International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-3 Issue-12, May 2014

Neural Networks New Capacity Factor Measurement for Improvement of SOM

Vahid Rahmati, Morteza Husainy Yar, Ali Reza Malekijavan

Abstract*— the artificial neural networks have an important role in current life with higher expectations. The art of using these ANNs give us a good insight for problem solving. For example the applications in pattern recognition and regression are two areas in which ANNs are working well. Signal processing itself investigates broad ranges of ANNs. The purpose of this paper is, scanning connection routes of one and two layer networks that may be used as default structure in data replacing and signal analyzing. When a signal is considered as a variable by a problem solver, the problem solver chooses the best possible ANNs to solve it. But the way we ensure the high possibilities of reliability for these types of networks, while the compatibility is still needed is important. We present several factors to measure the capability of a specific network, for a specific problem the one like E-machine learning. Formal proofs for claim are provided as well. Finally we try to optimize Kohonen SOM using factor C'.

***Index Terms*— Artificial neural networks, Kohonen SOM, Machine learning, ANN connection route**

I. INTRODUCTION

It becomes really interesting to transform problems in a way to solve them ironically like the one we see while working with computers. Solving is equal to parsing commands line by line or operation by operation. We can see this binary system does not work well when we expect an efficient (fast) solution [1] for some problems. For example finding a sampled data signal in a huge list, overflow occurs because of higher than usual memory access times as demands. The primary solution for this problem is, to use hash tables with appropriate lengths. Even by using hash tables we don't achieve best performance.

So using artificial neural networks (ANN) is necessary because it mitigates the learning process for an algorithm. For this purpose we go for an efficient algorithm, the one is needed to full fill the need of the problem. Now how do you call an algorithm, the efficient one? Faster is better, Simple, because the speed matters! , the next question is how do you call an ANN, a good ANN? Not simple, and doubts arise. If we define a good algorithm by its speed we can also define a good ANN by its capacity of learning and problem solving at the same time when it needs to handle a big problem.

Here big means the one that needs more than enough number of operations in binary system, or the algorithm with high (exponential) complexity time. Or a CPU clock eater giant. Consider that navigation of a drone or an airplane using neural networks and as auto pilot is strategic.

If we learn how to navigate or how to drive the ANN can learn it too! So if we have a good driver and a careless driver, we can have a good ANN navigation system and a bad one! So do you accept the risk of producing an unsafe product, when the safety of ANN is not estimated well?

We want to scan and evaluate the capacity of an ANN while considering its efficiency of learning. If our ANN learns fast it must learn reliable too. The factor of reliability (r) is connected to the number of errors. It means the number of terms and operations for convergence and the accuracy of solution prediction come together to show reliability factor. This is heavily depends on seeds for a task. Here seeds are equal to initializing values. If a full reliable system exists, it has a factor of r=1, and the worst case is r=0. The number of correct operations (pc) is the number of operations that lead us to a solution.

For drone example, the auto pilot worst case is an unsuitable weather. In other [2] parts we estimate the capacity of some networks to show the ability of these factors.

Here in a table we list several factors of measures that is used for optimizing a chosen ANN, We prefer to improve SOM type networks so the weighting problem wouldn't often happen as matrixes are updated during runs and not as predefined list of vectors of numbers. In this case creativity of SOM is considered more than static fixed weighting matrix based ANNs.

Table-1

Name	description	Name	description
ANN_i	Ith artificial neural network	r_i	Reliability for ANN and problem round i
w	Number of wrong operations	P_c	Number of correct steps
c	Number of correct operations	P_w	Number of wrong steps
Gp	Group of problems	C'	Learning capacity factor of an ANN
X_i	Input node	t	Total number of attempts to converge
Y_i	Output node	SOM	Self-Organizing Map

II. NEED FOR A GENERAL CAPACITANCE FACTOR

Known definition of capacity, that is the ability to model any given function does not bring good connections to natural neural networks. For human brain as a good example, the capacity has several meanings. One may say the capacity of memorizing, learning, reading,… and not necessary modeling [4], which is what we consider in this paper.

