as matrices. As traditional ANNs assume vector inputs, non-vectorial information must be previously converted to vector form. This process causes some problems, e.g. lost of information and decrease in learning speed. The purpose of this work is to point out the limits of traditional ANNs in handling non-vectorial data and consider matrix neural networks, which takes matrices directly as inputs.

ANNs, especially deep networks, are widely used to solve different problems, connected to information processing, such as image understanding, natural language processing, and so on. Because of its usefulness and tremendous application potential, neural networks have gone far beyond the scientific community and found application in many domains. In recent years, with the dramatic increases in the power of computers, the attention of researchers is focused on the deep neural networks that have significantly improved approximation capabilities compared to traditional neural networks. However, the price of that is the complexity of architecture and dramatically increasing time of learning.

Evolving from the simplest perceptron [1] to the most sophisticated deep neural networks [2], the basic structure of the most widely used neural networks remains almost the same. The basic structure of an ANN consists of artificial neurons grouped into layers, arranged one after another with feed-forward information flow from the previous layer to the next layer.

Traditionally neurons in layer line in a column like elements in a vector. There is no restriction on how the neurons should be arranged spatially, but vector-like form offers certain advantages such as the ease of visualization and the convenience of deduction of mathematical formulations of informational flow. Along with this, the information for processing is, naturally, presented as a sequence of vector observations, and the output signal also has the form of a vector. Consequently, non-vector inputs such as images must be transformed into a vector. This transformation has some significant disadvantages. Firstly, this is an increasing in the number of tunable weights. Secondly, spatial information among data elements may be lost during vectorization. Third, it leads to a decrease in the speed of the learning process.

To overcome the above-mentioned problems, studies related to two-dimensional matrix neural networks have appeared. Their input and output signals have a matrix form. It is worth mentioning that convolutional neural networks work with images (matrices) directly. However, the main difference between CNNs and two-dimensional matrix neural networks is that the input image after convolution and pooling is needed to be presented as a vector that is further processed by the traditional vector-based neural network. While in 2D neural networks, matrices pass through each layer without vectorization at all.

Based on the adaptive matrix bilinear model that was introduced in [3], a two-dimensional matrix neural network can be presented in the form

$$\hat{Y}(k) = \{\hat{y}_{j_1 j_2}\} = A(k-1)X(k)B(k-1), j_1 = 1,2,\ldots,n_1; j_2 = 1,2,\ldots,n_2 \quad (1)$$

where $A(k-1)$, $B(k-1)$ denote $(n_1 \times n_1)$, $(n_2 \times n_2)$-matrices of adjustable parameters that may be tuned by using gradient adaptation procedure.

Of course, it is possible to rewrite the model (1) in traditional vector form

$$\begin{cases} \vec{Y}(k) = (B^T(A)\vec{X}(k) = C\vec{X}(k) \\ \hat{\vec{Y}}(k) = (B^T(k-1)(A(k-1))\vec{X}(k) = C(k-1)\vec{X}(k) \end{cases} \quad (2)$$

where $\vec{Y}(k)$, $\hat{\vec{Y}}(k)$, $\vec{X}(k)$ are $(n_1 n_2 \times 1)$-vectors, $\otimes$ – the symbol of the tensor product. However, it should be noted that the use of the representation (2) is inappropriate for the two reasons. Firstly, the dimension of the matrix $C(k-1)$ increases dramatically, because with $n_1 > 2$, $n_2 > 2$ it turns out that $(n_1 n_2)^2 > n_1^2 + n_2^2$. Second, on each iteration k there is a need to solve a system of bilinear equations

$$B^T(k-1)(A(k-1) = C(k-1) \quad (3)$$

which in the general case is uniquely unresolvable. It is important to note that $(n_1 n_2)^2$ is the number of matrix parameters $C(k-1)$ to be evaluated, $n_1^2 + n_2^2$ is the number of matrix parameters $A(k-1)$ and $B(k-1)$ to be evaluated.

Based on model (1), its non-linear modification can be introduced:

$$Y(k) = \{y(j_1 j_2)\} = (\circ(A(k-1)X(k)B(k-1)) = (\circ U(k) \quad (4)$$

where $\circ$ is a $(n_1 \times n_2)$-matrix of activation functions.

References

1. F. Rosenblatt, The Perceptron: A Perceiving and Recognizing Automaton, Cornell Aeronaut, 1957, Report 85-60-1.
2. Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature, vol.521, pp.436-444, 2015.
3. Ye.Bodyanskiy, I.Pliss, and V.A.Timofeev, Discrete adaptive identification and extrapolation of two dimensional fields, Pattern recognition and Image Analysis 1995, vol.5, no.3, pp.410-416.