Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 7, No. 2, 2024

A Sigma-Pi-Sigma Neural Network Model with Graph Regularity Term

Qianru Huang *, Qingmei Dong, Yunlong Liu, Deqing Ji, Qinwei Fan

Xi'an Polytechnic University, Xi'an, Shaanxi, 710600, China

*** Corresponding author:** Qianru Huang

Abstract: In recent years, Sigma-Pi-Sigma neural network (SPSNN) as a special kind of higher-order neural network has attracted wide attention for its fast convergence speed and good approximation ability. However, an inappropriate number of hidden layer neurons may also lead to model underfitting or overfitting, which affects the performance and generalization ability of the model. Therefore, we propose a Sigma-Pi-Sigma neural network with graph regularity by adding a graph regularity term to the network. The results show that the proposed algorithm performs well in terms of training accuracy, testing accuracy and efficiency.

Keywords: Sigma-Pi-Sigma Neural Network; Graph Regularization; Squared Error.

1. Introduction

In recent years, neural networks have become one of the focuses of today's science and technology field and have received widespread attention due to their powerful learning ability and wide application prospects. Sigma-Pi-Sigma neural network (SPSNN) is a kind of feed-forward neural network [1], which consists of different orders of Sigma-Pi neural networks (PSNs), and is a high-order neural network with a multiplicative unit and stronger nonlinear mapping ability, with its fast convergence speed and good approximation ability. It is a higher order neural network with product unit and stronger nonlinear mapping ability, which has its fast convergence speed and good approximation ability [2]. Due to its strong nonlinear mapping ability, this network has been widely used in many fields such as medical and industrial [3-4].

Generally, we use gradient descent method to train the network [5]. However, when we use the squared error sum function to train the SPSNN, the inappropriate number of hidden layer neurons and the magnitude of the weights may lead to underfitting or overfitting of the model, which affects the performance and generalization ability of the model. Therefore, the structural design of BP neural networks and the acquisition of optimal parameters remain a great challenge.

For the time being, adding a regularization term to the error function is the most common way to solve this problem. The main common penalties we use to train networks are weight decay, weight elimination and approximate smoothing [6-8]. While the traditional conventional regularization term is very effective in sparsifying and preventing overfitting, it does not take into account the geometric structure of the original data [9]. Recently popular graph regularization terms perform well in this regard [10-11]. Therefore, in this paper, we consider introducing the graph regularization term in the SPSNN model to obtain the optimal parameters and the best network structure by applying the graph regularization term to the error function.

The rest of the paper is organized as follows: In Section 2, the sigma-pi-sigma neural network and the batch gradient algorithm based on graph regularization are described in detail. Section 3 analyses the experimental results. Finally, Section 4 gives some conclusions of this paper.

2. The Proposed Approach

(1) Sigma-Pi-Sigma neural network (SPSNN)

In this section, we introduce the SPSNN model. P, N, Q, and 1 are the number of neurons in the input layer, Σ_1 layer, $\prod$ layer, and Σ_2 layer, respectively (see Fig. 1). Let $\{X_j, O_j\}_{j=1}^{J} \subset R^P \times R$ be the set of training samples, where O_j is the desired output of input sample X_j and J is the number of training samples.

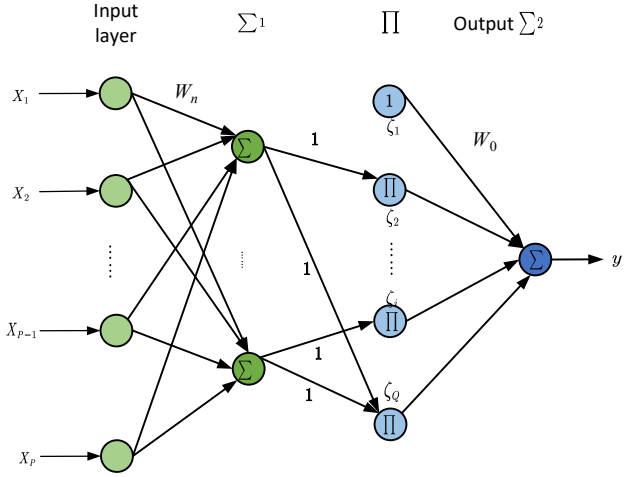

Figure 1. Topology of an SPSNN.

