ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 43-48

Input Parameters' Estimation via Neural Networks

Igor V. Tetko, Alexander I. Luik

Biomedical Department, Institute of Bioorganic & Petroleum Chemistry, Murmanskaya, 1, Kiev-94, 253660, Ukraine, tetko@bioorganic.kiev.ua

Abstract

We propose simple heuristic methods that can evaluate the relevance of input parameters after completing of neural networks training. Besides that, these methods allow correct computation of inputs' contribution to each problem, when learning multiple tasks simultaneously. The estimations are done on statistical base and are independent of learning procedures and cost functions. Our simulation on three different tasks shows that these approaches are effective.

I. Introduction and Background

When learning an artificial neural network the researcher is often faced with the problem of understanding the contribution of each parameter to the final relationship between inputs and outputs. Using only the most important parameters theoretically results in better generalization by all pattern recognition methods.

Hypothetically, the search of the best subset of parameters can be done by training the net under all possible subsets of the set of parameters. However, this exhaustive search is computationally unfeasible, unless dealing with a very small net and few training patterns. The problem is even more hard if we recall the difficulties with local minimum, initial random weights' initialization. The described here a simple procedure that can be useful for solving of such problems.

The estimation of input parameters' importance is in close relation to the net pruning. Several methods that were proposed for neural networks (NN) pruning can be related to the problem of input parameter estimation. By Siestma and Dow [1], for example, we should remove at the end of the training each unit from the input set and investigate if the network can retrain itself. This procedure must be applied dynamically, during the work of one network. Such pruning of input parameters can fall due to local minimum or insufficient number of neurons in hidden layers or we can delete by mistake the most relevant parameter. More preferable approaches, by our opinion, are those that can evaluate the performance of input parameters after completing NN learning. The least pertinent parameters should be deleted and NN be re-learned on diminished parameters' set. This concept was suggested by Mozer and Smolensky [2] who have introduced the idea of estimating the sensitivity of the error function due to eliminating of each unit.

Before brief describing the method of Mozer and Smolensky we'll introduce some notation. Let us consider neural network as in standard back-propagation. Unit i on layer s connected to unit j on layer $s+1$ via a synapse of strength w_{ij}, and the *jth* unit computes:

$$a_j^{s+1} = f(\sum_i w_{ij}^s \cdot a_i^s) \qquad (1)$$

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 43-48

The activation function $f(\cdot)$ most frequently used is the logistic function

$$f(x) = \frac{1}{1+\exp(-x)} \tag{2}$$

or its symmetric version $$f(x) = \tanh(x) \tag{3}$$

The global error of the net is defined as $$E = \sum_p \sum_i (a_i^n - t_i)^2 \tag{4}$$

where the inner summation is over all neurons that are considered as output units of the net, t_i is the desired output upon presentation of pattern p and the outer sum is over patterns of the training set.

In terms of connection elimination S_{ij}, the sensitivity with respect to w_{ij} will be defined as

$$S_{ij} = E(w_{ij} = 0) - E(w_{ij} = w_{ij}^f)$$

Here w_{ij}^f are final values of weights. Mozer and Smolensky estimate S_{ij} by

$$\hat{S}_{ij} = -\left.\frac{\partial E}{\partial w_{ij}}\right|_{w_{ij}^f} \cdot w_{ij}^f$$

They pointed, however, at the problem arising especially from gradient descent algorithm - partial derivatives tend to zero when error decreases. Authors suggested changing the cost function into

$$E = \sum_p \sum_i \left| a_i^n - t_i \right| \tag{5}$$

and described the usefulness of their estimation procedure on some examples. E. D. Karnin [3] described a simple "shadow procedure" for weight sensitivity estimation that overperformed Mozer and Smolensky method when the quadratic error function was used. He approximated the equation by summation, taken over all the discrete steps that the network passes while learning (for details see [3]):

$$S_{ij} \approx -\sum_1^N \frac{\partial E}{\partial w_{ij}}(n)\Delta w_{ij}(n)\frac{w_{ij}^f}{w_{ij}^f - w_{ij}^i} \tag{6}$$

where N is the number of training epochs, w_{ij}^i initial value of weight. The application of method was successfully demonstrated on a number of problems.

