Influence Activation Function in Approximate Periodic Functions Using Neural Networks

Luma N. M. Tawfiq
Ala K. Jabber
Dept. of Mathematics/College of Education for Pure Science (Ibn Al-Haitham) / University of Baghdad.

Received in :30 December 2012 , Accepted in: 9 September 2012

Abstract

The aim of this paper is to design fast neural networks to approximate periodic functions, that is, design a fully connected networks contains links between all nodes in adjacent layers which can speed up the approximation times, reduce approximation failures, and increase possibility of obtaining the globally optimal approximation. We training suggested network by Levenberg-Marquardt training algorithm then speeding suggested networks by choosing most activation function (transfer function) which having a very fast convergence rate for reasonable size networks.

In all algorithms, the gradient of the performance function (energy function) is used to determine how to adjust the weights such that the performance function is minimized, where the back propagation algorithm has been used to increase the speed of training.

Keyword: Artificial neural network, Training network, Activation Function.

Introduction

An Artificial neural network (Ann) is a simplified mathematical model of the human brain, It can be implemented by both electric elements and computer software. It is a parallel distributed processor with large numbers of connections, it is an information processing system that has certain performance characters in common with biological neural networks.[1] Ann have been developed as generalizations of mathematical models of human cognition or neural biology, based on the assumptions that:

1- Information processing occurs at many simple elements called neurons that is fundamental to the operation of Ann's.

2- Signals are passed between neurons over connection links.

3- Each connection link has an associated weight which, in a typical neural net, multiplies the signal transmitted.

4- Each neuron applies an action function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output sign.

The units in a network are organized into a given topology by a set of connections, or weights, shown as lines in a diagram .

Ann is Characterized by [2] :

1- Architecture: its pattern of connections between the neurons.

2- Training Algorithm : its method of determining the weights on the connections.

3- Activation function.

Ann are often classified as single layer or multilayer. In determining the number of layers, the input units are not counted as a layer, because they perform no computation. Equivalently, the number of layers in the net can be defined to be the number of layers of weighted interconnects links between the slabs of neurons. This view is motivated by the fact that the weights in a net contain extremely important information [3].

Ann have been exploited to approximate function to overcome the limitations of traditional classical methods . A number of researcher used Ann's in approximation problems say:[4],[5],[6],[7], [8]. In this paper we try to determine the optimal network architecture for approximation periodic functions.

A Framework for Distributed Representation[9]

A set of major aspects of a parallel distributed model can be distinguished :

- A set of processing units ('neurons' or 'cells') ;
- A state of activation y_k for every unit, which is equivalent to the output of the unit
- Connections between the units. Generally each connection is defined by a weight w_{jk} which determines the effect which the signal of unit j has on unit k ;
- A propagation rule, which determines the effective input s_k of a unit from its external inputs ;
- An activation function F_k, which determines the new level of activation based on the effective input $s_k(x)$ and the current activation $y_k(x)$ (i.e., the update) ;
- An external input (bias) b_i for each unit ;
- A method for information gathering (the training rule) ;
- An environment within which the system must operate, providing input signals and if necessary error signals.

Activation Function

The activation function (sometimes called a transfer function) can be a linear or nonlinear function. There are many different types of activation functions. Selection of one type over another depends on the particular problem that the neuron (or Ann) is to solve .

The activation function, denoted by $\phi : R \to R$ defines the output of a neuron, which is bounded monotonically increasing, differentiable and satisfies : $\text{Lim}_{x\to+\infty} \phi(x) = 1$ and $\text{Lim}_{x\to-\infty} \phi(x) = 0$.

The sigmoid function, is by far the most common form of activation function used in construction of Ann's. An example of the sigmoid function is the logistic function defined the range from 0 to +1, an important feature of the sigmoid function that it is differentiable [4].

It is sometimes desirable to have the activation function range from –1 to +1 allowing an activation function of the sigmoid type to assume negative values, in which case the activation function assumes an anti-symmetric form with respect to the origin, i.e., one that satisfies $f(-x) = -f(x)$,, for example ,the hyperbolic tangent function which has a particularly simple derivative: $\phi(x) = \tanh(x)$, $f'(x) = 1 - f(x)^2$.

In this paper, we study suitable activation function, depending on the results of training many Ann to approximate periodic functions with different activation function .

Training Ann[10]

Training is the process of adjusting connection weights w and biases b. In the first step, the network outputs and the difference between the actual (obtained) output and the desired (target) output (i.e., the error) is calculated for the initialized weights and biases (arbitrary values). During the second stage, the initialized weights in all links and biases in all neurons are adjusted to minimize the error by propagating the error backwards (the back propagation algorithm). The network outputs and the error are calculated again with the adapted weights and biases, and the process (the training of the Ann) is repeated at each epoch (The number of iterations of training process) until satisfied output y_k (corresponding to the values of the input variables x) is obtained and the error is acceptably small.

There are **three types** of training in which the weights organize themselves according to the task to be learnt, these types are : supervised, semi - supervised (or reinforcement) and unsupervised training.

In this paper an supervised training algorithm is used.

