RISP
NCSP'16
International Workshop on
Nonlinear Circuits, Communications and Signal Processing
Honolulu, Hawaii, USA, March 6-9, 2016

Improving Back Propagation of Feed-Forward Neural Network with Changing Sigmoid Functions

Shinsaburo Kittaka, Yoko Uwate, and Yoshifumi Nishio

Dept. of Electrical and Electronic Engineering, Tokushima University
2-1 Minami-Josanjima, Tokushima 770–8506, Japan
Email: {kittaka, uwate, nishio}@ee.tokushima-u.ac.jp

Abstract

Our study is about feed-forward neural network's learning method. Generally, the method of improving its learning is focused on learning rate and moment term. We focus on sigmoid functions. Sigmoid functions are used for converting input signal into output signal and adjusting connection weight of learning in feed-forward neural network. We change gradient of sigmoid functions and investigate our method's effect.

1. Introduction

Neural network is constituted by some element that is called neuron. This network is based on animal brain. With similarity brain, it is appropriate in recognize written characters and recognize voice. Neural network's features are parallel processing and learning. We focus on learning. Neuron in neural network transforms input signal to output signal with sigmoid functions and adjusts connection weight from error between actual value and estimated value.

We regard that learning system is appropriate in recognize brain signal, because brain signal has difference with each person and condition. In other research, the recognition brain signal rate is 86 percent [1] [2]. We submit that we should increase this rate. In this study, we focus on Feed-forward neural network's learning method. Feed-forward neural network's learning method is back propagation. We change sigmoid functions and improve back propagation.

2. Feed-forward neural network

Feed-forward neural network is one of neural network. It consists of input layer, hidden layer and output layer. Figure 1 shows feed-forward neural network's composition.

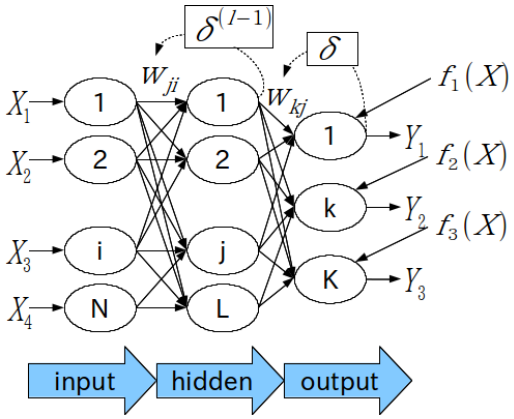

Figure 1: Feed-forward neural networks composition .

Signal runs only one direction. Neuron converts input signal to output signal that is from 0 to 1 with sigmoid funcitons.

$$f(x) = \frac{1}{1 + exp(-x)} \tag{1}$$

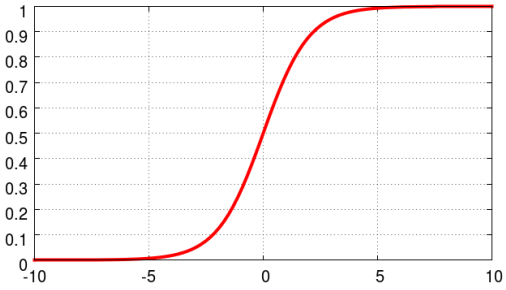

Figure 2: Sigmoid functions.

2.1. Back propagation

Back propagation is learning method of feed-forward neural network. Estimated value is inputed into output layer. Output layer calculates the difference of actual value and estimated value.(2) The connection weight that is between neuron and neuron adjusts based on this value.(3) $f_n(x)$ is actual value and Y_n is estimated value. W is the value of connection weight, η is learning rate and net_j is input signal from all connecting neurons.

$$\delta = f_n(x) - Y_n \tag{2}$$

$$\Delta W = \eta \delta_n f'(net_j) \tag{3}$$

Repeating this calculate makes output signal perfect.

3. Proposed method

In this study, we change gradient of sigmoid functions. We decrease the parameter "k" that is coefficient added into natural logarithm's multiplier.

$$f(x) = \frac{1}{1 + exp(-kx)} \tag{4}$$

Figure 3 shows sigmoid functions with changing "k" and Fig. 4 shows differential form of sigmoid functions.

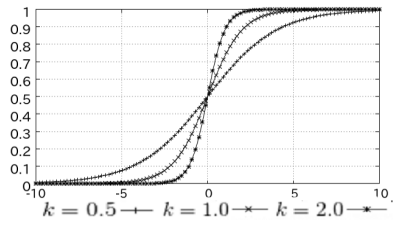

Figure 3: Sigmoid functions with "k".

