Sinc-function based Network

Madsen, Per Printz

Publication date:
1998

Document Version
Også kaldet Forlagets PDF

Sinc-function based Network

Per Printz Madsen
Ph.D, M.Sc.E.E.
Dept. of Control Engineering, Inst. of Electronic Systems
Aalborg University, Fredrik Bajers vej 7 DK-9220 Aalborg , Denmark
E-mail: ppm@control.auc.dk

December 9, 1998

Abstract

The purpose of this paper is to describe a neural network (SNN), that is based on Shannons ideas of reconstruction of a real continuous function from its samples. The basic function, used in this network, is the Sinc-function. Two learning algorithms are described. A simple one called IM-learning and a more complex one based on least square optimization. These two learning methods are illustrated by an example.

1 Introduction

In the last decade there has been put a lot of effort into research in Artificial Neural Networks (ANN). A large variety of ANN's has been developed, described and used. These different types of network have been used in many applications. One of the working fields is the use of ANN for simulation and controlling of dynamic processes fx [3]. In this case the main purpose for the ANN is to reconstruct a real continuous function from its samples. Multi-Layer Perception (MLP) and Radial Basis Function networks (RBF-network) are some of the commonly used ANN's in this working field. The main reason to choose these models is the fact, that they are able to approximate any multi-variable and non-linear function.

Function approximation is often based on an other method, that uses sinc-functions as the basic function. Through at least the last 20 years sinc-function has successfully been applied to function approximation and to the solution of differentials. In spite of this these methods have not reached the mainstream ANN community. Using ANN based on sinc-functions (SNN) different kind of advantages are provided. One of the most important advantage is, that the SNN is a more or less approximation to the ideal filter and for that reason, in some cases, gives at better understanding of how the network are working.

2 Basic theory

When using ANN for simulation purpose the main problem is how to reconstruct a real continuous function from its samples. The first to deal with this problem was Shannon [4].

The following is one of the main results in sampling theory.

Sampling Theorem 1 *Let $f : \mathcal{R} \mapsto \mathcal{R}$ be such, that both its Fourier and inverse Fourier transformation are well-defined. If the spectrum $F(\omega)$ is zero for $|\omega| > \omega_0$, then $f(x)$ can be exactly reconstructed from the characteristic samples $\{f(kx_\Delta)\}_{k\in Z}$, $x_\Delta = \pi/\omega_0$.* $\diamond$

In other words. If the function is band-limited, its spectrum is zero outside the interval $[-\omega_0, \omega_0]$ then the theory states, that the exact function, generated from its samples, is given by:

$$f(x) = \sum_{k=-\infty}^{\infty} f(kx_\Delta) \frac{\sin((\pi/x_\Delta)(x - kx_\Delta))}{(\pi/x_\Delta)(x - kx_\Delta)} \tag{1}$$

This function has a structure like a neural network with an infinite number of neurons in the hidden layer. The neuron function is the non-causal impulse response from an ideal low-pass filter given by: $g_k(x) = \frac{\sin((\pi/x_\Delta)(x-kx_\Delta))}{(\pi/x_\Delta)(x-kx_\Delta)}$. See figure: 1

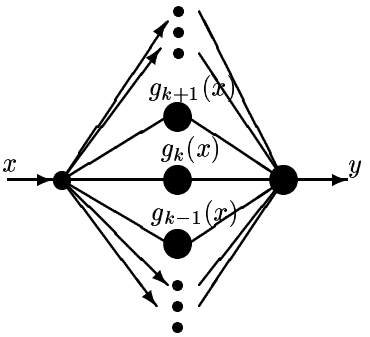

Figure 1: Structure of the ideal SNN for SISO-function estimation.

With respect to Theorem:1 there are two assumptions. Firstly the spectrum $F(\omega)$ vanishes for $|\omega| > \omega_0$. This can be obtained by an anti-aliasing pre-filter, that gives a suitable signal rejection beyond ω_0. Secondly if y is periodic, piecewise continuous, bounded and left- and right-hand derivative at every point exists, then the Fourier and inverse Fourier transforms are well-defined. Periodic functions can always be obtained by repeating a given interval of the function. Or in other words, if $\tilde{x} = x + k \cdot x_P$ where x_P is the length of one period, then the k'te period of the system output can be obtained by: $y(\tilde{x}) = f(x)$.