Supervised Training [10]

In which a “teacher” provides output targets for each input pattern, and corrects the network’s errors explicitly, that is, there is a set of training samples and neural network adjusts its connection weights according to the difference between the given outputs and the actual outputs.

Back Propagation Training Algorithm

Back propagation can train Ann with differentiable transfer functions to perform function approximation, other types of networks can be trained as well, although the multilayer network is most commonly used. The term back propagation refers to the process by which derivatives of network error, with respect to network weights and biases, can be computed, that is, during training an error must be propagated from the output layer back to the hidden layer in order to perform the training of the input-to-hidden weights. This process can be used with a number of different optimization strategies.

Standard back propagation is a gradient descent algorithm, as is the
Widrow - Hoff training rule, in which the network weights are moved along the negative of the gradient of the performance function. [11]

Back propagation algorithm is summarized as :
•Select a network architecture.
•Initialize the weights to small random values.
•Present the network with training examples from training set in some order. It helps to randomize the order of presentation of examples in each pass of the training set, for each training example .
•Forward pass: compute the net activations and outputs of each of the neurons in the network, e.g. y_j , F_k or ϕ_k
•Backward pass: compute the errors for each of the neurons in the network, e.g., E_k, E_j
•Update weights
•If the total error E falls below some threshold, then stop

Levenberg-Marquardt Algorithm (trainlm)

The Levenberg - Marquardt algorithm was designed to approach second order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares, then the Hessian matrix can be approximated as $H = J^TJ$ and the gradient can be computed as $g = J^Te$, where J is the Jacobian matrix, which contains first derivatives of the network errors with respect to the weights and biases, and *e* is a vector of network errors. The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton update: $W_{k+1} = W_k - [J^TJ + \mu I]^{-1} J^T e$
when the scalar $\mu = 0$, this is just Newton's method. When μ is large, this becomes gradient descent with a small step size. [10]

Approximation by Ann

For the general problem of function approximation, the universal approximation theorem proved in [5], [6] states that:

Theorem 1

Standard Ann with only one hidden layer can approximate any continuous function uniformly on any compact set and any measurable function to any desired degree of accuracy.

An immediate implication of the above theorem is that any lack of success in applications must arise from inadequate learning, insufficient number of hidden units, or the lack of a deterministic relationship between the input and the target. A second theorem proved in [12] provides an upper bound for the architecture of an Ann destined to approximate a continuous function defined on the hypercube in R^n .

Theorem 2

On the unit cube in R^n any continuous function can be uniformly approximated, to within any error by using a two hidden layer network having 2n+1 units in the first layer and 4n+3 units in the second layer.

Eqhaar [6] provides the architecture of an Ann destined to approximate any continuous function in R^n as follows :

Theorem 3

Any bounded continuous function in R^n can be uniformly approximated, to within any error by using one hidden layer network having 2n+1 units in the hidden layer and the architecture of Ann with multi hidden layer having double units which contained in the previous layer plus one .

Applications

We applied multilayer Ann with ridge basis function that have linear output unit and a single hidden layer with different sigmoid transfer functions. The number of hidden nodes in all problems is 2N+1, where N is the number of the input nodes. The training problems used problem domains periodic function approximation and we training each problems 10 different times and the weights of the networks are computed by back propagation algorithm with Levenberg - Marquardt training algorithm: trainlm .

Problem 1

Consider a function : $F(x) = 3 \sin(-2x)$; where $0 \le x \le 2\pi$

The numerical results of Ann with network structure 1–5–1,are introduced in table (1), while table (2) gives initial weight and bias for the designer network .

Problem 2

Consider a function : $F(x) = \cos(x+5) - 10$; where $0 \le x \le 2\pi$

The numerical results of Ann with network structure 1–5–1, are introduced in table 3, and table 4 gives initial weight and bias for the designer network .

Conclusions

It is very difficult to know which activation functions will be the fastest for a given problem. It will depend on many factors including the complexity of the problem, the number of data points in the training set, the number of weights and biases in the Ann, and the error goal, whether the Ann .

In general, the practical results on Ann showed the tansig activation function will have the fastest convergence, Then the logsig .

The performance of the various algorithms can be affected by the accuracy required of the approximation.

