An Optimized Process Neural Network Model*  

Guojie Song¹, ³, Dongqing Yang¹, Yunfeng Liu², Bin Cui¹, Ling Wu¹, and Kunqing Xie³

¹ School of Electronic Engineering and Computer Science, Peking University, Beijing, China
² Computer Center of Peking University, Beijing
³ National Laboratory on Machine Perception, Peking University, Beijing

Abstract. In this paper, we proposed an optimized process neural network based on fourier orthogonal base function, which can deal with both static value and time-varied continuous value simultaneously. To further improve its performance, we optimize the network topological structure, which adopts fourier expansion based preprocessing. Experiments based on the real datasets show that our proposed churn prediction method has better maneuverability and performance. Most important of all, our method has been used in real applications in China Mobile which is the major telecommunication company of the world.

1 Introduction

Many techniques have emerged for the purpose of classification, such as artificial neural networks, SVM and classification trees [1,2,5,6]. Unfortunately, inputs of all these methods are static values. However, real applications also include many time-varied continuous values. Such time continuous values are always being summarized firstly by using techniques, such as sum, average etc, and then taken as input in static value by existing methods. However, many useful information for churn prediction are contained in such time-varied continuous values, but which will be removed with the execution of above preprocessing unfortunately.

To solve this problem, He and Liang proposed artificial process neuron model [3] in 2000. From a point view of architecture, process neuron is similar to conventional artificial neuron, which can be considered as a general form of traditional neuron model. The major difference is that the input, the output and the corresponding connection weight of process neuron are not static values but time-varied functions, and capable of imitating time-varied continuous value perfectly. Process neural network (PNN) is composed of densely interconnected process neurons. A particularly important element of designing process neuron is the choice of base function, which can influence its performance greatly. In general, the characteristics of PNN is its high prediction accuracy and express ability. But the efficiency of PNN is always concerned by users.

In this paper, an optimized process neural network, named MPNN, has been proposed, which can deal with both traditional static data and the time-varied continuous

* This work is supported by the National Natural Science Foundation of China under Grant No. 60473051 and No.60642004 and IBM and HP Joint Research Project.
data simultaneously. Fourier orthogonal base function has been chosen as the base function of process neuron to expand each time varied continuous series efficiently. To avoid repeating computation of fourier base function expansion and time aggregation operation, an optimized MPNN has been proposed, which use the preprocessing technique based on fourier transform to simplify the structure of MPNN. The effectiveness and efficiency of MPNN have also been proved by our extensively experiments based on the real dataset. The system with the proposed churn prediction model has been implemented, and used in China Mobile Communications.

The remaining of the paper is organized as follows. In Section 2, we presents the proposed MPNN model. A performance study of the proposed method is demonstrated in Section 3, and finally we conclude our study in Section 4.

2 MPNN: A Mixed PNN Model

2.1 Topological Structure of MPNN

The MPNN proposed is composed of four layers: input layer, two hidden layers and one output layer. Input layer is composed of \( n + \sum_{i=1}^{m} A_i \) units, which includes both continuous time-varied data, such as billing data, and discrete data, such as customer gender etc. Thus, the input of MPNN include \( n \) input node dealing with continuous time series data and \( \sum_{i=1}^{m} A_i \) nodes for discrete static input data, where each input node \( d_{i,A_i} \) corresponds to one value of property \( A_i \), with 1 if \( \text{value}(A_i) = d_{i,A_i} \), else with 0. The first hidden layer is composed of \( n \) process neurons and \( m \) traditional neurons. Process neurons deal with \( n \) time continuous input data and traditional neurons only accept \( \sum_{i=1}^{m} A_i \) discrete static data from the input layer. The second hidden layer is composed of \( p \) traditional neurons and the last layer is output layer. To reduce the complexity, we will only consider the case of one output unit.

2.2 Relationship Between Input and Output

Input: The inputs of the first layer can be expressed as two parts: \( X(t) \) for time varied continuous time series and \( D \) for discrete static input, denoted as \( X(t) = (x_1(t), x_2(t), ..., x_n(t))D = (d_{1,1}, ..., d_{1,A_1}, ..., d_{m,1}, ..., d_{m,A_m}) \).

The outputs of the first hidden layer: We first consider the computation of \( j \)-th process neuron connected with continuous input \( X(t) \).

\[
y_{j}^{(c,1)} = f_p(\sum_{i=1}^{n} \int_{0}^{T} w_{ci,j}(t)x_i(t)dt),
\]

where \( w_{ci,j}(t) \) is the link weight function between \( j \)-th process neuron in the first hidden layer and the \( i \)-th unit of \( X(t) \) in the input layer (\( i, j \in [1, n] \)).

If we take fourier base function \( s_i(t) \), \( i \in [1, L] \), as process base function to expand process weight function and input \( x_i(t) \), formula [2.2] will be

\[
y_{j}^{(c,1)} = f_p(\sum_{i=1}^{n} \int_{0}^{T} \left( \sum_{q=1}^{L} \sum_{z=1}^{L} c_{iz}(z)w_{c_{iq}}^{(q)}s_{q}(t)s_{z}(t) \right)dt)
\]

\[
= f_p(\sum_{i=1}^{n} \sum_{q=1}^{L} \sum_{z=1}^{L} c_{iz}(z)w_{c_{iq}}^{(q)} \int_{0}^{T} s_{q}(t)s_{z}(t)dt)
\]

(1)
Thus, the outputs in the first hidden layer can be expressed as

\[
y_j^{(c,1)} = \sum_{i=1}^{n} \sum_{q=1,z=1}^{L} c_{iz}^{(z)} w_{ciq}^{(q)}
\]

(2)

where \(f_p\) is the activation function of the process neurons in the first hidden layer.

For traditional neurons in the first layer, its outputs can be expressed as \(y_k^{(d,1)} = f_d(\sum_{h=1}^{A_k} d_{kh} w_{dhk})\), where \(w_{dhk}\) is the link weight between \(k\)-th neuron in the first hidden layer and the \(h\)-th value of property \(d_k\) in the discrete input layer \((k \in [1, m], h \in [1, A_k])\). Thus, the outputs of the first hidden layer is \(\text{out}^{(1)}_{r} = y_j^{(c,1)} + y_h^{(d,1)}\), where \(r \in [1, m + n]\).

The outputs of the second hidden layer: Based on the outputs of the first hidden layer, the outputs of the second hidden layer can be expressed as \(y_i^{(2)} = f_o(\sum_{r=1}^{m+n} \text{out}^{(1)}_{r} v_{rl})\), where \(v_{rl}\) is the link weight between the \(l\)-th neuron in the second hidden layer and the \(r\)-th neuron in the first hidden layer. \(f_o\) is the activation function in the neuron in the second hidden layer.

Output: The output function of the MPNN can be expressed as \(y(t) = f_o(\sum_{l=1}^{p} y_i^{(2)} u_l)\), where \(u_l\) is the link weight between the \(l\)-th neuron in the second hidden layer and the output node. \(f_o\) is the activation function of the output node.

### 2.3 Learning Algorithm

Assume that we have \(K\) learning sample functions:

\[
\begin{bmatrix}
x_{11}(t), & x_{12}(t), & \ldots, & x_{1n}(t), & y_1(t) \\
x_{21}(t), & x_{22}(t), & \ldots, & x_{2n}(t), & y_2(t) \\
& \vdots & & \vdots & \\
x_{K1}(t), & x_{K2}(t), & \ldots, & x_{Kn}(t), & y_K(t)
\end{bmatrix}
\]

where the first suffix \(i\) in \(x_{ij}(t)\) denotes the serial number of learning sample, and the second suffix \(j\) denotes the serial number of component in input function vector. \(y_k(t)\) is the expected output function for input \(x_{k1}(t), x_{k2}(t), \ldots, x_{kn}(t), (k \in [1, K])\).

Suppose that \(\tilde{y}(t)\) is the desired output function, and \(y(t)\) is the corresponding actual output function of the MPNN, then the mean square error of the MPNN output can be written as

\[
E = \frac{1}{2} \sum_{b=1}^{K} (y_b(t) - \tilde{y}_b(t))^2 = \frac{1}{2} \sum_{b=1}^{K} \left[ f_o \left( \sum_{l=1}^{p} y_i^{(2)} u_l \right) - \tilde{y}_b(t) \right]^2
\]

\[
= \frac{1}{2} \sum_{b=1}^{K} f_o \left( \sum_{l=1}^{p} f_d \left( \sum_{r=1}^{m+n} \sum_{q=1,z=1}^{L} c_{izb}^{(z)} w_{ciq}^{(q)} + \sum_{h=1}^{A_k} d_{khb} w_{dhk} \right) v_{rl} \right) u_l - \tilde{y}_b(t) \right]^2
\]

For analysis convenience, \(Z_b, Q_b, P_b\) and \(H_b\) are defined respectively as

\[
H_b = \sum_{h=1}^{A_k} d_{khb} w_{dhk}, \quad P_b = \sum_{i=1}^{n} \sum_{q=1,z=1}^{L} c_{izb}^{(z)} w_{ciq}^{(q)}
\]

\[
Q_b = \sum_{r=1}^{m+n} (f_d(H_b) + f_p(P_b)) v_{rl}, \quad Z_b = \sum_{l=1}^{p} Z_b u_l
\]
According to the gradient descent method, the learning rules are defined as follows

\[
wc^{(q)}_i = wc^{(q)}_i + \alpha \Delta, \quad wc^{(q)}_i = wc^{(q)}_i + \beta \Delta w_{d,h,k}, \quad v_{r,l} = v_{r,l} + \gamma \Delta v_{r,l}, \quad u_{l} = u_{l} + \eta \Delta u_{l}, \quad \text{where} \quad \alpha, \beta, \gamma, \eta \quad \text{are the learning rate, and} \quad i \in [1, n], \quad k \in [1, m], \quad h \in [1, A_k], \quad l \in [1, p], \quad q, r \in [1, L].
\]

\[
\Delta wc^{(q)}_i = -\frac{\partial E}{\partial wc^{(q)}_i}, \quad \Delta w_{d,h,k} = -\frac{\partial E}{\partial w_{d,h,k}}, \quad \Delta v_{r,l} = -\frac{\partial E}{\partial v_{r,l}} = -\Delta u_{l} = -\frac{\partial E}{\partial u_{l}} = -\Omega, \quad \text{where} \quad \Omega = \sum_{b=1}^{K}(f_o(Z_b) - \tilde{y}_b(t))f'_o(Z_b).
\]

In this paper, all activation functions have been substituted by Sigmoid function, i.e. \( f_o(u) = f_g(u) = f_p(u) = f_d(u) = \frac{1}{1+e^u} \), with \( f'_o(u) = f_s(u)(1 - f_s(u)) \).

### 2.4 Topological Structure

Based on formula 2 if we take fourier base function as the base function of process neuron in MPNN, each input time varies function \( x_i(t) \) can be expanded with a set of fourier coefficient \( c_i \) and corresponding weight \( w_i \). If we take \( x_i(t) \) as input of process neuron in MPNN during each time iteration of training process, it should be expanded one time by using DFT accompanied with one time aggregation operation by using formula \( \int_T^0 s_q(t)s_z(t)dt \). In fact, such costs of the fourier expansion can be avoided if each input time series \( x_i(t), i \in [1, n] \), has been preprocessed before it is input into MPNN. Because \( c_i \) is a constant and has no relationship with the training process, so taking fourier coefficient \( c_i \) as input will not influence the final results. By using such preprocessing strategy, the process neuron in MPNN have become a traditional neuron by losing its time aggregation ability.

### 3 Data and Experimental Results

#### 3.1 Characteristics of Input Data

We get the real data from the China Mobile Communication Company for Churn prediction. We sample the dataset from Jan. 2004 to April 2004. After filtering the data with missing values, we select 220 thousands samples. 2000 samples have been selected randomly for training data set and 10000 samples for testing data set. The ratio of churner is 20%. The description of the variables for this research is presented as follows.

**Time varied continuous data:** It includes three kinds of usage series data: call time, the number of messages and the number of different telephones communicated with him/her). Each data element in series is accumulated in term of day, and 91 element spanning three months have been generated in each series.

**Traditional static discrete data:** Two discrete variables have been selected, which are customer’s gender(male:0, female:1) and age (being discreted into five segments, (0-20), (20-30), (30, 40), (40, 60) and (60-100) in advance).
Mark of churn: According to the definition of churn management strategies, different in each province, the mark of churn for each customer can be defined with whether he is churn in (churn:1, nonchurn:0), that is in June 2004.

3.2 Experimental Results

All experiments are conducted on a PC with Pentium IV 1.4G CPU and 512MB main memory, running Windows XP operation system. The MPNN code was written in C++. Some existing methods, e.g. PNN, ANN [6] and Decision Tree(C4.5)[5] have been compared. The neural networks used in our experiments are multilayer perceptrons with a single hidden layer which contains 10 nodes and they were trained by the back propagation algorithm with the learning rate 0.3 and the momentum term 0.7. The triangle function has been chosen as base function of PNN, and the input is three kinds of time continuous data. Continuous data have been averaged before input ANN and C4.5. We compared their performance about predicting accuracy, lift value and execution time. The model predicts correctly with churn if the predicted user’s churn probability is bigger than (or equal to) 0.5 or the user doesn’t churn with the probability less than 0.5. Otherwise the Model predicts wrong.

Precision measurement: In this section, two basic measures, precision and recall, have been introduced to evaluate our prediction methods. The experimental results can be seen in Table 1. It can be observed that our method MPNN has the best performance in detecting the churn with precision 87.5% and recall 81.5%. C4.5 and ANN is less than PNN no matter for precision or recall.

Lift value measurement: We applied MPNN to the training dataset to predict the churn or no churn of the subscribers in the testing dataset. In the telecommunications industry, the churn and no churn prediction is usually expressed as a lift curve. The lift curve plots the fraction of all churners having churn probability above the threshold against the fraction of all subscribers having churn probability above the threshold.

The lift curves are shown in Figure 1(a). As described in figure, when compared with C4.5 and ANN, MPNN identified more churners than them under the same fraction of subscribers. It is important to note that PNN also identified more churners than C4.5 and ANN. When compared with PNN, MPNN identified more churners than PNN did.

To better compare the performance of these models, let us consider the lift factor, which is defined as the ratio of the fraction of churners identified and the fraction of subscribers contacted. It is important to note that the lift factor for the random churn predictor is 1. Owing to the limited number of staff in the carrier’s customer services center, it can only contact 5% of all subscribers. The lift factors for these models were contacted under different fraction of subscribers are shown in Figure 1(b). Again, MPNN obtained
higher lift factors than PNN, which in turn obtained higher lift factors than ANN and C4.5 when the first 20% subscriber are contacted.

**Computation efficiency measurement:** To evaluate their computation efficiencies, Table 2 shows the execution times for MPNN, PNN, C4.5 and ANN under different sample data size.

The experimental results showed that MPNN accomplished the churn prediction task almost the same as ANN, since fourier base function can deal with continuous data in linear time, and make MPNN with the same function as ANN. Of the four approaches, C4.5 required the least execution time to complete since C4.5 used less number of iterations than the rest three models. However, C4.5 is unable to produce churn prediction as accurate as others (as described in Table 1, Figures 1). Thus, our MPNN model not only achieved higher accuracy but also with less execution time.

### 4 Conclusion

In this paper, we proposed an optimized process neural network model, and its performance has been investigated. The result shows that the classification accuracy of MPNN, 87.15%, is better than of ANN, Decision Tree (C4.5).

### References