Let $W_i = (w_{i1}, w_{i2}, \cdots, w_{iP}) \in \mathbb{R}^P$ be the weight vector between the input layer and the Σ_1 layer at $i = 1, 2, \cdots, N$ Let $W_0 = (w_{01}, w_{02}, \cdots, w_{0Q}) \in \mathbb{R}^Q$ be the weight vector connecting the Σ_2 layer to the $\prod$ layer. $W = (W_0^T, W_1^T, \cdots, W_N^T)^T \in \mathbb{R}^{Q+NP}$ We write all the ownership values in a tightened form. Let $y^j = g(W_0 \cdot \varsigma^j)$ as the real output, i.e.,

$W = (W_0^T, W_1^T, \cdots, W_N^T)^T \in \mathbb{R}^{Q+NP}$. define $g: R \to R$ to be the activation function. $\tilde{E}(w)$ to be the conventional squared error function:

$$\tilde{E}(W) = \frac{1}{2}\sum_{j=1}^{J}\left(y^j - O^j\right)$$

$$= \sum_{j=1}^{J} g_j(W_0 \cdot \zeta^j)$$

Where $g_j(r) = \frac{1}{2}(g(r) - O^j), \quad r \in R,\ 1 \le j \le J$.

At the same time, we take the following forms.

$$\zeta^j = (\zeta_1^j, \zeta_2^j, \cdots \zeta_Q^j)$$

$$= (\prod_{i\in\Lambda_1}\sigma_i, \prod_{i\in\Lambda_2}\sigma_i, \cdots \prod_{i\in\Lambda_Q}\sigma_i)$$

$$= (\prod_{i\in\Lambda_1} g(W_i \cdot X^j), \prod_{i\in\Lambda_2} g(W_i \cdot X^j), \cdots \prod_{i\in\Lambda_Q} g(W_i \cdot X^j))$$

So, $\tilde{E}(w)$ can be represented as follows:

$$\tilde{E}(W) = \sum_{j=1}^{J} g_j(\sum_{q=1}^{Q} w_{0q} \cdot \prod_{i\in\wedge Q} g(w_i \cdot X^j))$$

and we can get the partial derivative as follows:

$$\tilde{E}_{w_0}(W) = \sum_{j=1}^{J} g_j'(w_0 \cdot \zeta^j)\zeta^j$$

In addition, according to $\zeta_q^j = \prod_{i\in\Lambda_q} g(W_i \cdot X^j)$,we get

$$\frac{\partial \zeta_q^j}{\partial W_n} = \begin{cases} (\prod_{i\in\Lambda_q \setminus n}\sigma_i)\cdot g'(W_i \cdot X^j)X^j, & if\ q \ne 1\ and\ n \in \Lambda_q \\ 0, & if\ q \equiv 1\ or\ n \notin \Lambda_q \end{cases}$$

Then, in light of the above definitions and (II.4), we have

$$\tilde{E}_{w_n}(W) = \sum_{j=1}^{J} g_j'(w_0 \cdot \zeta^j)\sum_{q\in\vee n}(w_{oq} \cdot \frac{\partial \zeta_q^j}{\partial w_n})$$

$$= \sum_{j=1}^{J} g_j'(w_0 \cdot \zeta^j)\sum_{q\in\vee n} w_{oq} \cdot \prod_{i\in\wedge q\setminus n}\sigma_i^j$$

$$\cdot \prod_{i\in\wedge Q} g(w_i \cdot X^j) \cdot X^j$$

A Sigma-Pi-Sigma neural network with graph regularity terms (SPSNNGR)

In this section, we describe our SPSNN model with graph regular terms, firstly we construct the weighted graph from the input data, secondly, we compute its Laplace matrix L based on the information given by the weighted graph, and finally we compute the graph regular terms based on the above information.

In this paper, we use a Gaussian kernel to measure the similarity between any two instances xi and xj to generate a weighted graph G. Let $\{X^j, O^j\}_{j=1}^{J} \subset \mathbb{R}^p \times \mathbb{R}$ be a given set of training samples, we have

$$W_{ij} = \begin{cases} e^{-\|x_i - x_j\|^2/\sigma^2} & if\ i \ne j \\ 0, & if\ otherwise \end{cases}$$

Where $\sigma^2 = \frac{1}{J}\sum_{j=1}^{J}\|x^j\|^2$.

Therefore, the normalized Laplace matrix L of the weighted graph G is defined as

$$L = D - W$$

where $D_{ii} = \sum_{j\ne i} W_{ij}$.