We proposed here a new simple approach based on some heuristic assumption about NN operation.

II. The estimation methods

Let us consider the possible states of neuron j on layer s. Unit j can work in different modes depending from the sum: $\sum_i w_{ij} \cdot a_i^s$

1) if the sum is great (positive or negative) a unit works near saturation;
2) if the sum is small it works in a linear mode (see Figure 1).

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 43-48

Both modes are important for net functioning. The neurons working in 1*st* mode primarily have larger inputs than those working in 2*nd* mode. The relative strength s_{ij} between i and j neuron is in dependence from the $s_{ij} = \dfrac{\left\| w_{ij} \cdot a_i^s \right\|}{\max\limits_k \left\| w_{kj} \cdot a_k^s \right\|}$. The neuron having the greatest $s_{ij}=1$ exerts the most significant influence on neuron j in next layer and vice versa. Hear $\max\limits_k$ is taken over all weights ending at neuron j, $\|\cdot\|$ is used norm taken over all pattern $p=1,...,N$ of the training set. We used square of Euclidean norm, but other norms should be investigated. The sensitivity of a unit i due to outgoing weights can be introduced as:

$$S_i^s = \sum_j (s_{ij}^s)^2 \cdot S_j^{s+1} = \sum_j \frac{\left\| w_{ij}^s \cdot a_i^s \right\|}{\max\limits_k \left\| w_{kj}^s \cdot a_k^s \right\|} S_j^{s+1} = \sum_j \frac{(w_{ij}^s)^2 \cdot \sum\limits_p \left| a_i^s \right|^2}{\max\limits_k \left| (w_{kj}^s)^2 \cdot \sum\limits_p \left| a_k^s \right|^2 \right|} S_j^{s+1} \quad (7)$$

This is a recurrent formula calculating the sensitivity of a unit on layer s via the neuron sensitivities on layer $s+1$. The sensitivities of output neurons are set to 1. All sensitivities in a layer are usually normalized to a maximum value of 1. If we set only one neuron on the output layer to 1 and others to 0, we'll obtain on inputs the relevant significance of parameters to the considered pattern. If we set all output neurons to 1, we'll calculate the relevance of each input to the whole task. Here, no assumptions were made about the forms of error function or learning procedure.

The equation (7) can be applied when both finite and infinitive numbers of training examples are used. In first case we should run all samples from training set and in second case estimate $\sum\limits_p \left| a_i^s \right|^2$ statistically, using reasonable number of examples. However, we can simplify the equation (7) in assumption that $\sum\limits_p \left| a_i^s \right|^2$ is practically the same for all neuron in layer s: $\sum\limits_p \left| a_i^s \right|^2 \cong \sum\limits_p \left| a_j^s \right|^2$ (8)

We'll obtain $$S_i^s = \sum_j (s_{ij}^s)^2 \cdot S_j^{s+1} \approx \sum_j \frac{(w_{ij}^s)^2}{\max\limits_k \left| (w_{kj}^s)^2 \right|} S_j^{s+1} \quad (9)$$

Both forms of equation were used to evaluate the performance of input parameters, as demonstrated in next section.