The ideal SNN network requires an infinite number of periods and for that reason an infinite number of neurons in the network. In practical use the approximation must be based on n samples distributed equally over M periods. The number of samples must at least fulfill the Nyquist sampling rate (two samples per period of the highest frequency). If the highest frequency isn't known the number of samples must be based on experiments. A useful rule is; if the system is a monotone nonlinear function, then the minimum is three samples per period. But often it is necessary to use four to six per period, or even more.

If the total input interval for the learning data-set is $[\tilde{x}_{\min}, \tilde{x}_{\max}]$, then the first characteristic sample is placed in $\tilde{x}_1 = \tilde{x}_{\min}$ and the last characteristic sample is placed in $\tilde{x}_n = \tilde{x}_{\max}$. The $n-2$ other samples are placed with equal space between these two limits.

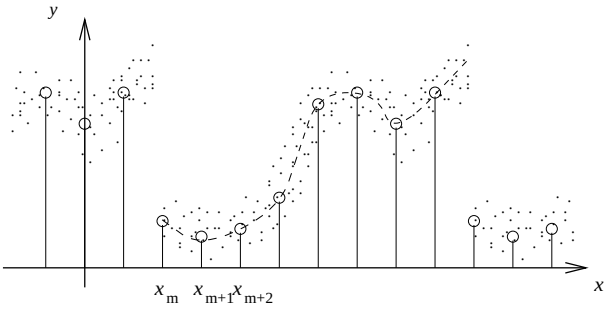

Figure 2: The characteristic samples of a given function.

Figure 2 shows a little more than one period of a function, the characteristic samples for this function and furthermore some learning data "·".

n-Dimensional sampling Theorem 1 *Let $f : \mathcal{R}^n \mapsto \mathcal{R}$ be such, that both its n-dimensional Fourier, and its inverse Fourier transforms are well-defined. If the spectrum $F(\omega_1, \omega_2, \ldots, \omega_n)$ is zero outside a bounded subset of $\mathcal{R}^n$, given by: $|\omega_1| < \omega_{01}, |\omega_2| < \omega_{02}, \ldots, |\omega_n| < \omega_{0n}$, then $f(x)$ can be exactly reconstructed from its characteristic samples taken over a lattice of points $\vec{x}_k = k_1 v_1 + k_2 v_2 + \cdots + k_n v_n$ where: $k_1, k_2, \ldots, k_n$, are integers and k is the index over the total number of points, where v_i are small enough to ensure non-aliasing in the spectrum of the sampled signal.*

If $\vec{x} = (x_1, x_2, \ldots, x_n)$ and the Nyquist sampling rate in each direction is fulfilled [1] then the n-dimensional case is given by eq:2

$$f(\vec{x}) = \sum_{k_1=-\infty}^{\infty} \cdots \sum_{k_n=-\infty}^{\infty} f(\vec{x}_k) \cdot g(\vec{x} - \vec{x}_k) \tag{2}$$

where:

$$g(\vec{x}) = \prod_{m=1}^{n} \frac{\sin(\omega_m x_m)}{\omega_m x_m} \tag{3}$$

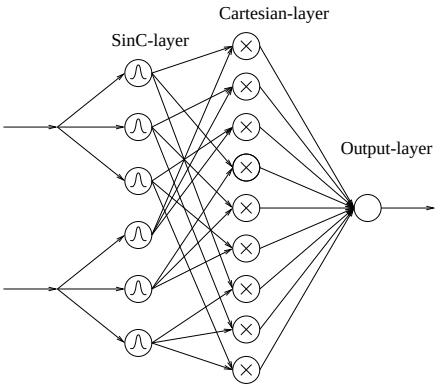

Figure 3: A two input one output SNN network.

Figure: 3 shows the structure of a two inputs and one output SNN network. The first layer (The SinC-layer) contains one neuron for each point on this input lattice. In this case the two inputs are divided into tree samples. The space between each characteristic sample in each direction, and thereby the number of neurons in the SinC-layer, ought to be chosen so that the Nyquist sampling rate in each direction is fulfilled. The next layer (The Cartesian-layer) calculates the $g(\vec{x})$ function from eq: 3. Each output from the Cartesian-layer is multiplied with the corresponding desired output: $f(\vec{x}_k)$ and then added in the output-layer.

