Determination of Features of the Dependence of Neuron Net Output Coordinates on Other Coordinates and Parameters of Neuron Net

Vladimir Ptitchkin
Byelorussian State University of Informatics and Radioengineering

ABSTRACT

The paper presents the deduction of main relationships of back-propagation algorithm on the basis of approximation of the unknown dependence of the efficiency factor on neural net parameters by the linear part of Taylor series. Such an approach for deduction of known results is intended to relax the limits of application of the back-propagation algorithm and to eliminate some of its shortcomings on the basis of new methods of approximation of the before-mentioned dependence. As an example, this paper examines the possibility of application of back-propagation algorithm in the case of non-differentiable activation functions.

1: Introduction

The back-propagation algorithm is of great importance for the theory and practice of neural nets. It is the algorithm for learning of multilayer neural nets. It is based on the iterative procedure of finding of minimal mean-square error in the parameters space using the method of quickest descent [1].

In general, the problem of determination of new values of the adjustable parameters of neural net can be considered as the problem of the approximate expression of the dependence of the efficiency factor of the neural net on before-mentioned parameters, and also as a problem of application of some characteristics of this dependence for parameters correction. We shall understand the approximate determination of the dependence as one of the forms of its approximations, for example, a finite part of series; let's understand the characteristics of this dependence as the coefficients of series. From this point of view, the traditional back-propagation algorithm can be considered as the application of coefficients of approximate description of the before-mentioned dependence in the form of Taylor series linear part, because the coefficients of Taylor series are the parameter derivatives of the output coordinate.

We think that many shortcomings of the back-propagation algorithm are the results of application of activation function derivatives in the observation points. Some of these shortcomings can be eliminated by application of another approaches for the approximation of the unknown dependence of the efficiency factor on net parameters. For example, the application of statistical linearization method allows to deduct the linear approximation of dependence between coordinates, without taking into consideration the differentiability of activation functions [2]. In such conditions, it is naturally enough to return to the application of sign-functions as activation functions.

However, the attempt not to use the derivatives of neural net non-linear characteristics leads the loss of theoretical ground of back-propagation algorithm in the form of the method of quickest descent. For example, applying the method of statistical linearization for approximate description of the dependence of neural net output coordinates on any other coordinates, as original approach for neural nets analysis, we have not found any groung for development of this procedure for net learning [2]. From the heuristical point of view, the non-essential variations in the procedure of deduction of main relationships for back-propagation algorithm (in comparison with Taylor series approximation) must not lead to the impossibility to apply the linear approximations in such a manner as in back-propagation algorithm.

There are no doubts that the application of approximation-based considerations for deduction of main relationships for back-propagation algorithm (using Taylor series as a tool for their deduction) can give, as a result, only well-known relationships [1].

However, it's interesting to obtain these relations in just such a way. Such result will allow to generalize and adjust the above-mentioned procedure for specific conditions. This is the main goal of the paper proposed. The additional result is the obtaining of the simplified (from the engineering point of view) procedure of deducting of main relationships for back-propagation algorithm; this procedure, as we think, can be used for training specialists.

2: Matrix description of neural net

Every neuron, neural layer and neural net (NN) as a whole can be descripted by sets of states, input coordinates and output coordinates, which are represented by vectors S, U, and Y respectively. There function in a steady-state mode can be descripted by two equations: transition equation $S = A(U)$ and output equation $Y = F(S)$.

The formal neuron can be represented as a multi-input summator and non-linear converter, which are connected serially. Therefore, transition equation can be denoted as summator equation, and states can be denoted as summator outputs. Let's denote the above-introduced coordinates U, S, and Y as summator inputs, summator outputs, and non-linear elements outputs, respectively. The summator equations are linear for the single neuron, as well as for the neural layer.

