Improving Time Series Prediction by the Selective Desensitization Neural Network Based on Synaptic Weight Analysis

Shoichi Someno[1], Kazumasa Horie[2], Tomoki Ichiba[1], Tomohiro Aki[1], Masahiro Morita[1]

[1] Graduate School of Systems and Information Engineering
University of Tsukuba,
1–1–1 Tennodai, Tsukuba-shi, 305–8573 Japan

[2] Center for Computational Sciences
University of Tsukuba,
1–1–1 Tennodai, Tsukuba-shi, 305–8577 Japan

Abstract

The selective desensitization neural network (SDNN) is a function approximator with high learning ability. It is suitable for time-series prediction problems with concept drift. However, it is not very effective at dealing with a long time window, because the computational costs increase with the square of the number of input variables. In this paper, we present a method for reducing the computational costs of the SDNN for time-series prediction and eliminating trials and errors by analyzing the synaptic weights of the SDNN after training. We confirmed, using an artificial problem, that the variance of synaptic weights reflects the significance of the corresponding input variable. We also applied this analysis to a real-world problem and improved prediction accuracy by expanding the time window without increasing computational costs. This method may be applied to other kinds of problems and thus, extend the application range of the SDNN.

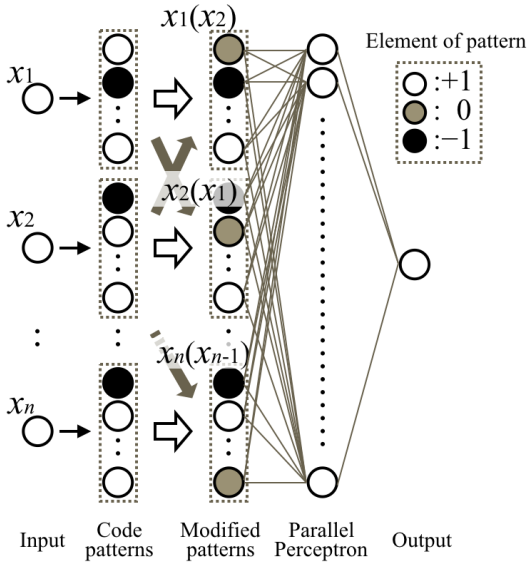

Figure 1: Structure of an SDNN with n dimensional input

1. Introduction

The selective desensitization neural network (SDNN) is a function approximator with high expression and generalization abilities and numerous other merits such as low hyperparameter dependency and suitability for online incremental learning [1]. In light of these advantages, Ichiba et al. [2] applied the SDNN to predicting time series data with concept drift and achieved better performance than previous studies.

However, the computational costs (computation time and required memory) of the SDNN increase with the square of the number n of input variables because the basic SDNN calculates the output from the $m = n(n-1)$ variable pairs, which limits the size of n. In the case of time series prediction, the length of the time window is restricted, even if a longer time window is desired [3]. Although we can reduce the computational costs by eliminating insignificant variable pairs [4], prioritization of their significance is difficult; it usually requires prior knowledge of the problem or many trials and errors.

We have recently found that it is possible to analyze the significance of variable pairs based on the synaptic weights of the trained SDNN. Specifically, we found that in the approximation task of an artificial function with additional input variables that were irrelevant to the function value, the variance of the synaptic weights for pairs of irrelevant variables was less than that for pairs of relevant variables. However, it was not clear whether this analysis is effective at evaluating the significance of variable pairs and reducing the computational costs in general cases.

In the present study, we analyze the synaptic weights of the SDNN for time series prediction and aim to improve the prediction accuracy with lower computational costs and minimal effort.

2. Selective Desensitization Neural Network

The SDNN is constructed by introducing two manipulations, pattern coding and selective desensitization into a parallel perceptron in Figure 1.

2.1 Pattern coding

First, the analogue input value is converted into a high-dimensional binary vector (code pattern). Specifically, we quantize the input range into q bins and create a lookup table in which the code patterns P^1, P^2, ..., P^q are assigned to each bin. Each code pattern P^1, ..., P^q is the L dimension binary code pattern (L represents even numbers), consisting of equal numbers, $+1$ and -1. Furthermore, it is preferable for the correlation between two consecutive patterns to be high and that between two patterns that are further apart to be closer to 0.