A human can solve a problem that is new and we don't have any model or function for it, he solves because natural neural network is creative not because it is able to model something that already exists.

Manuscript published on 30 May 2014.
*Correspondence Author(s)
Vahid Rahmati, is currently studying M.sc at Shahid Sattari University, Tehran, Iran.
Morteza Husainy Yar, is currently studying M.sc at Shahid Sattari University, Tehran, Iran.
Dr. Ali Reza Malekijavan, is a professor at Shahid Sattari University, Tehran, Iran.

The ability of natural neural networks to bring new solutions or the things that didn't really exist before has one meaning, and that is creativity and not power of modeling [5].
Modeling is vital for a task to be done, but again, mechanically, and also when the function is ready. So finding a new definition for capacity is good enough. Now ANNs are practically used by modeling a problem solver, but they are not necessary capable to solve all types of problems. This fact must be distinguished because natural neural networks can do this! The question is where is the difficulty for ANNs being creative? The answer is not known [6].
Capacity can be considered as the total ability to do several tasks (of the same type or in the same family of problems) by the average of reliability factors.

III. FACTORS OF MEASURES

Here we list factors of learning estimation for an ANN so we can measure a network for a specific problem or a group of problems.
Factor of *wrong* and *correct* operations are defined as:

$$p_w = 1 - \frac{w}{t} \quad (1)$$

$$p_c = 1 - \frac{c}{t} \quad (2)$$

Where w is the number of wrong operations while c is the number of correct operations and t is the total umber of attempts to solve the problem [3]. It is very clear that:

$$t = w + c \quad (3)$$

Reliability factor: Shows reliability of an algorithm or an ANN for doing a task and is defined by this identity:

$$r \triangleq \frac{p_c}{1+p_w} \quad (4)$$

If r is equal to 1, then the reliability is 100% and, ANN does the job fully. The worst case is r=0 and happens when the worst case for t is infinity, and algorithm doesn't converge at all.
Group of problems: is a family of similar problems and showed by:

$$Gp_m = \{p_1, p_2, \dots, p_m\} \quad (5)$$

Capacity of an ANN: is the capacity of learning that is proportional to reliability average for an ANN, over m problems. This factor (C') is defined as:

$$C' \triangleq \frac{1}{m}\sum_{i=1}^{m} r_i \quad (6)$$

If the capacity is equal to 1, the ANN is capable to solve all problems in Gp_m.

IV. CALCULATING C' OVER GPM

The main issue is how to initialize an ANN for several problems. We don't provide any seed, or initialization method, but analyzer must do so otherwise that ANN is tabbed into a specific class of problem and natural neural networks are not tabbed into one type of problem so distinction wouldn't happen. Means we just try to use the factors calculated above to trust an ANN. If the ANN_1 provides a better capacity respect to ANN_2, it must be used. Here you can see that the flexibility of your ANN can be measured. And if user switches to a new problem in Gp_m the ANN doesn't give up and provides an optimal solution.
Gp_m consists of m problems that are in the same group, or simply similar problems. if p_1 is a problem of solving XOR operations, then we cannot expect p_2 to be vector arranging problem. The m problems must be consistent of the same type [7]. If it is necessary to solve more than one type of problem, one can use a different differential equation to change weights in the network. Or the user of these factors must use new activation functions, without changing the architecture of network. We do not concentrate on changing algorithm purpose so the C' factor can be calculated fairly. If the problems in Gp_m are not really from the same family of problems the C' can't be trusted, if the user switches between problems without seeding [8] and initializing the weights. By this assumption we measure the capacity to verify the quality of an ANN for doing multi task missions.

V. C' MEASUREMENT

Assumptions:

- Activation function for Pi is defined and used by user for each separate problem in Gpm.
- Weights are varied respect to ith problem, Pi.
- Activation function converges.