III. Examples

We have initialized the networks with small random weights in a range [-1,1] , then applied back-propagation (with momentum) for supervised learning of a training sequence. The outputs of each node, as well as input to the net, were in the

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 43-48

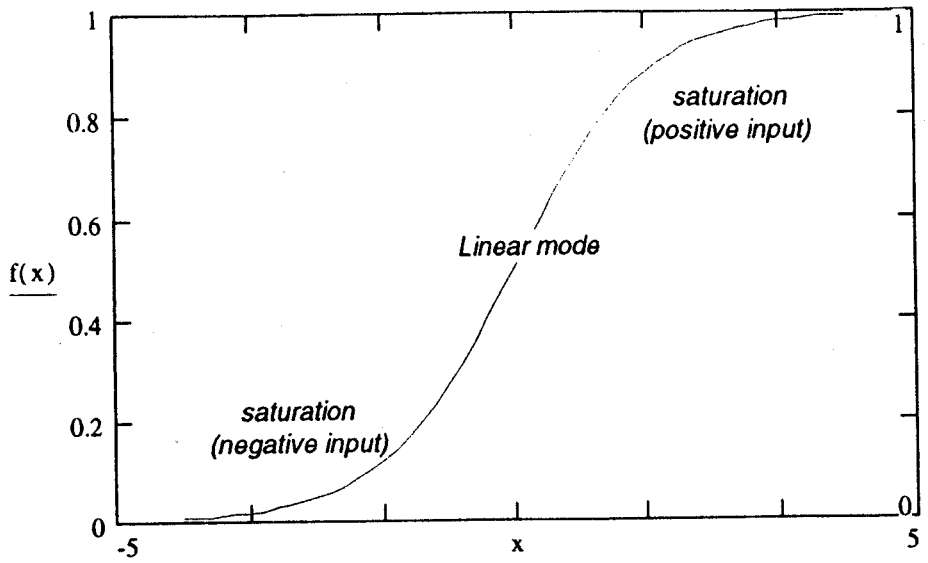

Figure 1. Modes of an activation function $f(x)=\frac{1}{1+\exp(-x)}$ *in dependence of* $\sum_p \left\langle a_i^s \right\rangle^2$ *input from lower layer nodes.*

range [0,1] and logistic activation function (2) was used. We made 20 attempts in order to obtain statistically significant results.

Example 1

This example is adopted directly from [2] and [3]. Mozer and Smolensky termed it "the rule-plus exception problem" and the same designation was used by E. D. Karnin. The method of [2] fails to find correct sensitivities when used in conjunction with quadratic error function (4), while being successful with linear cost function (5). E. D. Karnin demonstrated successful application of its sensitivity estimation by (6) with quadratic function. We will demonstrate that our method shows the result similar to [3].

The problem is to learn the Boolean function $AB+\bar{A}\bar{B}\bar{C}\bar{D}$. Clearly, the "rule" ($AB$) is more important than the "exception" ($\bar{A}\bar{B}\bar{C}\bar{D}$) accounting only 1 decision (out of 16). Four inputs, two hidden, and one output network (4-2-1 net) was trained to learn the function. The same net was used by E. D. Karnin (except he used activation function (3) and bias neurons). The criterion to complete training was $\Delta E<0.1$ (learning rate =0.1 and momentum =0.8). All first layer sensitivities had been calculated by both methods (7) and (9) were ranged from 1 (biggest) to 4 according to their magnitude (here and below sensitivities calculated by (7) are shown outside the parentheses and by (9) inside):

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 43-48

Input node	The ranks of calculated sensitivities			
	1	2	3	4
A	10(11)	9(8)	1(1)	0(0)
B	10(9)	10(10)	0(1)	0(0)
C	0(0)	1(1)	12(10)	7(9)
D	0(0)	0(1)	7(8)	13(11)

The sensitivities of input neurons C & D calculated by both methods were statistically less (preliminary rank 3,4) in comparison with those of neurons A & B (preliminary rank 1,2).

Example 2.

