ハイパフォーマンス
コンピューティング 78－3
（1999. 10. 7）

Forecasting and precision on using multi-layer neural network

Tomoo AOYAMA and Hanxi ZHU

The Faculty of Engineering, Miyazaki University
Gakuen Kibanadai-nishi 1-1, Miyazaki 889-2192, Japan
E-mail: aoyama@esl.miyazaki-u.ac.jp

Abstract

We discuss a general method to predict phenomena. The method is based on functions of multi layer neural networks and recurrent representations of periodic functions. The method can evaluate precisions for the predictions. The principle is due to the differentials of the neural networks.

1. Introduction

Many studies of the extrapolation for time dependent phenomena have been published. In the studies, there were some methods that are adopted neural networks [1,2]. Using neural networks, explicit descriptions (functions) are not necessary. Futures of phenomena are predicted from observed data only. It is very useful and practical, but the usefulness is a defect at same time. If explicit functions are unknown, it is hard to calculate precisions of extrapolations. We often find examples of prediction in published papers [3], but hardly discover examinations using neural networks. We consider that the neural network is one of functions. The function is defined at learning data, and it is discrete essentially. But we use it as continuous, and make predictions. It is unreasonable. If the neural network is a continuous function, its differential should be defined. Where we don't consider fractal functions. The explicit differential representation is published [3]. Then, function characters near discrete points of learning data are evaluated by the differential. The fact makes a discrete function into continuous; at least, an index for predictions is got.

2. Multi layer neural networks

We used a neural network without feedback loop, whose structure is three layers. In the layer structure, informed propagations are following.

$\{x1,x2,....xn\} \equiv x$,

$yj=\Sigma Vjixi$, $pj=f(yj)$,

where a vector $\{x1,x2,....xn\}$ has an individual information, and its suffix corresponds to numbered neurons in the first layer. The neurons don't include bias ones. [We substitute the bias neurons for threshold-values of actions of neurons.] Suffices "i" and "j" are used for 1st and 2nd layer, respectively. Vij is a matrix of connection weights between neurons in 1st and 2nd layer. The "yj" is a variable as temporary defined. And f() is a function simulated a neuron, which must be a differential function. If it is not differential, following learning equations are not defined. We name the function as neuron-function. The "pj" is a vector for output of neurons in 2nd layer. Thus, we got following relations for informed propagations between 2nd and 3rd layers.

$\{p1,p2,....,pm\} \equiv p$,

$zk=\Sigma Wjkpj$, $ok=g(zk)$,

$\{o1,o2,...ok\} \equiv o$.

Where Wjk is a matrix of connection weights, and g() is a neuron-function. Suffix "k" is used for 3rd layer, so a vector $\{o1,o2,...ok\}$ is output for neurons in 3rd layer. In the neural network, an individual information $\{x1,x2,....xn\}$ is transformed into a vector $\{o1,o2,...ok\}$. Where dimension of the vector is converted into different one, therefore, we must take care of the transformations in cases of "n < k" and large "m"-values.

These relations are realized in one individual input/output datum, and the relations also stand up in case of plural data. Thus, we write as following.

$x \rightarrow \{x\ \mu\}$, $p \rightarrow \{p\ \mu\}$, $o \rightarrow \{o\ \mu\}$

In the multi layer neural network, a

corresponding relation is organized.

$\{x\ \mu\} \Leftrightarrow \{o\ \mu\}$

The relation is not a one-to-one correspondence. Followings are allowed.

$\{o\ \mu\} = \{o\ \nu\} = \ldots\ldots = \{o\ \xi\}$

But, next relations are not done.

$\{x\ \mu\} \neq \{x\ \nu\} \neq \{x\ \xi\}$

We arrived at following differential coefficients between output and input.