The summator equations can be represented in matrix form: $S = B + AU$, where B is a column of free terms. The elements of this column are the threshold values of formal neurons, only with the inverse sign. The matrix A, in turn, contains the synaptic weights. The threshold values and the synaptic weights are the parameters of NN. There is one particular feature of conversions implemented by NN: the non-linear conversion of each state is independent from the conversions of any other state. It means that the dimensions of vectors Y and S are equal, and also, each coordinate of Y is functionally dependent on the respective coordinate of S. In another words, the vector equation $Y = F(S)$ can be considered as a complex of one-dimensional functions $y_i = f_i(s_i)$. For homogeneous NN, the activation functions for all neurons are the same, i.e. $f_i = f$. For simplicity, we shall consider only homogeneous NN with serial connections.

Let's suppose that all neurons are numbered in such a way that when the output of i-th neuron is calculated, all inputs of this neuron are already calculated. Such a condition determines the separation of neurons into layers. The collating of neurons in the net allows to collate net coordinates, and the separation of neurons into layers allows to represent the vectors in a form of blocks. Let's denote

$$Y^0 = (Y_0, Y)', Y = (Y_1, Y_2, ..., Y_m)',$$
$$S^0 = (S_0, S)', S = (S_1, S_2, ..., S_m)',$$

where Y_j, and S_j are the vector of input coordinates and the vector of states for j-th layer, respectively. Some uncertainty in separation of coordinates and neurons into layers is introduced by sensor elements. Let's consider them as neurons of zero layer. In NN, the inputs of any neuron are the outputs of some other neurons; therefore, the coordinates of vector U can be excluded from the set of equations representing the NN. Such an exclusion is especially simple after the separation of neurons into layers and collation of output coordinates into ranks. The vector of states for any layer of NN with serial connections can be determined through the vector of outputs of the preceding layer:

$$S_i = B_i + A_i Y_{i-1}, (i = 1, 2, ..., m). \quad (1)$$

If, as a result of the linearization, the connection between coordinates y_i and s_i is expressed as

$$y_i = q_i + k_i s_i, \quad (2)$$

then the set of such equations can be represented in a matrix form for NN, as well as for any particular layer:

$$Y_i = Q_i + K_i S_i, (i = 1, 2, ..., m), \quad (3)$$

where Q_i is a column of free terms, K_i is a diagonal matrix of linearization coefficients for activation functions.

In accordance with our goal, we shall apply the expansion into Taylor series for linearization. Assume that there is a signal $Y_0 = Y^*_0$ on the input of NN. Let's denote by asterisk (*) the values of all NN coordinates determined by this input signal. From (1), we shall obtain:

$$S^*_i = B_i + A_i Y^*_{i-1}, S_i - S^*_i = A_i (Y_{i-1} - Y^*_{i-1}). \quad (4)$$

Let's represent deviation equations in the following form:

$${}^0S_i = A_i {}^0Y_{i-1}, \quad (5)$$

where

$${}^0S_i = S_i - S^*_i, {}^0Y_{i-1} = Y_{i-1} - Y^*_{i-1}$$

Now we shall start the linearization of non-linear equation $y_i = f_i(s_i)$. Assuming that the functions f_i are differentiable everywhere in their domains, let's apply for them the expansion into Taylor series in the vicinity of $s_i = s^*_i$. Then, instead of the general expression (2), we can use more specific and convenient expression:

$$y_i - y^*_i = k^*_i (s_i - s^*_i) \text{ or } {}^0y_i = k^*_i {}^0s_i, \quad (6)$$

where

$${}^0y_i = y_i - y^*_i, \; {}^0s_i = s_i - s^*_i, \; k^*_i = f_i'(s^*_i),$$

i.e. $k^*_i = f_i'(s^*_i)$ is a derivative of i-th activation function in a "point" of the corresponding value of input coordinate. Therefore, the vector of output coordinates of i-th layer also can be expressed in matrix form:

$${}^0Y_i = K^*_i {}^0S_i, \quad (7)$$

where K^*_i is a diagonal matrix of linearization coefficients, i.e. the derivatives in the "point" under analysis.

From (5) and (7), we can obtain:

$${}^0Y_i = K^*_i A_i {}^0Y_{i-1} \; (i = 1, 2, ..., m). \quad (8)$$

This relationship is an approximate expression of dependence of i-th layer output vector on the output vector of preceding layer or (the same) the dependence on the input vector of the i-th layer. However, this approximate relationship allows to obtain the exact expression for the derivation of one coordinate with respect to another. This possibility is guaranteed by the application of just Taylor series for function approximation.