References

1. Galushkin, I. A., (2007), "Neural Networks Theory", Berlin Heidelberg.
2. Hristev, R. M., (1998), " The ANN Book ", Edition 1, Springer.
3. Villmann, T., Seiffert, U., and Wismüller, A., (2004), "Theory and Applications of Neural maps", Esann2004 Proceedings - European Symposium on Ann, pp. 25 - 38.
4. Cybenko, G., (1989), "Approximation by superpositions of a sigmoidal function", Mathematics of Control, Signals, and Systems, Vol. 2, pp. 303-314.
5. Hornik, K., and Stinchombe, M., (1992), "Multilayer feed-forward networks are universal approximators", in f2Artificial Neural Networks: Approximation and Learning Theory, H. White et. al., Eds., Blackwell press, Oxford.
6. Tawfiq, L. N. M., and Naoum, R. S., (2007), "Density and approximation by using feed forward Artificial neural networks", Ibn Al-Haitham Journal for Pure & Applied Sciences, Vol. 20 , No. 1.
7. Eqhaar, Q. H., (2007), "Comparison Between Artificial Neural Networks with Radial Basis Function and Ridge Basis Function for Interpolation Problems, MSc Thesis, Baghdad University, College of Education Ibn Al-Haitham.
8. Su, T., Jhang, J., and Hou, C., (2008), " A Hybrid Artificial Neural Networks and Particle Swarm Optimization for Function Approximation", International Journal of Innovative Computing, Information and Control ICIC International, Vol. 4, No. 9, pp. 2363–2374.
9. Tawfiq, L. N. M., and Eqhaar, Q. H., 24 to 26 March (2009), "On Multilayer Neural Networks and its Application For Approximation Problem, 3rd scientific conference of the College of Science", University of Baghdad.
10. Kröse, B., and van der Smagt, P., (1996), "An introduction to Neural Networks, Eighth edition , Amsterdam.
11. Tawfiq, L. N. M., and Naoum, R. S., (2005), "On Training of Artificial Neural Networks" , AL-Fath Jornal, No 23.

12. Hetcht-Nielsen, R., (1989), "Theory of the back propagation neural networks", Proc. Inter. Joint Conf. Neural Networks, Vol. I, pp. 593- 611.

13. Turmon, J. M., (1995), "Assessing Generalization of Feed forward Neural Networks", PhD. thesis, University of Cornell.

Table 1: Training results after several independent trials for problem 1 algorithms

The training Epochs and Performance				
NO.	ActivationFunction	Perf. Test	Time/S	Epochs
1	tansig.	0.00033561	0.032	9
2	logsig.	0.00047319	0.172	69
3	satlins	0.0082778	0.047	7
4	radbas	0.19392	0.562	245
5	hardlim	0.85046	0.032	3
6	satlin	2.3777	0.047	9
7	tribas	2.9771	0.047	10
8	purelin	3.8175	0.031	3
9	softmax	4.2142	0.094	4
10	traincgr			

The parameters	
Time	Time
Show	Show
Epochs	Epochs
Goal	Goal

Table 2 : The initial Weight and Bias value for problem 1

net.IW{1,1}	net.LW{2,1 }	net.B{1,1 }	net.B{2,1}
2.228169203	-0.86937099	-14	0.99906335
2.228169203	*-0.53140053*	*-10.5*	
-2.228169203	0.86619684	7	
2.228169203	-0.87374421	- 3.5	
- 2.228169203	- 0.47156467	0	

Table 3: Training results after several independent trials for problem 2

The training Epochs and Performance				
NO .	Activation Function	Perf. Test	Time/S	Epochs
1	tansig.	0.00015586	0.046	10
2	logsig.	0.00045284	0.047	8
3	satlins	0.011864	0.047	13
4	radbas	0.00027366	0.046	10
5	hardlim	0.094551	0.015	3
6	satlin	0.011963	0.047	17
7	tribas	0.00040426	0.032	7
8	purelin	0.22308	0.046	4
9	softmax	0.24979	0.031	3
10	traincgr			

The parameters	
Time	Time
Show	Show
Epochs	Epochs
Goal	Goal

Table 4 : The initial Weight and Bias value for problem 2

net.IW{1,1}	net.LW{2,1}	net.B{1,1}	net.B{2,1}
2.228169203	-0. 65523339	-14	0.1748828
2.228169203	-0.92603403	-10.5	
-2.228169203	-0.374891	7	
-2.228169203	0.63456135	3.5	
2.228169203	- 0.5307266	0	

تأثير دالة الاستثارة في تقريب الدوال الدورية باستخدام الشبكات العصبية

لمى ناجي محمد توفيق
علاء كامل جابر
قسم الرياضيات / كلية التربية للعلوم الصرفة (ابن الهيثم) / جامعة بغداد

أستلم البحث في 30 أيلول 2012 ، قبل البحث 9 كانون الأول 2012

المستخلص

الهدف من هذا البحث هو تصميم شبكات عصبية مسرعة كطريقة لتقريب الدوال الدورية وهذا يعني تصميم شبكات مرتبطة بالكامل تتضمن روابط بين كل العقد في الطبقات المتجاورة و التي تستطيع تعجيل زمن التقريب ، تقليل حالات الإخفاق ، الفشل و زيادة احتمالية الحصول على التقريب المثالي الرئيسي ، دربنا الشبكات المقترحة بطريقة ليقنبرك- ماركواردت و من ثم تسريع الشبكات المقترحة من خلال اختيار أفضل دالة استثارة (دالة انتقال) إذ أن بعضها يمتلك نسبة تقارب سريعة جدا لشبكات ذي إحجام معقولة، في كل تلك الخوارزميات استخدمنا ميلَ دالةِ الأداءَ (دالة الطاقةِ) لتَحديد كيفية تَعديل الأوزانِ من خلال تصغير دالةِ الأداءَ ، إذ استخدمت خوارزميةِ الانتشار المرتد لزيَادَة سرعةِ التدريب.

الكلمات المفتاحية : الشبكات العصبية الصناعية ، تدريب الشبكات ، دوال الاستثارة .