Theorem 1:
If 3 previous assumptions hold, then The C' can be measured and is always less than and equal to 1.
Proof 1: first of all we must consider activation function like $f_i(x)$ for multi input/output nodes X_i, Y_i so that the relation is:

$$y_j = f_j(b_j + \sum_i x_i w_{ij}) \quad (7)$$

If value for y_j converges and the number of operations [9] for worst case is t then P_w is less than 1 and the same story for r_i happens. So the r factor is less than one. Please notice that a certain algorithm must have a worst case means: "The maximum number of operations is needed to fulfill the need of the system" .
Example of C' MEASUREMENT FOR SOM network
In this example we try to measure C' factor for a Kohonen Self-Organizing Map. As we know these types of network are like natural neural networks that form a topologic structure, the one we don't expect from other types of ANNs. Here we can see this type of map.
Here we have a graph shows the structure of a Kohonen SOM, that contain clusters in which input signals are injected

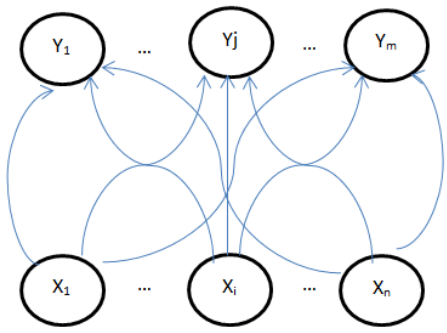

Fig. 1. The structure of Kohonen-SOM

The weights are not shown, and are depended on problems. Here we do not want to use C'[11] for a class of problems, but can use it to measure at least for one problem with 2 different cases. For each input vector, we have to calculate j parameters by (8):

$$D(j) = \sum_i (w_{ij} - x_i)^2 \quad (8)$$

Now we consider α (0) =0.7, α (t+1) =0.5α (t), then the weight matrix is:

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

$$w_1 = \begin{bmatrix} 0.2 & 0.75 \\ 0.5 & 0.45 \\ 0.55 & 0.7 \\ 0.9 & 0.25 \end{bmatrix}, w_2 = \begin{bmatrix} 0 & 1 \\ 0.001 & 045 \\ 0.55 & 0.06 \\ 0.89 & 0 \end{bmatrix}$$

If R=0 we can find the C' for weight matrix for w_1 and w_2.
To solve this we simply apply algorithm by initializing weights and using (9) and new weight matrix formula. The worst case total operations(or steps) is considered as t=200.
For w_1 we have: p_c=0.5, p_w=0.5 hence r=0.33.
For w_2 we have: p_c=0.75, p_w=0.25 hence r=0.6.
For both problems in Gp_2 the C'=0.465.
As we see capacity depends on weights as well. The same problem can be solved successfully if the weights are in range.

VI. APPLICATION OF C' FOR SOM OPTIMIZATION

The main goal for calculating C' is to optimize the weights matrix. Here we don't define any initializing values but we can improve it in running time. The optimization is equal to making r equal to 1. It is very clear that for the first round the C' is absolutely non-one[10]. So the set of problems in Gp_m defines the accuracy of capacity factor. For example if Gp_5 is used 5 problems are equal to 5 rounds in this algorithm and r is calculated separately and C' is the average value.
When the algorithm is initialized by user weight matrix the first few rounds show the reliability, and we easily measure C', if C' is near zero we modify weighting formula, if it's near 1 we don't modify formula. Each modification depends on m or index of problem group.
By this trick we alter weight matrix to converge as fast as possible. C' is in fact a dynamic factor that is modified over rounds. This inline C' calculation makes the algorithm faster. The memory consumption for calculating C' is not too much and can be ignored.
Here we provide modified version of Kohonen SOM algorithm. By this C' we can alter w_{ij} to help the algorithm converge faster[12]. Here we modify Kohonen SOM algorithm and make it a bit faster by improving weighting process. This happens while the algorithm is running.
Algorithm:

1. Define wij weights and learning rate.
2. While not stopped loop for 2 to 8.
3. For each vector x repeat 3 to 5. If new vector then reset pc and pw.
4. Find D (j) by (8).
5. Make D (j) minimum for specific j.
6. Calculate pc and pw for number of loops from 2 to 8.
7. Calculate C' in Gpm where m is the number of input vectors.
8. If d= wij (new)-wij (old) < 10-3 and 1-C'<10-3 go to 12
9. wij (new) = wij (old)+ 0.5α (xi-wij(old)).
10. If d= wij (new)-wij (old) < 10-3 go to 12
11. wij (new) = wij (old)+ α (xi-wij(old))
12. Go to 2
13. Stop.

In this algorithm the weighting matrix is modified according to convergent speed the step 8 can use an alternative form, but the convergence criteria must be considered as well.

VII. CONCLUSIONS

We measured a factor called C' to increase speed of a chosen SOM. This lets us to make better algorithms. In this paper we used C' as a new factor. But this capacity factor does not ensure convergence. The only important factor for convergence is the weighting matrix itself. We brought an example of modified Kohonen SOM. It makes the algorithm faster. We calculated this for 10 vectors of different length by using C'. In figure 2 readers can see the improvements.

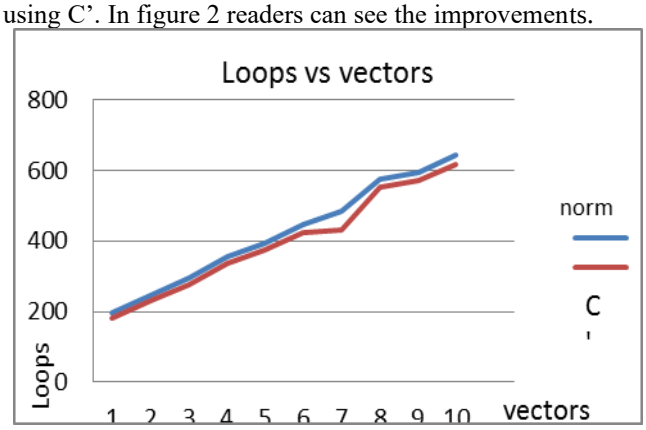

Figure 2-Loops vs. vectors blue is normal algorithm vs red that is optimized

REFERENCES

1. Christopher M. Bishop (Jan 18, 1996), Neural Networks for Pattern Recognition
2. Simon O. Haykin (Nov 28, 2008), Neural Networks and Learning Machines
3. Sandhya Samarasinghe (Sep 12, 2006), Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition
4. Ke-Lin Du and M. N. S. Swamy (Dec 27, 2013), Neural Networks and Statistical Learning
5. Jeff Heaton (Oct 2, 2008), Introduction to Neural Networks for C#
6. Laurene V. Fausett (Dec 19, 1993), Fundamentals of Neural Networks: Architectures, Algorithms And Applications
7. Jeff Heaton (Oct 1, 2008), Introduction to Neural Networks for Java
8. Kevin Gurney (Aug 7, 1997), An Introduction to Neural Networks
9. Olaf Sporns (Oct 1, 2010), Networks of the Brain
10. Kohonen, T. ; Neural Networks Res. Centre, Helsinki Univ. of Technol., Espoo, Finland ; Oja, E. ; Simula, O. ; Visa, A.more authors, Engineering applications of the self-organizing map
11. SangHak Lee ; Ubiquitous Comput. Res. Center, Korea Electron. Technol. Inst., South Korea ; JuneJae Yoo ; TaeChoong Chung, Distance-based energy efficient clustering for wireless sensor networks
12. Jari Kangas,Teuvo Kohonen,Developments and applications of the self-organizing map and related algorithms

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)