3 Method

It is necessary to consider three subjects, when the SNN is applied to a given application. These subjects are: (1) To choose the space between the characteristic samples in each direction in the input. (2) To apply a sinc-function to each point in this input lattice. (3) To estimate a common output value in each point on this input lattice based on some learning data.

Re point one: This can be a difficult task. If you have some ideas of the frequency response of the system, then you can use this information. Bear in mind that the frequency is the transform of the amplitude $\boldsymbol{x}$. This kind of network is a low-pass filter with a cutoff frequency $\omega_o = \pi / x_\Delta$.

Re point two This step is a trivial task.

Re point three Eq: 2 requires that $f(\vec{x}_k)$ is known. $f(\vec{x}_k)$ is the desired output for each input point in the input lattice. These values can be calculated from the training data. This calculation or network learning can be carried out in different ways. Two of these methods are respectively learning based on interval means and learning based on least square estimation.

3.1 Network learning

The SNN can reconstruct an infinite length periodic function based on samples in the input lattice. In the real case it is not convenient to work with an infinite number of samples and thereby have an infinite number of neurons in the SinC-layer. In practice two periods before and two periods after the original interval give good results.

The parameters in the network is ω_m and $f(\vec{x}_k)$. See eq: 2 and 3. The learning algorithm, described in this paper, demands that ω_m is chosen before learning, so that Shannon samplings theorem is fulfilled.

The learning is therefore a method to estimate $f(\vec{x}_k)$. This estimate is given by: $\hat{f}(\vec{x}_k)$

In other words the learning is a method to estimate characteristic output in each point in this input lattice. Interval means (IM) is based on a calculation of the means of the desired output.

The learning algorithm for IM-learning:

1. Select the next input/desired-output data pair $(\vec{\boldsymbol{x}}, \vec{\boldsymbol{y}}_d)$.
2. Decide which point in the input lattice, that is closest to $\vec{\boldsymbol{x}}$. This can be done by calculation p_i, given by:

$$p_k = \begin{cases} 1 & \text{if } ||\vec{\boldsymbol{x}}_k - \vec{\boldsymbol{x}}|| \leq ||\vec{\boldsymbol{x}}_j - \vec{\boldsymbol{x}}|| \text{ for all } j \\ 0 & \text{else} \end{cases}$$

3. If $p_k = 1$ then $\vec{\boldsymbol{Y}}_k = \vec{\boldsymbol{Y}}_k + \vec{\boldsymbol{y}}_d$ and $c_k = c_k + 1$
4. If more learning-data, then go to step 1.
5. $\vec{\boldsymbol{Y}}_k = \vec{\boldsymbol{Y}}_k / c_k$ for all k

j are an index over the total number of input/desired-output pairs in the learning-data. $\vec{\boldsymbol{Y}}_k$ and c_k are initialized to 0. The estimation of $f(\vec{\boldsymbol{x}}_k)$ is then given by $\hat{f}(\vec{\boldsymbol{x}}_k) = \vec{\boldsymbol{Y}}_k$.

One of the most important features of the SNN network is that the network output depends linear of the parameters in the network. This means that it is possible, given some learning data, to find the optimal parameters in a least squared sense. See fx [2] "LS-learning". If $\vec{\boldsymbol{y}}_d = \sum \cdots \sum f(\vec{\boldsymbol{x}}_k) \cdot g(\vec{\boldsymbol{x}} - \vec{\boldsymbol{x}}_k) + \vec{\boldsymbol{v}}$ where $\vec{\boldsymbol{v}}$ is a vector of white noise elements, then the LS-learning method will find the right parameters $\hat{f}(\vec{\boldsymbol{x}}_k) = f(\vec{\boldsymbol{x}}_k)$. If $\vec{\boldsymbol{v}}$ is not white noise, then the LS-learning method won't find the right parameters. But at least it will find a set of parameters, which gives the best fit to the learning-data in a least squared sense.

4 Results and Discussion

The test system is given by:

$$y = \exp(x) + \sin(3\pi x) + v \tag{4}$$

where v is white noise with the variance of 0.1;

4.1 IM-learning

The test of IM-learning is based on five periods of the interval $x = [0, 1[$ or one and a half period of the sinus. If the system is a pure sin-function then Shannons theorem states, that three samples in each period are enough. But because of the non ideal conditions it is chosen to use six samples $\Delta x = 1/6$ in the interval from $[0, 1[$ or 30 samples over the whole sequence.

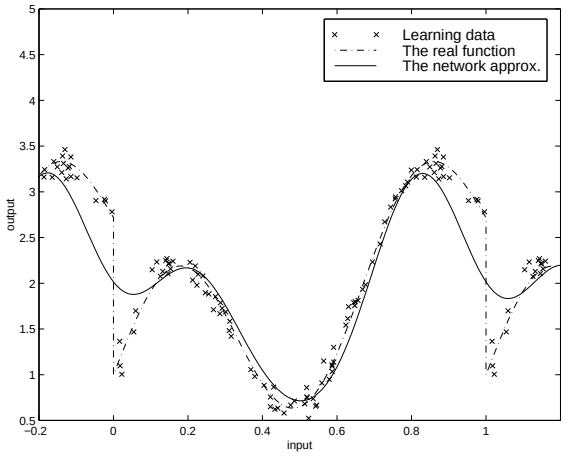

Figure 4: Estimations result based on IM-learning.

Figure 4 shows, that the network output in the area between 0.2 to 0.8 gives a good approximation to the system. The reason for this is, that the system especially in this area, is rather low frequency.

In the area about input equ. zero and one there is a rather large difference between the system and the network output. In this area there is a jump in the output or in other words a very high frequency. This big difference is due to the fact, that the network produce a low pass version of the signal, this is of course what it is meant to do.

4.2 LS-learning

LS-learning is also based on five periods of the interval $x = [0, 1[$. Figure 5 shows the result of this learning. If the estimated parameters dosn't have to be used as estimates of the characteristic samples, then it is not necessary to use more than one period of the signal. This is very important because, when using only one period, it is possible to reduce the number of neurons in the SinC-layer, and thereby drastically reduce the number of neurons in the Cartesian-layer.

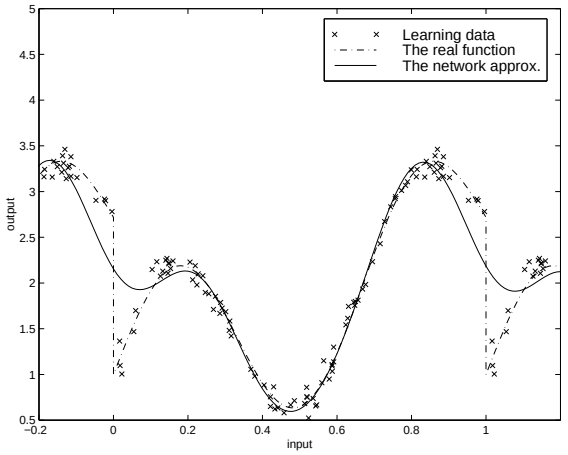

Figure 5: Estimations result based on LS-learning .

Figure 5 shows, that LS-learning has the same quality as IM-learning, but LS-learning has a tendency to follow the signal better than IM-learning.

5 Conclusion

The SNN-network has been described. This network is based on Shannons samplings theorem. This means, that the network is based on SinC neuron-functions. Two learning algorithms are described. The first one is called IM-learning. IM-learning is based on a simple calculation of the mean output in each input interval, given by the characteristic samples. This learning method produces acceptable results if five periods of the original learning-data are used. The other learning method is based on optimal parameter estimation in a least squared sense. This is possible because the network output depends linear of the parameters in the network. This method is called LS-learning. LS-learning gives better results than IM-learning. An other advantage of LS-learning is, that this learning method does not require that the learning is based on a number of periods of the original signal.

References

[1] K. J. Hunt, G. R. Irwin, and K. Warwick. *Neural Network Engineering in Dynamic Control Systems.* Springer-Verlag, first edition, 1995.

[2] Lennart Ljung and Torsten Söderström. *Theory and Practice of Recursive Identification.* The MIT Press, first edition, 1983.

[3] Ole Sørensen. *Neural Networks in Control Applications.* PhD thesis, Aalborg University, June 1994.

[4] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, pages 37,10–21, 1949.