In order to make the outputs of hidden layers from the same class as close as possible and thus improve the network accuracy, we define the penalty term function R as

$$R = \frac{1}{2}\sum_{i,j=1}^{J}(f(x_i) - f(x_j))^2 W_{ij}$$

$$= \sum_{i=1}^{J} f(x_i)^2 D_{ii} - \sum_{i,j=1}^{J} f(x_i)f(x_j)W_{ij}$$

$$= Tr(VDV^T) - Tr(VWV^T)$$

$$= Tr(VLV^T)$$

$Tr(X)$ in the above equation is the trace of the matrix X, $f(x_i)$ is a function of the mapping of sample points x_i in high-dimensional space to low-dimensional, and V is the low-dimensional representation matrix.

Therefore, the error function can be defined as

$$E(W) = \tilde{E}(W) + \frac{\lambda}{2}R(W_0, W_n)$$

$$= \tilde{E}(W) + \frac{\lambda}{2}tr(f(W_n X)Lf(W_n X)^T)$$

For the sake of simpler calculations, we consider

$$E(W) = \tilde{E}(W) + \frac{\lambda}{2}tr((W_n X)L(W_n X)^T)$$

So, we can have

$$E_{w_0}(W) = \tilde{E}_{w_0}(W) = \sum_{j=1}^{J} g_j'(w_0 \cdot \zeta^j)\zeta^j$$

$$E_{w_n}(W) = \tilde{E}_{w_n}(W) + \frac{\lambda}{2}tr(f(W_n X)L(f'(W_n X)X)^T)$$

The initial weight W^0 is given, and the iterative updates of the weights of the SPSNNGR algorithm are as follows:

$W^{m+1} = W^m + \Delta W^m \qquad m = 1, 2, \cdots$ and $\Delta W^m = -\eta E_W(W^m)$.

where η is the learning rate.

3. Simulation Results

(1) Function Approximation Problems

To illustrate the effectiveness of the SPSNNGR algorithm, the following is an example to compare the SPSNN algorithm and the SPSSNNGR algorithm with a nonlinear function that

is formulated as

$$y = 0.2 \cdot (\sin(x) + 1.2)$$

For the function approximation problem, we used the network utilizing this structure to simulate all the algorithms(input $P = 2$, $\sum_1$ layer $N = 4$, Π layer $Q = 16$ and Σ_2 layer). In this experiment, 151 input variables were selected in the interval $[-3,3]$. The initial weight W is randomly selected in the interval $[-0.5,0.5]$, the learning rate $\eta = 0.01$ and the penalty term coefficient $\lambda = 0.001$. The experimental termination condition is that the error is less than 1×10^{-5} or the number of iterations reaches 5000 times. Figure 2 shows the network approximation graph of SPSNNGR, and Figure 3 shows the error function graphs of SPSNN algorithm and SPSNNGR algorithm, it can be seen that the SPSNNGR algorithm has good approximation ability and the error of SPSNNGR algorithm decreases faster, which indicates that our proposed algorithm performs better.

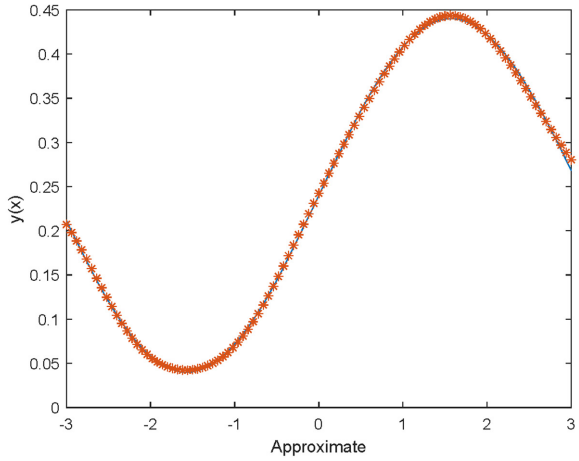

Figure 2. Approximation result of SPSNNGR

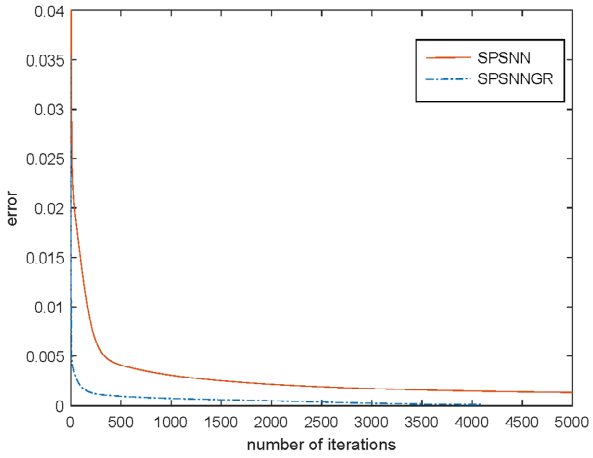

Figure 3. Errors of SPSNN and SPSNNGR.