For 2-layer-network:

$\delta yj = Wij * \delta xi$

$\delta oj = f(\delta yj) = f'(yj) * Wij * \delta xi/1!$

$+f''(yj)\{Wij * \delta xi\}^{**}2/2! + \ldots\ldots$

Then,

$\delta oj/\delta xi = f'(yj) * Wij$

$+ f''(yj) * Wij^{**}2 * \delta xi/2 + \ldots$

For 3-layer-network:

$\delta yj = Vij * \delta xi$

$\delta pj = f(\delta yj) = f'(yj) * Vij * \delta xi/1!$

$+f''(yj) * (Vij * \delta xi)^{**}2/2!$

$\delta zk = Wjk * \delta pj$

$\delta ok = g(\delta zk) = g'(zk) * Wjk * \delta pj/1!$

$+g''(zk) * (Wjk * \delta pj)^{**}2/2!$

$\delta ok = g'(zk) * \Sigma j[Wjk * \{f'(yj) * Vij$

$+f''(yj) * Vij^{**}2 * \delta xi/2\}] * \delta xi$

$+(1/2) * g''(zk)$

$* \Sigma j[Wjk^{**}2 * \delta pj * \{f'(yj) * Vij$

$+f''(yj) * Vij^{**}2 * dxi/2\}] * dxi$

Consequently,

$\partial ok/\partial xi = g'(zk) * \Sigma j[Wjk * f'(yj) * Vij]$

$+(1/2) * g''(zk) *$

$\Sigma j[Wjk^{**}2 * f'(yj)^{**}2 * Vij^{**}2] * \delta xi + \ldots$

in ANN [4], absolutely $g''0=0$, then:

$\partial ok/\partial xi = \Sigma j[Wjk * g'(qj) * Vij * f'(yj)]$

$= \partial\ o/\partial\ x$ [this is a vector representation.]

For 3-layer multi infinitesimals:

$\delta yj = \Sigma i[Vij * \delta xi]$

$\delta pj = f(\delta yj) = f'(yj) * \Sigma i[Vij * \delta xi]$

$+\{f''(yj)/2\} * \{\Sigma i[Vij * \delta xi]\}^{**}2$

$\delta zk = \Sigma j[Wjk * \delta pj]$

Therefore,

$\delta ok = g(\delta zk)$

$= g'(zk) * \Sigma j[Wjk * \delta pj]$

$+\{g''(zk)/2\} * \{\Sigma j[Wjk * \delta pj]\}^{**}2$

In ANN or 1st. order approximation, absolutely $g''0=0$, then:

$\delta ok = g'(zk) * \Sigma j[Wjk * f'(yj) * \Sigma i[Vij * \delta xi]]$

Where f' and g' are differentials for f and g functions.

The neuron-functions f and g are sigmoid functions generally. But it is desirable that the functions are replaced by other ones. We often replace g-function with linear function. Such neural network is called ANN, which is excellent to extrapolate on various phenomena.

3. Recurrent representation for functions

3.1 Takens's embedding theorem

Takens' embedding theorem teaches us possibility for the short-range prediction of time series data gotten by chaotic phenomena. The embedding theorem shows that the trace described by multi dimensional variables is calculated by one kind of variable in them. It is impressive and extensions of the theory are broad and general. Principles of the theory are based on the Jacobian matrix defined nearby the last point. It is possible that the principle is extended by using facilities of neural networks. The neural networks evaluate status around learning points from all data, and they are not local but global. Therefore, we believe a significance in investigations of the prediction use of neural networks.

3.2 recurrent representations for functions

We investigate a recurrent representation of functions in order to eliminate periodic conditions for functions. The representation is gotten from following facts,

(1) sampling values of function are finite vector {x1,x2,....,xn}.

(2) a set of partial vector of the vector is written as {{x1,x2,...xk},{x2,x3,...xk+1},.....}, where an index k is less than n. "k-1" is called a window width. The index k has not meaning of neurons in third layer here. The width is determined by following index E,

$Dij = \{\Sigma l[(xi+l - xj+l)^{**}2]\} * 0.5,$

$E = \min\{Dij;\ i>j\}.$

We used a condition that E is bigger than 0.02. The condition was got empirically.

If the condition is negligible, the back propagation learning will be very difficult. If the learning may be completed, the generating potential plane is steep, and

the steepness connects to the over learning for the neural networks.

(3)elements of the set can be corresponded to a scalar series {xk+1,xk+2,...} as following: {{x1,x2,...xk}<-->xk+1,
{x2,x3,...xk+1}<-->xk+2,....}
They are set of partial vector and scalar series, and fragmentations of functions, which are learning data set for a neural network simultaneously. A property of neural networks combines a vector with another, which includes a scalar. That is {xi,xi+1,...xi+k-1}-->xi+k, where i=1,2,....
The relation is equivalent to a local prediction or fragmented forecasting.
If the last partial vector is written as {xj,xj+1,...xj+k-1}, and the corresponding scalar is xj+k. In that case, what kind of character does the vector {xj+1,xj+2,...xj+k} have ? The {xj+1,xj+2,...xj+k} is a new undefined vector, but it is also an input datum for neural network. Therefore, a new scalar is calculated from the input datum. Is the scalar written as xj+k+1?
In generally, it cannot be allowed. But if the sampled function is periodic and the new vector is included in learning data set, it will be allowed. In such a situation, the calculated scalar is xj+k+1, and a new vector {xj+2,xj+3,...xj+k+1} is also used for advanced prediction. These iterative series describe a periodic function implicitly. We call the iteration as a recurrent representation for the function. When the representation is valid, a multi layer neural network is nearly equal to the sampled function.

4. Precision for recurrent representation

In neural networks, input data are multi dimensional. In order to simplify following discussion, we put the multi dimensional data on a multi dimensional space. From now on, input data are points on the space, so we write learning data as learning points.

4.1 Fractal dimensions

We found an examination for the prediction by using neural networks. It is based on the fractal Brown's function that is a statistical extension of the fractal dimensions.

FB(y)
=probability[|1/dx|**H・{f(x+dx)-f(x)}<y]
The function "FB()"is a distribution function, and is determined by observed data. Where a variable "H" is related with the fractal dimension, whose details are listed in [1]. The "H" variable is an index that determines reliability for the prediction. It is effective to economic forecasting. However, the index shows linear responses between the linear and random changes. We are sure that the responses are not appropriate for precision index. So we consider a new non- linear index.

4.2 Euclid distance

When sigmoid functions are adopted in neural networks, output responses of the neural network near learning points are roughly equal to that of the points. And Euclid difference increases monotonously with distances from the learning points. The fact suggests implicitly that precision for the recurrent representation can be estimated by the distance. But the distance would not be quantitative index, because of the monotonous increasing only. We cannot discover quantitative variance around the learning points. We are sure that Euclid distance is 0th order approximation for the precision. The explicit formula "d" for the 0th approximation is,
d=min{D1,D2,....DN}.
Where each "Di" is,
Di={Σj[(xj-xi)**2+(xj+1-xi+1)**2+....
+(xj+k-1 - xi+k-1)**2]}**0.5,
and {xi,xi+1,...xi+k-1} (i=1,2,..,N) is the learning data, and {xj,xj+1,.....,xj+k-1} (j=1,2,...,N) is an input point for prediction.

4.3 Differential distance

Revising the defect for the variance around the learning points, we use differential coefficients for neural network. By using analogical derivations for Taylor's expansion, we get following a scheme,
{δj,..., δj+k-1}
={(xj-xi),(xi+1-xj+1),..,(xj+k-1 - xi+k-1)},
where index "i" means the nearest point

from the input point. Vector representation is,

$\delta = x' - x$,

$\Delta = (\partial o / \partial x) \cdot \delta / 1!$
$+(\partial^{**}2\, o / \partial x\, \partial x) \cdot [\delta \times \delta]/2! + \ldots.$,

where symbol "×" means outer (tensor) products for the vector. The Δ has not a character for the distance. So we transform it into,

$\Delta' = \{\Sigma j[((\partial oi / \partial xj)*(xj-xi))^{**}2$
$+((\partial oi+1 / \partial xj+1)*(xj+1-xi+1))^{**}2$
$+\ldots..$
$+((\partial oi+k-1 / \partial xj+k-1)*(xj+k-1-xi+k-1))^{**}2]$
$\}^{**}0.5$.

The scalar "Δ'" corresponds with Euclid distance. This scheme is valid on the learning points, but is invalid on far from the points. Such situations will be found during calculations to predict. So we are sure the index Δ' is 1st order approximation.

Or, in a case of ANN, we use first order approximation,

$\delta ok = g'(zk)^* \Sigma j[Wjk^* f'(yj)^* \Sigma i[Vij^* \delta xi]]$,
$\Delta'' = \{\Sigma k[\delta ok^{**}2]\}^{**}0.5$.

4.4 Invalid for prediction

We defined the nearest point from an input point for prediction, whose index was "i". Similarly, the index of the nearest point for next prediction is defined, and it is "i'" here. The location of the index "i'" should be onward from "i", or be equal to "i". We write the condition as "i' >= i". In learning process of the neural network, such a sequent is not explicitly. However, when we consider the recurrent representation for the function, the sequent is included in the consideration implicitly. If the sequent property is break, it is sure that a new aspect is arisen in the prediction. The prediction includes new information that is not found in learning data. Therefore we should reject the prediction.

In ANN only, the output is out of range [0,1] because of linear "g"-function that is a neuron function in the 3rd layer. If the case happens, it shows that the prediction contravenes the definitions for the neural network. Then, we should regard the prediction as invalid.

4.5 Two conceptions for prediction

When we forecast phenomena by using neural networks, at first we construct a neural network by learning equations. The forecasting is evaluated based on the constructed network; at the time there are two choices for adopting prediction data.

One is used last observed data just before the forecasting. The conception is called as "one step ahead prediction" or "short-range prediction". It is similar to Takens' embedding theory. But, in the prediction principle, a basic quantity is not corresponded with local information such as the Jacobian. Moreover, the quantity is not gotten by the last data, but information from rather past data. Then, it is somewhat extension for Takens'.

The other is used output data calculated only by the neural network. The conception is called as "long range prediction". It is out-of-range of Takens' embedding theory, whose efficiency is only examined by numerical calculations. It is natural that the forecasting from the former is higher precision than that of later.

5. Numerical calculations

5.1 Lorenz' chaos

A definition of the chaos is,

$xi+1 = xi^*(1-xi)^*A$,

where A is a constant (3.66667), and x0 is an initial value (0.5). These values should be selected reasonably. If not, the iteration does not generate chaos.

We calculated two sets {x0,x1,...,x39} and {x40,x41,...,x159}. We used the former set for learning of a neural network, and the later for examinations of forecasting.

In learning, the window width is set as 8, then the number of effective learning data is 40-8=32. The results were listed in figure 1. In the figure, the left side of a dotted vertical line means a learning term. The right side is forecasting. On the upper part in the figure, a solid curve means forecasting, and a dotted curve does true values of the chaos. On the lower part, a solid line means the difference between

the forecasting and the true values.

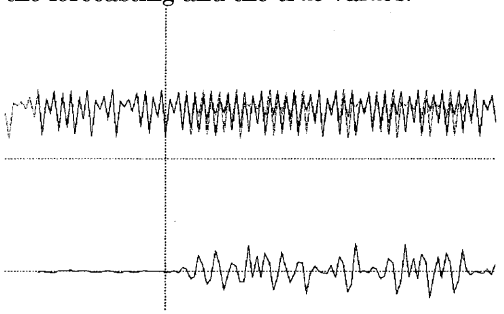

Figure 1. Forecasting and error for Lorenz' chaos

The difference is negligible in the learning term, however it is not in the forecasting. That means the learning has completed, but the forecasting can't be done. We must pay attention to the conclusion is got under knowledge of true values of the chaos. The Euclid and differential distances are listed on solid lines in figure 2. These distances show that the forecasting is uncertain.

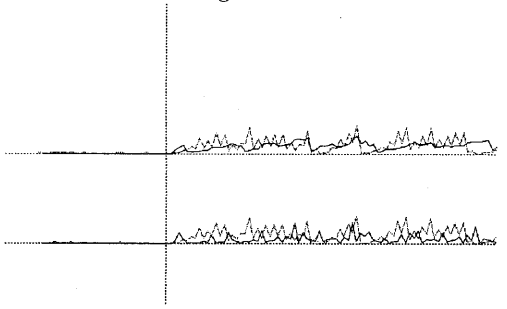

Figure 2. Euclid and differential distances

Long-range predictions for chaos are very difficult. On the other short-range, we got high accuracy forecasting, that is in figure 3.

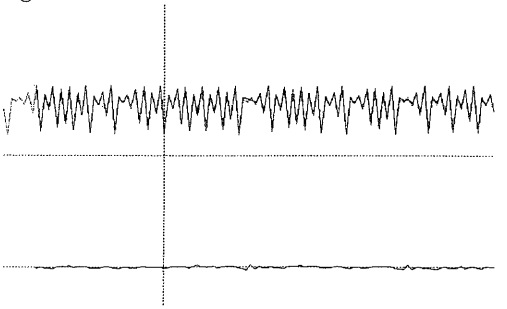

Figure 3. Forecasting and error for Lorenz' chaos on short range prediction

It is clear that the forecasting is accurate and the differentials are negligible. In the case, the Euclid and differential distances are listed in figure 4.

These distances show that the forecasting is accurate considerably.

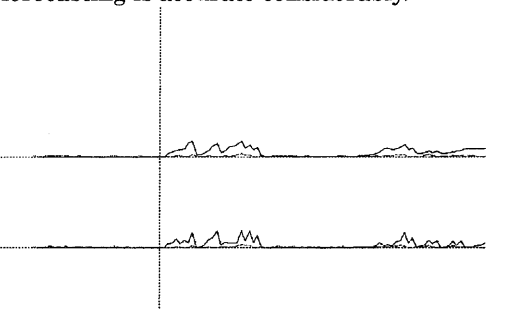

Figure 4. Euclid and differential distances on short-range prediction

6. Conclusion

We investigated differential coefficients for multi neural networks, and derived indexes to forecast. We discussed recurrent representations of periodic functions, and slightly extended them, which were used in the processing of neural networks. We tested these processing techniques on the forecasting for Lorenz's chaos. As the results, we believe that the indexes and extensions are useful in order to predict nearly periodic phenomena.

references

[1] I. MATSUBA, "Chaos and Forecasting", Mathematical Science, pp.64(Science Co. Ltd., Tokyo, 1992.6), Japanese.

[2] Mitsue ONODERA, U.NAGASHIMA, H.YOSHIDA, T.AOYAMA, and H.HOSOYA, "Neural Network Reproduction of Time Series Data with Varying Amplitudes and Frequencies", J. Chemical Software, Vol.4, pp.57(1998), Japanese.

[3] Tomoo AOYAMA, Yoshimi ISU, and U.NAGASHIMA, "Extrapolations by using neural-networks and a recurrent representations of functions", IPSJ SIG Notes, 95-HPC-59(1995.12.11), Japanese.

[4] We don't know certainly when "Analogue type Neural Network (ANN)" was used. We adopted it in paper; T.AOYAMA, Y.SUZUKI, and H.ICHIKAWA, "Neural Networks to quantitative structure-activity-relationships", J.Med.Chem, 33, 2583(1990).