There are several methods of creating such code patterns [1]. In this study, the following procedure is used. First, we randomly create a pattern whose $+1$ and -1 are equal numbers, and designate it P^1. Then, we randomly select each r elements of $+1$ and -1 from P^1, and we invert these signs to get P^2. Likewise, P^3 is created by inverting the signals of $2r$ in the elements of P^2. Similarly, P^u is sequentially created by inverting $2r$ elements of P^{u-1}. The expression and generalization ability of the SDNN are influenced by the dimension number of code pattern L, division number of input range q, and the number of inversion of the adjacent pattern r. In this study, these parameters were selected based on preliminary experiments.

2.2 Selective desensitization

Selective desensitization is an operation that involves integrating two binary code patterns into one ternary $(-1, 0, 1)$ code pattern by modifying one with the other. For the explanation of the modification procedure, we specifically consider the instance where the code pattern $S = (s_1, s_2, ..., s_L)$ is modified with the code pattern $C = (c_1, c_2, ..., c_L)$. It is also to be assumed that there is a random one-to-one correspondence between the elements of S and the elements of C. If there is a correspondence between s_μ and c_ν, the modified element s'_μ is represented as follows:

$$s'_\mu = \frac{1 + c_\nu}{2} s_\mu \tag{1}$$

In the case of $c_\nu = +1$, the value of s_μ is retained as it is. In the case of $c_\nu = -1$, it is converted to a neutral value ($= 0$). Now, half of the elements of C are -1, therefore, half of the elements of the ternary code patterns $(s'_1, s'_2, ..., s'_L)$ are 0 , while the other half are ± 1. This code pattern is called "S modified by C" and is represented as $S(C)$. The signals of all the modified code patterns enter into the parallel perceptron.

2.3 Parallel perceptron

The parallel perceptron is created by paralleling simple perceptrons that output either 0 or 1, and determining the overall output based on the sum of the output values of all the simple perceptrons. Error correction learning is used in parallel perceptron learning. For example, let us assume l out of K simple perceptrons should output 1, but only $v(< l)$ of them actually outputted 1. At this point, we select $l - v$ perceptrons with internal potentials nearest the threshold value among the $K - v$ perceptrons that outputted 0 and train them. In the case of $v > l$, we select $v - l$ perceptrons with internal potentials nearest the threshold value among the perceptrons that outputted 1.

3. Experiment 1

In Experiment 1, we examine the relationship between the variance of the synaptic weights and the significance of the variables using artificial time series.

3.1 Methods

We use a chaotic time series $\{d_1, d_2, \ldots, d_{1000}\}$ generated by the logistic map [5] and small noise. Specifically, the i-th component d_i is represented in the following equation:

$$d_i = 3.9 d_{i-1}(1 - d_{i-1}) + N(0, 0.02^2) \tag{2}$$

, where $N(0, 0.02^2)$ denotes Gaussian noise with zero mean and a standard deviation of 0.02.

At each time step t, the SDNN receives d_{t-1}, ..., and d_{t-6} as the input variables x_1, ..., x_6 and is trained to predict d_t. The training is performed completely online; thus, the input vector is given only once.

In the case of $499 \leq t < 1000$, the SDNN also receives d_t, ..., d_{t-5} and outputs $\hat{d}_{t+1}$, the predicted value of d_{t+1}, to calculate the mean absolute error (MAE).

Then, we analyze the synaptic weights of the parallel perceptron. First, the input synapses are divided into 30 groups, corresponding to the 30 variable pairs $x_1(x_2)$, ..., $x_6(x_5)$, where $x_i(x_j)$ represents the modification of the variable x_i by x_j. For each group, the variance of synaptic weights is calculated. Afterwards, we arrange the variable pairs in descending order of variance. Finally, we repeat the above training and error evaluation using first m pairs only, with m varying from 1 to 29.

The parameters of the SDNN are $L = 1000$, $K = 700$, $q = 501$, and $r = 2$.

3.2 Results

Figure 2 shows the predicted and actual values of the time series for $970 \leq t \leq 1000$, indicating that the SDNN can efficiently predict this time series. The MAE was 0.037.

Figure 3 shows the variance of synaptic weights for each variable pair. We see that the value of the variance depended mainly on the modified variable. The variance for $x_i(x_j)$ tended to decrease as i and j increased. As x_1 is the most

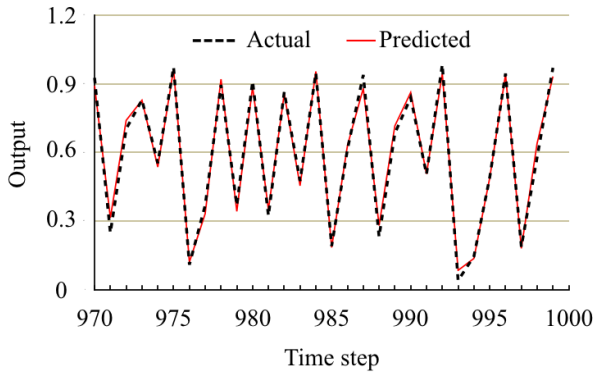

Figure 2: Predicted and actual values of the time series (a part of data)

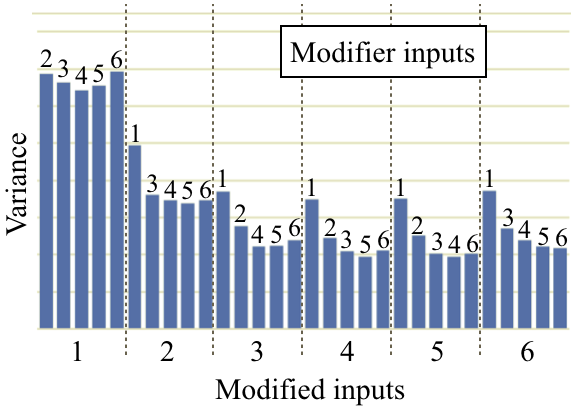

Figure 3: Variance of synaptic weights for each variable pair

important factor for predicting the next value, and prediction from x_i ($i = 2, ..., 6$) becomes more difficult with an increase in i, because of noise and the chaotic property of the logistic map, this result is considered to indicate that the value of the variance reflects the significance of the variable and variable pair for prediction.

Figure 4 shows the prediction error (MAE) against the number m of variable pairs used for training and prediction. From this result, we see that removing variable pairs with low variance reduced not only the computational costs but also the prediction error, indicating that these pairs were unnecessary for prediction. We also see that although it is possible to base prediction on only one pair in this task, using many pairs with large variance improved the prediction accuracy, probably because the influence of the noises contained in the different variables was canceled.

4. Experiment 2

In Experiment 2, we apply the above method to a real-world task of time-series prediction.

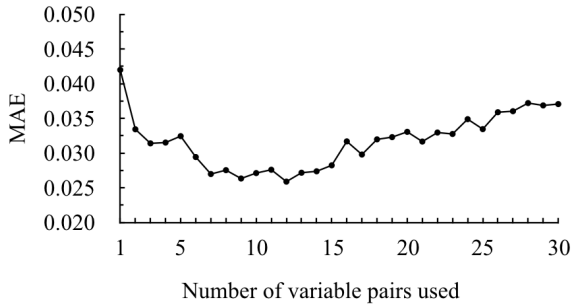

Figure 4: Prediction error when only high variance pairs were used

4.1 Methods

Ichiba et al. [2] applied SDNN to the prediction of the future closing price of Nikkei 225 on the Tokyo Stock Exchange[1] based on previous closing prices. First, we briefly explain their method.

The dataset is a time series of 4906 components $\{d_1, ..., d_{4906}\}$. The task is to predict the price of Nikkei 225 N days ahead based on the prices of the last k days ($N = 1, 3, 5$, $k = 30$).

At time step t, k data points $d_{t-N}, \ldots, d_{t-N-k+1}$ are inputted in the SDNN in this form: $x_i = d_{t-N-i+1}/\bar{d}_{t-N}$ ($i = 1, \ldots, 30$), where $\bar{d}$ is the average of $d_{t-N}, \ldots, d_{t-N-199}$, and was trained to predict $d_t/\bar{d}_{t-N}$. At $t > 1000$, it also predicted $\hat{d}_{t+N}/\bar{d}_t$ based on $d_t, \ldots, d_{t-k+1}$, and calculated the prediction error between d_{t+N} and $\hat{d}_{t+N}$.

In the present experiment, we first analyze the "original" SDNN used in [2] for $N = 3$ in the same way as in Experiment 1.

Based on the results of this analysis, we revise the SDNN by eliminating insignificant variable pairs and extending the size, k, of the time-window and adding some new pairs. Then, we test the revised SDNN for N = 1, 3, and 5, and examine the possibility of improving the prediction accuracy without increasing the computational costs, while keeping trials and errors to the minimum.

The parameters of the SDNN are fixed at $L = 1250$, $K = 700$, $q = 1251$, $r = 1$.

4.2 Results

Figure 5 shows the variance of the synaptic weights for all (870) pairs $x_i(x_j)$ ($i, j = 1, \ldots, 30$, $i \neq j$). It is seen that the variance for $x_i(x_j)$ mainly depends on i and is particularly large for $i \leq 6$, suggesting that the prices within the five previous days are particularly significant. It is also seen that the prices around 21 days prior might be more significant than those around ten days or 28 days prior.

[1] https://finance.yahoo.com/quote/% 5EN225/

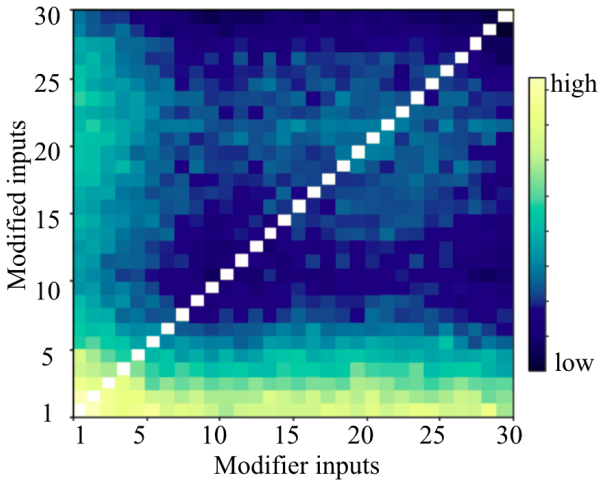

Figure 5: Variance of synaptic weights for each variable pair

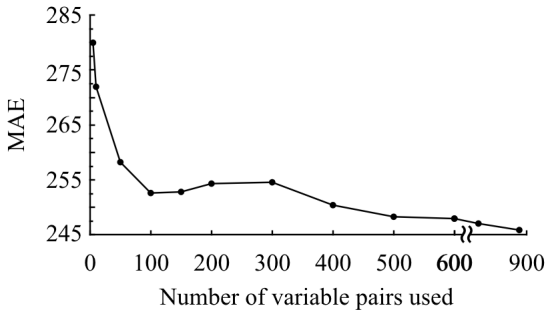

Figure 6: Prediction error when only high variance pairs were used

Figure 6 shows the prediction errors (MAE) when only m variable pairs with larger variance were used for training and prediction. The error tended to decrease as more pairs were used, but the decrease was gentle for $m > 100$.

Based on these results, we extended the time window and revised the SDNN. Specifically, we reduced 770 pairs, excluding the top 100 pairs from the original SDNN. Then, we selected 28 variables $x_1, \ldots, x_7, x_{12}, x_{17}, \ldots, x_{112}$ and added 756 pairs by combining them. As 38 pairs were included in the top 100 pairs, 818 pairs were used in the revised SDNN.

Table 1 compares the errors of the revised SDNN with $m = 818$ and that of the original SDNN with $m = 870$. Although the revised SDNN showed almost the same error as the original for $N = 1$, it exhibited considerably less error for $N = 5$, indicating that the extension of the time window was more effective for longer term prediction.

5. Conclusions

We have analyzed the SDNN for an artificial time-series prediction task and shown that the variance of the synaptic weights for a variable pair reflects the significance of the pair, and that we can reduce both the prediction error and the computational costs by removing pairs with low variance. We have also applied this method to a real-world time series prediction task on stock price, and demonstrated that with minimal trials and errors, it can improve prediction accuracy without increasing computational costs.

Furthermore, the method of analyzing and revising the SDNN presented here may be applied to different types of problems; thus, it extends the application range of SDNN.

Table 1: Prediction errors for the original and revised SDNNs

	$N = 1$		$N = 3$		$N = 5$	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
Original	138.4	194.1	245.9	335.3	327.7	439.8
Revised	138.0	193.7	242.4	329.3	314.8	423.5

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP18H03304.

References

[1] K. Nonaka, F. Tanaka and M. Morita: Empirical comparison of feedforward neural networks on two-variable function approximation, IEICE Trans. Inf. & Syst. (Japanese Edition), Vol. J94-D-II, No. 12, pp. 2114–2125, 2011.

[2] T. Ichiba, K. Horie, S. Someno, T. Aki, and M. Morita: Application of the selective desensitization neural network to concept drift problems, RISP International Workshop on Nonlinear Circuits, Communications and Signal Processing, 2019 (accepted).

[3] A. Yamaguchi, S. Maya, T. Inagi, and K. Ueno: OPOSSAM online prediction of stream data using self-adaptive memory, IEEE International Conference on Big Data, 2018 (accepted).

[4] K. Horie, A. Suemitsu, T. Tanno, and M. Morita: Direct estimation of wrist joint angular velocities from surface EMGs by using an SDNN function approximator, International Conference on Neural Information Processing, 2016.

[5] G. Boeing: Visual analysis of nonlinear dynamical systems: chaos, fractals, self-similarity and the limits of prediction, Systems, Vol. 4, 2016.