3: Determination of linearized relationships between net coordinates

It's not difficult to obtain the linearized expression of dependence of states or outputs vector of i-th layer on the states or outputs vector of (i-2)-th layer, etc. For this investigation, the output vector of the last layer is of special interest. For obtaining the derivative in the particular point of NN output coordinates (or, the same, in the particular point of output coordinate of last m-th layer) ${}^0Y_m$ with respect to the output coordinate of j-th layer ($0 \le j < m$) we need to apply the expression (8) k times, where $k = m-j$:

$${}^0Y_m = (K^*_m A_m)(K^*_{m-1} A_{m-1}) \ldots (K^*_{m-k} A_{m-k}) {}^0Y_{m-k-1}. \quad (9)$$

The parentheses in this expression are set for better readability only, because the operations of multiplication may be performed in any order. However, such a form of expression facilitates the correction of error in the case of missing some intermediate term.

Applying the expression (5) with $i = m-k$

$${}^0S_{m-k} = A_{m-k} {}^0Y_{m-k-1}$$

we can obtain the expression of linear dependence of NN output coordinate on the state of j-th layer:

$${}^0Y_m = (K^*_m A_m)(K^*_{m-1} A_{m-1}) \ldots (K^*_{m-k+1} A_{m-k+1}) K^*_{m-k} {}^0S_{m-k}. \quad (10)$$

From (5) and (7) we can obtain that

$${}^0S_i = A_i K^*_{i-1} {}^0S_{i-1}. \quad (11)$$

From here, we can deduct the following expressions:

$${}^0S_m = A_m(K^*_{m-1} A_{m-1}) \ldots (K^*_{m-k} A_{m-k}) {}^0Y_{m-k-1}, \quad (12)$$

$${}^0S_m = A_m(K^*_{m-1} A_{m-1}) \ldots (K^*_{m-k+1} A_{m-k+1}) K^*_{m-k} {}^0S_{m-k}. \quad (13)$$

Thus, we can obtain the derivative of any coordinate (output or state) for i-th layer with respect to any coordinate of the preceding layer. The derivatives are required for determination of efficiency factor gradient; therefore, it's necessary to determine the expressions of output vector derivatives with respect to inputs of summators of all layers. Thus, the most interesting result for this investigation is the expression (10).

If the determination of output coordinates derivatives was the only goal of NN analysis, the expression (10) would be the final result making the following discussions unnecessary. However, it seems unreasonable to calculate derivatives for every particular layer in order to determine the derivatives for all layers, because the same set of calculations must be repeated for each layer. In such conditions, it seems naturally enough to use the results of derivatives calculation for i-th layer for calculation of derivatives for (i-1)-th layer. Of course, such a method gives real benefit only when the number of layers and/or neurons is great enough. Possibly, under small number of layers and neurons, it is more reasonable to perform more calculations using simple algorithm, than to reduce the

number of arithmetic operations at the expense of complicating the algorithm. Even if we shall not take into consideration the difficulty of understanding the back-propagation algorithm in the recurrent form, just the possibility to eliminate the recurrence seems especially important for the following generalizations.

Thus, in conditions when the number of neurons is not very great, the recurrent implementation of back-propagation algorithm is not of great importance. However, we shall consider this question, first of all, in order to prove that the proposed scheme of investigation leads (from a particular point of view), as a final result, to the back-propagation algorithm.

4: The recurrent form of dependencies

Let's present the expression (10) for two specific cases ($k = 0$, $k = m-1$)

$$ {}^0Y_m = K^*_m {}^0S_m , \tag{14} $$

$$ {}^0Y_m = (K^*_m A_m)(K^*_{m-1}A_{m-1}) \dots (K^*_2 A_2)K^*_1 {}^0S_1 . \tag{15} $$

Let's present the NN output vector ${}^0Y_m$ as a function of state vector ${}^0S_j$ in matrix form:

$$ {}^0Y_m = \Delta_j {}^0S_j . \tag{16} $$

Thus, we can consider the matrix Δ_j as the matrix of output vector derivatives with respect to the states of i-th layer. It's obvious that

$$ \Delta_m = K^*_m . \tag{17} $$

From (10) under k=1, taking into consideration (16)

$$ {}^0Y_m = (K^*_m A_m)K^*_{m-1} {}^0S_{m-1} = \Delta_m A_m K^*_{m-1} {}^0S_{m-1} = $$
$$ = \Delta_{m-1} {}^0S_{m-1} $$

we can obtain:

$$ \Delta_{m-1} = \Delta_m A_m K^*_{m-1} . \tag{18} $$

In turn, under k=2, taking (15) and (18) into consideration

$$ {}^0Y_m = (K^*_m A_m)(K^*_{m-1}A_{m-1})K^*_{m-2} {}^0S_{m-2} = $$
$$ = \Delta_{m-1} A_{m-1} K^*_{m-2} {}^0S_{m-2} = \Delta_{m-2} {}^0S_{m-2} $$

we can obtain

$$ \Delta_{m-2} = \Delta_{m-1} A_{m-1} K^*_{m-2} . \tag{19} $$

Now we can deduce the generalized expression

$$ \Delta_{m-k-1} = \Delta_{m-k} A_{m-k} K^*_{m-k-1} , \tag{20} $$

which can be considered as recurrent equation with initial conditions (17).

This recurrent expression coincides with the main relationship of back-propagation algorithm (up to the designations). The deduction of this expression seems natural if we use matrix terms. In accordance with this algorithm, the calculations are performed in two "directions". The forward-direction calculations include calculation of all NN coordinates in the order of layer number increment, starting from the values of input coordinates. After the determination of output coordinates for the last layer, the reverse-direction calculations are performed, which include calculations of "generalized" errors for the last layer (i.e. the components of vector Δ_m, and vectors Δ_i for all preceding layers, in the order of layer number decrement.

It must be noted that if we use the direct expressions of linear equations of connection between NN coordinates, then the division of calculations into two stages (forward-direction and reverse-direction) will be unnatural. Indeed, the forward-direction calculation of NN coordinates, from layer to layer, consist in sequential calculation of coordinates (i.e. output and state) for each layer, starting from the first layer, i.e. in calculation of Y_0, S_1, Y_1, S_2, Y_2, ..., S_m, Y_m.

After the calculation of the column Y_1, it's possible to determine analytical expressions of dependence of output Y_i on all columns of states S_j under $j \le i$. It's obvious that under i=1, i.e. on the first step of the sequential processing of layers, we can obtain only one dependence:

$$ {}^0Y_1 = K^*_1 {}^0S_1 , \tag{21} $$

However, on he second step we can obtain two dependencies:

$$ \begin{aligned} {}^0Y_2 &= K^*_2 {}^0S_2 , \\ {}^0Y_2 &= K^*_2 A_2 K^*_1 {}^0S_1 , \end{aligned} \tag{22} $$

and on the third step there are three dependencies:

$$ \begin{aligned} {}^0Y_3 &= K^*_3 {}^0S_3 , \\ {}^0Y_3 &= K^*_3 A_3 K^*_2 {}^0S_2 , \\ {}^0Y_3 &= K^*_3 A_3 K^*_2 A_2 K^*_1 {}^0S_1 \end{aligned} \tag{23} $$

and so on.

To make obvious the recurrent nature of the proposed algorithm, let's introduce the designation Δ_{ij} for matrix of connections between vectors Y_i and S_j. Using this matrix, we can express the dependence of the first of the above-mentioned vectors from the second in the following form:

$$ {}^0Y_i = \Delta_{ij}\, {}^0S_j \quad (j \le i). \tag{24} $$

Using this designation, let's present (21) in the following form:

$$ {}^0Y_1 = \Delta_{11}\, {}^0S_1, \tag{25} $$

where $\Delta_{11} = K^*_1$; (22) can be expressed as follows:

$$ {}^0Y_2 = K^*_2\, {}^0S_2 = \Delta_{22}\, {}^0S_2, $$
$$ {}^0Y_2 = K^*_2 A_2 K^*_1\, {}^0S_1 = K^*_2 A_2 \Delta_{11}\, {}^0S_1 = \Delta_{21}\, {}^0S_1, \tag{26} $$

where $\Delta_{22} = K^*_2$, $\Delta_{21} = K^*_2 A_2 \Delta_{11}$; and (23) can be expressed in the following form:

$$ {}^0Y_3 = K^*_3\, {}^0S_3 = \Delta_{33}\, {}^0S_3, $$
$$ {}^0Y_3 = K^*_3 A_3 K^*_2\, {}^0S_2 = K^*_3 A_3 \Delta_{22}\, {}^0S_2 = \Delta_{32}\, {}^0S_2, $$
$$ {}^0Y_3 = K^*_3 A_3 K^*_2 A_2 K^*_1\, {}^0S_1 = K^*_3 A_3 \Delta_{21}\, {}^0S_1 = \Delta_{31}\, {}^0S_1, $$

where

$$ \Delta_{33} = K^*_3, \quad \Delta_{32} = K^*_3 A_3 \Delta_{22}, \quad \Delta_{31} = K^*_3 A_3 \Delta_{21}. $$

In general, we can conclude that after processing of i-th layer ($i \ge 1$) we have i matrices Δ_{ij} ($1 \le j \le i$) calculated. For next, (i+1)th layer, we can calculate i+1 matrices in accordance with the following rule:

$$ \Delta_{(i+1)j} = K^*_{(i+1)} A_{(i+1)}\, \Delta_{ij}, \ (1 \le j \le i) $$
$$ \Delta_{(i+1)(i+1)} = K^*_{(i+1)}. $$

After processing of the last (m-th) layer we have m matrices Δ_{mj} ($1 \le j \le m$) calculated. They are used as the above-mentioned matrices Δ_j. Indeed, the meaning of these matrices is determined by connection equation (16); after its comparison with (24), we can see that $\Delta_{mj} = \Delta_j$.

Thus, we can conclude that there are at least three methods that can be applied for calculation of the linearized dependence of NN output coordinate on the states of each layer ("in the vicinity of equilibrium position"). The first method consists of direct calculation of each of the dependencies in the matrix form. The second method includes two stages: 1) the calculation of "equilibrium position" for all coordinates of each layer (the forward-direction calculations) and 2) the calculation of dependence of the last layer output coordinates (i.e. the output coordinates of NN) on the states of preceding layers (the reverse-direction calculation). In the third method, it is proposed to determine the relationship for output coordinate in parallel with the calculation of "equilibrium position", i.e. the values of NN coordinates determined by the value of the input coordinates.

5: Determination of output coordinates derivatives with respect to parameters

In the back-propagation algorithm, it's necessary to calculate the derivatives of output coordinates with respect to parameters (not with respect to the states of layers). While the relation between these derivatives is rather simple, let's present some relationships between them. First, the matrix Δ_j between the centralized values ${}^0Y_m = \Delta_j\, {}^0S_j$ can be considered as the matrix of derivatives of non-linear dependence of Y_m on S_j with $S_j = S^*_j$. Such an assertion is grounded on application of Taylor series for approximation of all non-linear functions used in this conversion. Each element of matrix $\Delta_j = (\delta_{j\,pq})$ is the derivative of p-th component of NN output vector (m-th layer) with respect to q-th component of states vector of j-th neuron layer $s_{j\,q}$.

$$ \delta_{j\,pq} = \partial y_{m,\,p} / \partial s_{j,\,q}. $$

If given the matrix expression of dependence of j-th states column S_j on output vector of the preceding layer Y_{j-1} (see (1))

$$ S_j = B_j + A_j Y_{j-1} $$

we take only the q-th line

$$ s_{j,\,q} = b_{j,\,q} + a_{j,\,q,\,1} y_{j-1,\,q,\,1} + a_{j,\,q,\,2} y_{j-1,\,q,\,2} + \dots, $$

where $b_{j\,q} = -\theta_{j\,q}$, then it is obvious that

$$ \partial y_{m,\,p} / \partial b_{j,\,q} = \partial y_{m,\,p} / \partial s_{j,\,q} = \delta_{j\,pq}, $$

$$ \partial y_{m,\,p} / \partial a_{j,\,q,\,k} = \partial y_{m,\,p} / \partial s_{j,\,q} \times \partial s_{j,\,q} / \partial a_{j,\,q,\,k} = $$
$$ = \delta_{j\,pq} \times y_{j-1,\,q,\,k}. \tag{27} $$