We made example 1 more difficult and used the network (4-4-2) to learn simultaneously "the rule-plus-exception" problem and parity function $p(x)=p(B+C+D)$: $p(x)=\begin{cases}1, & if\ x-odd\\ 0, & if\ x-even\end{cases}$. The first output neuron corresponds to Boolean function $AB+\bar{A}\,\bar{B}\,\bar{C}\,\bar{D}$ and the other to $p(x)$ accordingly. The same as in Example 1 learning and ranging procedures were done. The input parameters' contribution to each function was calculated as mentioned above: we set to 1 the output node corresponding to analyzed function while others to 0. The number of attempts when input nodes were recognized as the most important - rank 1,2 for first or rank 1,2,3 for second function were:

Input node	Number of attempts when node was recognized as the most important	
	$AB+\bar{A}\,\bar{B}\,\bar{C}\,\bar{D}$	$p(B+C+D)$
A	18(18)	0(1)
B	20(20)	20(20)
C	0(1)	20(19)
D	2(1)	20(20)

Statistical analysis by majority shows us redundancy of C and D neurons for function $AB+\bar{A}\,\bar{B}\,\bar{C}\,\bar{D}$ and A neurons for parity function.

Example 3

Another example is from the field of continuous functions' approximation. We tried to fit a network to data originating from the functions

$$f_1(x)=a\cdot sin(2\pi x)+b,\quad a=0.25,\quad b=0.6$$
$$f_2(x)=a\cdot x^2+b,\quad a=0.8,\quad b=0.1$$

Two input data sets were used:

1*st* $x_1[i]$, $i=1,..,N$; $N=50$ values randomly distributed on [0.1,0.9] and **2*nd*** the same as **1*st*** but with a noise distributed by normal law. On the output nodes we also used noised by normal distribution data $y_j[i]=f_j(x_j[i])+noise$, $j=1,2$.

Cross sums of square error $\Omega_{jk}=\Sigma(f_j(x_k[i])-y_j[i])^2$ were:

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 43-48

Input node	The ranks of calculated sensitivities			
	1	2	3	4
A	10(11)	9(8)	1(1)	0(0)
B	10(9)	10(10)	0(1)	0(0)
C	0(0)	1(1)	12(10)	7(9)
D	0(0)	0(1)	7(8)	13(11)

The sensitivities of input neurons C & D calculated by both methods were statistically less (preliminary rank 3,4) in comparison with those of neurons A & B (preliminary rank 1,2).

Example 2.

We made example 1 more difficult and used the network (4-4-2) to learn simultaneously "the rule-plus-exception" problem and parity function $p(x)=p(B+C+D)$: $p(x)=\begin{cases}1, & if\ x-odd\\ 0, & if\ x-even\end{cases}$. The first output neuron corresponds to Boolean function $AB+\bar{A}\,\bar{B}\,\bar{C}\,\bar{D}$ and the other to $p(x)$ accordingly. The same as in Example 1 learning and ranging procedures were done. The input parameters' contribution to each function was calculated as mentioned above: we set to 1 the output node corresponding to analyzed function while others to 0. The number of attempts when input nodes were recognized as the most important - rank 1,2 for first or rank 1,2,3 for second function were:

Input node	Number of attempts when node was recognized as the most important	
	$AB+\bar{A}\,\bar{B}\,\bar{C}\,\bar{D}$	$p(B+C+D)$
A	18(18)	0(1)
B	20(20)	20(20)
C	0(1)	20(19)
D	2(1)	20(20)

Statistical analysis by majority shows us redundancy of C and D neurons for function $AB+\bar{A}\,\bar{B}\,\bar{C}\,\bar{D}$ and A neurons for parity function.

Example 3

Another example is from the field of continuous functions' approximation. We tried to fit a network to data originating from the functions

$$f_1(x)=a\cdot sin(2\pi x)+b,\quad a=0.25,\quad b=0.6$$
$$f_2(x)=a\cdot x^2+b,\qquad a=0.8,\quad b=0.1$$

Two input data sets were used:

1*st* $x_1[i]$, $i=1,..,N$; $N=50$ values randomly distributed on [0.1,0.9] and **2***nd* the same as **1***st* but with a noise distributed by normal law. On the output nodes we also used noised by normal distribution data $y_j[i]=f_j(x_j[i])+noise$, $j=1,2$.

Cross sums of square error $\Omega_{jk}=\Sigma(f_j(x_k[i])-y_j[i])^2$ were: