On theNormalised Input Sensitivities in Neural Networks

Ion Ciuca and Elena Jitaru
Research Institute for Informatics
8-10 Averescu Avenue,
71 316 Bucharest
ROMANIA

Abstract: The neural networks applications in different areas take long and sometimes costly expertise as for example in medicine, etc, given that some features seem to bear no relevance to outputs. This paper considers the possibility of reducing the input space size by discarding irrelevant features in input vectors based on the input sensitivity. To get the computing of the respective sensitivities free of the output determination, the paper introduces normalised input sensitivities in relation with multilayer feedforward (MLP) neural networks.

1. Introduction

The neurocomputing paradigm has as support neural networks. They are capable of inductively constructing an inner representation model of the input-output function. This function is yielded by the input data viewed as a set of training examples. A neural network is a novel computational paradigm which resembles a Von-Neumann or Turing machine a lot.

With all different acceptances attached to them, "neural networks" are nevertheless mainly referred as : descriptive, computational and normative, with the first type being characteristic to emulating low-level operations in the human brain.

Unlike expert systems of which nature is a deductive one, neural networks learn to perform their designated tasks by examples, using inductive learning algorithms. Neural networks acquire knowledge from empirical databases to be used as "training sets" in a non-linear environment.

Many neural network architectures (the so - called mapping neural networks) can approximate a continuous input-output function with a finite number of variables. This procedure is known as universal approximation. Moreover, such architectures can operate generalisations, that is to correctly react to inputs not met with in the training process.

Many applications, as for example in medicine, when a syndrome needs be diagnosed, both traditional methods and the neural network paradigm ask, in general, for expensive laboratory investigations and tests. As a matter of fact just a few analyses and tests prove to be relevant. Using neural networks make it possible to select the symptoms which are more relevant in analysing the input sensitivity. This paper discusses normalised input sensitivities thereby troublesome output can be prevented, and describes an algorithm for discarding irrelevant features .

2. Normalised Input Sensitivities

In the following, we will consider recent methods for quantifying the input sensitivity [1],[2]. Such methods also give some hints at the internal neural network operation.

Focus will be on the following two methods:

- Jacobian sensitivity matrix[2];
- Logarithmic sensitivity matrix[2];

These methods will be analysed on the Multilayer Feedforward Perceptron(MLP) architectures with one hidden layer and with two hidden layers.

2.1. Jacobian Sensitivity Matrix

Let a full connected MLP neural network with a single hidden layer be as in Figure 1, where $h_j = g(\sum_i w_{ji} x_i)$, $y_k = g(\sum_j W_{kj} h_j$ and g() is a sigmoidal function and where - for clarity - there are drawn only some of the elements.

It is well-known that the input sensitivity is defined by a Jacobian matrix of outputs, with respect to input vector components, that is:

$$S_{ki} = J_{ki} = \frac{\partial y_k}{\partial x_i} \tag{1}$$

Using an asymmetric sigmoidal function, $g(u) = \frac{1}{1+e^{-u}}$ with derivative $\frac{dg}{du} = g(1-g)$ we get:

$$S_{ki}=J_{ki} = y_k(1- y_k)\sum_j W_{kj}\, w_{ji}\, h_j(1 - h_j) \qquad (2)$$

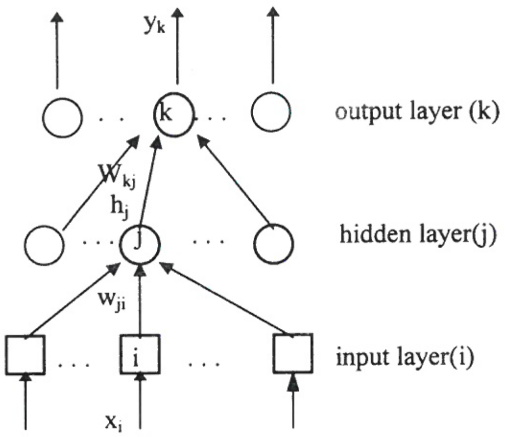

Figure 1. MLP Neural Network with A Single Hidden Layer

Lisboa et al [2] have arbitrarily given up the $y_k(1- y_k)$ factors from this expression so that some troubles met with in trying to implement this sensitivity should be shot. These are saturation factors in the sensitivity expression. It would be more natural to try to consider the following definition of the normalised sensitivity :

$$\sigma_{ki} = \frac{|J_{ki}|}{\sqrt{\sum_l J_{kl}^2}} = \frac{|\overline{S}_{ki}|}{\sqrt{\sum_l \overline{S}_{kl}^2}} \qquad (3)$$

where $$\overline{S}_{ki} = \sum_j W_{kj}\, w_{ji}\, h_j(1 - h_j) \qquad (4)$$

Now, it is obvious that σ_{ki} does not depend any more on the $y_k(1- y_k)$ factors.

Expressions (3) and (4) must be considered for every pattern p but for sparing minute notation the index p has been omitted.

2.2. Logarithmic Sensitivity Matrix

Such cumbersome factors as $y_k(1- y_k)$ should have been avoided , and so Lisboa et al [2] defined the logarithmic sensitivity with respect to input with the following equation:

$$S_{ki} = \frac{\partial \ln(|T_k - y_k|)}{\partial \ln x_i} \qquad (5)$$

where T_k = target output k component

Using the same MLP neural networks, as in Figure 1, S_{ki} becomes

$$S_{ki} = \frac{y_k(1-y_k)}{T_k - y_k}\, x_i \sum_j W_{kj}\, w_{ji}\, h_j(1 - h_j) \qquad (6)$$

To leave aside the output factor $y_k(1-y_k)$ we also define the normalised logarithmic sensitivity by means of the following relation

$$\sigma_{ki} = \frac{|S_{ki}|}{\sqrt{\sum_l S_{kl}^2}} \qquad (7)$$

and use the same definition for $\overline{S}_{ki}$ as we did for Jacobian Sensitivity Matrix, and thus Equation (7) becomes

$$\sigma_{ki} = \frac{|x_i \overline{S}_{ki}|}{\sqrt{\sum_l x_l^2 \overline{S}_{kl}^2}} \qquad (8)$$

Although the MLP neural networks with one hidden layer are universal approximators, let us also present the input sensitivities for MLP neural networks with two hidden layers as in Figure 2 where $h_j = g(\sum_i w_{ji} x_i)$, $H_k = g(\sum_j W_{kj} h_j)$, $y_l = g(\overline{W}_{lk} H_k)$ with g() a sigmoidal function.

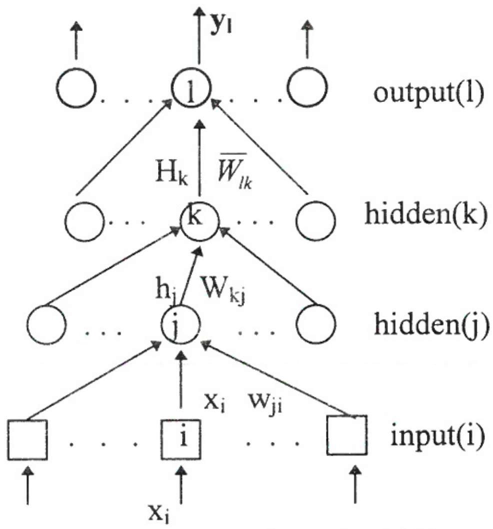

Figure 2. MLP Neural Network with Two Hidden Layers (The Elements occurring in Expressions Are Only Specified)

For an asymmetric sigmoidal function the normalised Jacobian sensitivity has the expression:

$$\sigma_{li} = \frac{|J_{li}|}{\sqrt{\sum_k J_{lk}^2}} = \frac{|\bar{S}_{li}|}{\sqrt{\sum_k \bar{S}_{lk}^2}} \quad (9)$$

now noted as:

$$\bar{S}_{li} = \sum_j \sum_k H_k(1-H_k)h_j(1-h_j)\bar{\bar{W}}_{lk}W_{kj}w_{ji} \quad (10)$$

The normalised logarithmic sensitivity is given by the expression

$$\sigma_{li} = \frac{|S_{li}|}{\sqrt{\sum_k S_{lk}^2}} = \frac{|x_i \bar{S}_{li}|}{\sqrt{\sum_k x_k^2 \bar{S}_{lk}^2}} \quad (11)$$

3. An Algorithm for Irrelevant Features Discarding

The algorithm presupposes that only features that yield a normalised sensitivity higher than a threshold for at least one pattern will be retained, and all the others will be discarded from all the patterns, thus enabling the definition of the algorithm as an irrelevant feature discarding algorithm.

At the end of this algorithm, a testing phase is going to be executed with the training set but using zero values in place of the removed features. If computed outputs turn to be the same as the desired ones, that means that the feature discarding algorithm has used a correct threshold; otherwise the threshold must be reduced and the algorithm be reexecuted.

After having found all the irrelevant features, and after discarding them, the neural network is ready for testing (production) phase.

The algorithm is the following:

Let MAXS be a vector initialised to zero in all its components. This vector will always contain the maximum values of normalised input sensitivity induced by every feature in all the output components over all training set of patterns p, that is:

MAXS[i]= $\max_{p,k}\{\sigma_{ki}^{p}\}$ where p=pattern index and k=output component

The **algorithm** runs as follows:

Step 1. Propagate a training pattern p from input to output.

Step 2. Compute normalised input sensitivity for every output component k with respect to every feature i of the propagated pattern p, and update one component after another of the MAXS vector . An entry of MAXS will each time contain the maximum values of normalised sensitivity of the feature associated with it, a value considered over all the output components k and all the patterns so far propagated .

Step 3. Go to step 1 for next input training pattern p.

Step 4. Get a threshold as a percentage of the maximum value over all the entries in the MAXS vector.

Step 5. After having propagated all input patterns of the training set, the MAXS vector gets examined element by element. The element with a value smaller than the given threshold will be viewed as a potential irrelevant feature to discard from all patterns of both training and testing sets.

Step 6. Once all potential irrelevant features known, a test phase of the training set takes place. But this time zeros will be substitued for all potential irrelevant features. If the neural network produces the desired outputs (in a tolerance) for every pattern, then potential irrelevant features are indeed irrelevant features and go to next step 7, else reduce the threshold, re-consider all features as relevant and go to step 5.

Step 7. Now , the network is ready to be used with the testing set (production phase) having the size reduced by the number of removed features.

4. A Medical Case

At present, for the neural network integrated environment developed under WINDOWS platforms, the normalised input sensitivities have already been implemented as an irrelevant feature algorithm for both Jacobian and Logarithmic sensitivities.

The algorithm was applied in acute intoxication diagnosis when 28-3-5 and a 28-3-2-5 MLP neural network trained with all usual training algorithms was used.

Table 1 presents the data used for training and testing sets. Rows are input patterns and columns are features.

Table 1. Acute Intoxication Data

Input patterns — class

x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	c
1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	2
1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	2
1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	2
0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	1	2
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	3
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	3
1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	3
1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1	0	0	3
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	1	0	4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	4
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	4
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	
0	1	1	1	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	
1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1	0	1	0	1	0	1	0	1	
0	0	0	1	1	1	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	1	1	1	0	

where:

x0=Headache
x1=Muscular weakness
x2=Dizziness
x3=Narcosis
x4=Vertigo
x5=Muscular pains
x6=Convulsions
x7=Hypothermia
x8=Memory troubles
x9=Cough
x10=Delirium
x11=Gas eye
x12=drowsiness
x13=nausea
x14=Vomiting
x15=Diarrhea
x16=Abdominal colic
x17=Hematuria
x18=Pulmonary edema
x19=Hyperthermia
x20=Irritation of eyes
x21=Chest pain
x22=tightness of chest
x23=chills
x24=Dyspnea
x25=cyanosis
x26=Jaundice
x27=Hemorrhage

c = class of intoxication (y0,y1,y2,y3,y4)
= 0 Acute carbon monoxide intoxication
= 1 Acute sulphate hydrogen intoxication
= 2 Acute zinc/copper intoxication by inhalation
= 3 Acute nitrate intoxication
= 4 Other acute intoxications

The data with specified output target class were used as a training set whereas the data without specified target output were used as a testing set.

After applying the irrelevant feature discarding algorithm, the input space size was reduced from 28 to 26 in case of Jacobian sensitivity - for a threshold equal to 0.07 - and from 28 to 19 in case of Logarithmic sensitivity - for a threshold equal to 0.393 for both neural networks with one and with two hidden layers.

Logarithmic sensitivity presented 9 features as removable: x2,x6,x7,x8,x9,x21,x22,x23 and x26.

Jacobian sensitivity presented 2 features as removable: x8 and x9.

5. Conclusion

The paper introduces the **normalised input sensitivities** and presents an irrelevant features. discarding algorithm. Input sensitivities concern MLP neural network.

The algorithm has been applied after training the neural network with the initial training set. After having discarded the irrelevant features, the neural network was ready for the next testing phase (production phase), with patterns exempted from irrelevant features.

The algorithm has been successfully tested in a medical application, namely for acute intoxication data classification when 28-3-5 and 28-3-2-5 MLP neural networks trained with all usual training algorithms are used.

The feature discarding algorithm let the input space size be reduced from 28 to 26 - for Jacobian sensitivity matrix - and from 28 to 19 - for logarithmic sensitivity. As the input feature x_i expression explicitly refers the input sensitivity, Logarithmic sensitivity is better ranked than Jacobian sensitivity, 'and now it affects the sensitivity function directly. The removed features included small variance features.

REFERENCES

1. MOZER, M. C. and SMOLENSKY, P.: **Skeletonization: A Technique for Trimming the Fat from A Neural Network,** in D.S.Touretzky (Ed.) Advances in Neural Information Processing Systems 1, MORGAN KAUFMANN, San Mateo, USA, April 1989, pp. 107-115.
2. LISBOA, P. J. G, MEHRIDEHNAVI., A. R. and MARTIN, P. A.: **Opening Pandora's Box: A Logarithmic Sensitivity Matrix for the Interpretation of Neural Networks,** Proc. of the International Conference on NNESMED, Plymouth, UK, August 1994, pp 1-8.
3. HERTZ, J., KROGH A. and PALMER, R.: **Introduction to the Theory of Neural Computation,** Santa Fe Institute, ADDISON- WESLEY PUBLISHING COMPANY, Chapter 6, 1991, pp. 147-158.