Thus, after determination of the derivatives of output column with respect to the states of all NN layers, we can determine the derivatives of output coordinates with respect to all NN parameters. Of course, for coordinate values we take their values in the experiment

under consideration. In turn, on the basis of output coordinate derivative with respect to the parameters, we can determine the derivatives of the square error with respect to the same parameters. In fact, this is the end of the deduction of main relationships of back-propagation algorithm.

It's not difficult to see that the relationships obtained coincide (up to the designations) with the well-known derivative expressions for back-propagation algorithm. The main difference is not in final results but in the way by which these results are obtained. In classical presentation of the back-propagation algorithm, the problem under consideration is the determination of derivatives of composite function (i.e. mean-square error) as such, without any connection with the problem of transfer function approximation by power polynomials. In such conditions, the application of chain rule and recurrent-type calculations for determination of composite functions derivatives seems natural and even necessary. However, the application of derivatives in any case implies the approximation of functions in the vicinity of some point. In the proposed approach, the starting point for finding better parameters values (in comparison with the values used in the experiments) is the approximation of the dependence of efficiency factor on parameters in the range of variables values in the training set. We think that in such conditions the application of derivatives is not a matter of principle, but providing they are used, the proposed algorithm gives the same results as the classical approach for back-propagation algorithm. This was the main goal of the investigation proposed, and we think that this goal is achieved.

6: Determination of linearized relationships without application of derivatives

As a rule, the practical implementation of back-propagation algorithm includes some violations of the assumptions used for proving of algorithm convergency. These violations include not only the selection of the final step, but also the correction of parameter values after the presentation of each pattern (the pulse method). At the same time, all considerations are valid only provided that the values of parameters remain the same during the presentation of all the patterns of the training set. Although the pulse method gives some benefits in many cases, it requires additional investigations. To simplify the following discussions with respect to the possibility of NN learning without the determination of derivatives, let's assume that the new values of parameters for back-propagation algorithm are implemented only after the end of processing of the training set.

In such conditions, the determination of "derivatives" of efficiency factor with respect to the NN parameters without the differentiation of activation functions can be considered as the determination of coefficients of linear regression of efficiency factor on the outputs of summators. These coefficient can be determined, for example, using the method of least squares. These coefficients may be considered as the "generalizer errors" of back-propagation algorithm. However, the calculation of the derivatives of efficiency factor with respect to the parameters in accordance with (27) is strictly grounded on the basis of differential calculus; at the same time, the application of (27) in the case under consideration requires strict grounds. As a heuristical recommendation, it may be proposed to apply the mean value of output coordinate instead of the value taken from the particular experiment.

To investigate the acceptability of the proposed recommendation for practical purposes, the numerical experiments, as a minimum, are necessary. As a test example for such experiments, we can use two NNs, the only difference between which is the activation function of formal neurons. For one of these NNs, the sign-function must be used as the activation function. For another NN, the sigmoid-function must be used with such a great value of the parameter that makes this function practically indistinguishable from the sign-function. It's obvious that, providing the beforementioned heuristical recommendation is valid, the values of gradient of the efficiency factor for such NNs must be approximately the same. Such numerical experiment was carried out for one test example, and the results were satisfactory. Of course, the single example is not adequate ground to recommend the method for wide practical application; however, we think that the main goal of this investigation is achieved. This goal was to demonstrate the opportunities of back-propagation algorithm, without respect to its particular implementation.

References

[1] Rumelhart D.E., Hinton G.E., Williams R.J. Learning internal representations by error propagation. In Parallel distributed processing, vol. I, Cambridge, MA: MIT Press, 1986.

[2] Птичкин В.А. Анализ нейросетевых преобразований линеаризационными методами. // Нейрокомпьютер, М., 1999