(2) Real-World Classification Problems

To further validate the algorithmic performance of SPSNNGR, we conducted numerical experiments on 4 real datasets, all selected from the UCI database. In this experiment, we compare the performance of the SPSNN algorithm and the SPSNNGER algorithm.

We set the initial network architecture of SPSNN as(input $P = 2$, $\sum_1$ layer $N = 4$, Π layer $Q = 16$ and Σ_2 layer), where the learning rate $\eta = 0.001$ and the penalty term coefficient $\lambda = 0.001$. In addition, our initial weights W are randomly generated from the interval, and we set the training to stop when the error is less than 1×10^{-5} or the number of iterations reach 5000. We used a cross-validation approach that, 70% of each dataset is selected as the training set, and the rest is used as the test set, to obtain the average results of each algorithm under 10 simulation experiments, and to provide a detailed analysis and comparison of their performance.

Table 1. Performance comparison of classification problems

Data Set	Accuracy Rate	SPSNN	SPSNNGR
Tic	Training	0.9584	0.9642
	Testing	0.9061	0.9229
Heart	Training	0.9670	0.9952
	Testing	0.9187	0.9890
Wireless	Training	0.8350	0.8486
	Testing	0.8171	0.8404
Wholesale	Training	0.8539	0.8571
	Testing	0.8045	0.8271

The performance comparison results are shown in the Table 1, we can clearly see that for most of the datasets, the SPSNNGR algorithm has higher training and testing accuracy compared to the SPSNN algorithm, which indicates that the SPSNNGR algorithm is more generalized and stable.

4. Conclusion

In his paper, a new Sigma-Pi-Sigma neural network model with graph regularity is established by introducing the graph regularity term, and the algorithm not only improves the generalization ability of the network, but also improves the convergence speed. In addition, we conducted numerical experiments to verify the effectiveness of the new algorithm, and it can be seen that, compared with the SPSNN algorithm, the new algorithm not only has good performance in the function approximation experiments, but also has higher accuracy and better generalization performance in the classification experiments

References

[1] C K Li. A Sigma-Pi-Sigma Neural Network (SPSNN)[J]. Neural Processing Letters, 2003(17): 1-19.

[2] Q W Fan, Q Kang, J M Zurada. Convergence analysis for sigma-pi-sigma neural network based on some relaxed conditions[J]. Information Sciences, 2022(585): 70-88.

[3] S N Arsla. A hybrid sigma-pi neural network for combined intuitionistic fuzzy time series prediction model[J]. Neural Comput and Applic,2022(34): 12895-12917.

[4] R S Neville. Reuse of information in multi-layer sigma-pi neural networks[J]. Connection Science, 2006(18): 43-59.

[5] G Welper. Universality of gradient descent neural network training[J]. Neural Networks, 2022(150): 259-273.

[6] Y Liu, D K Yang, F Li. Smoothed regularizer learning for split-complex valued neuro-fuzzy algorithm for TSK system and its convergence results[J]. Journal of the Franklin Institute, 2018 (355): 6132-6151.

[7] P May, E Zhou, C W Lee. A Comprehensive Evaluation of Weight Growth and Weight Elimination Methods Using the

Tangent Plane Algorithm[J]. International Journal of Advanced Computer Science and Applications, 2013(6):149-156.

[8] E Geoffrey. Connectionist learning procedures[J]. Artificial Intelligence, 1989(40):185-234.

[9] P L Zhai, S H Zhang. Learnable Graph-Regularization for Matrix Decomposition[J]. Association for Computing Machinery, 2023(17):1-20.

[10] R Nassif, S Vlaski. A Regularization Framework for Learning Over Multitask Graphs[J]. in IEEE Signal Processing Letters, 2019(26):297-301.

[11] Y T Fan, W Y Yang. A backpropagation learning algorithm with graph regularization for feedforward neural networks[J]. Information Sciences, 2022(607): 263-277.