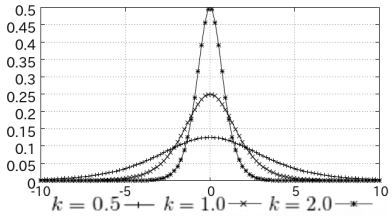

Figure 4: Differencial form of sigmoid functions.

First, we search learning late that has the smallest error. Second, the parameter "k" is changed. Third, we decrease "k" by 0.001 at iteration. Finally, we change the decreasing value of "k".

4. Simulation result

In this simulation, we investigate 2 simulations. One is Iris plants and another one is sin function. We define parameters as learning iteration = 1000 and the number of hidden layer's neurons = 4.

4.1. Iris plants

We divide Iris plants into 3 types. There are total of 150 data entries. The data records 4 attributes and 3 classifications. The 4 attributes are sepal length, sepal width, petal length and petal width. The 3 types of iris flower are Iris Setosa, Iris Versicolor and Iris Virginica. First, we search the learning rate (a) that has the smallest error. Table I shows the error with changing learning rate from 0.1 to 1.0.

Table 1: The error with changing a

a	0.1	0.2	0.3	0.4	0.5
ave	0.02896	0.02438	0.02974	0.02261	0.02144
a	0.6	0.7	0.8	0.9	1.0
ave	0.02144	0.02079	0.02137	0.02112	0.02144

We define as the learning rate = 0.7, because it has the smallest error. We pursue simulation with this learning late. Next, we search the "k" that has the smallest error. Table II shows the error with changing the "k" from 0.6 to 1.5.

Table 2: The error with changing "k"

k	0.6	0.7	0.8	0.9	1.0
ave	0.02295	0.02231	0.02177	0.02221	0.02079
k	1.1	1.2	1.3	1.4	1.5
ave	0.02076	0.02237	0.02267	0.02364	0.02365

In this simulation, we obtain smaller error than the original method. However, it is very small difference. We consider that this simulation does not obtain good results. Next, we decrease "k" by 0.001 at iteration and change the starting decreasing time. Table III shows the error with decreasing "k".

In this simulation, we obtain very small error. This method is effective to improve back propagation. In Table III, best parameters are "k" = 0.9, the decreasing value = 0.001 and the starting decreasing time = 500. Finally, we change the decreasing value of "k". We set best parameters in Table III and investigate the error with changing decreasing value of "k" at iteration from 0.0006 to 0.0015. Table IV shows results.

Table 4: The error with changing decreasing value

value	0.0006	0.0007	0.0008	0.0009	0.001
ave	0.01748	0.01547	0.01490	0.01271	0.01078
value	0.0011	0.0012	0.0013	0.0014	0.0015
ave	0.01544	0.01711	0.01214	0.01693	0.01285

The smallest error is 0.01078 with the decreasing value of "k" = 0.001. In this simulation, we obtain half of the original method's error.

4.2. Sin function

Next, we transform input signal from angle into sin function. There are total of 180 data entries. The data records angle from 0 to 180 and sin function from 0 to 1. First, we search the learning rate (a) that has the smallest error. In this time, the learning late that has smallest error is big. So, table V shows the error with changing learning rate from 2.1 to 3.0.

Table 5: The error with changing a

a	2.1	2.2	2.3	2.4	2.5
ave	0.01716	0.01667	0.01678	0.01592	0.01687
a	2.6	2.7	2.8	2.9	3.0
ave	0.01651	0.01632	0.01527	0.01503	0.01541

We define the learning rate = 2.9, because it has the smallest error. We pursue simulation with this learning late. Next, we search the "k" that has the smallest error. Table VI shows the error with changing the"k" from 1.1 to 2.0.

Table 6: The error with changing "k"

k	1.1	1.2	1.3	1.4	1.5
ave	0.01604	0.01415	0.01493	0.01349	0.01511
k	1.6	1.7	1.8	1.9	2.0
ave	0.01383	0.01331	0.01422	0.01311	0.01335

We obtain smaller error than the original learning. 1.9 is the best parameter of "k" in table VI. Next, we decrease "k" by 0.001 at iteration and change the starting decreasing time. Table VII shows the error with decreasing "k".

This method has smaller error than the orginal method. However, we can not obtain effective results like first simulation. In this simulation, best parameters are "k" = 2.0, the decreasing value = 0.001 and the starting decreasing time = 200. Finally, we change decreasing value of "k". We set best parameters in Table VII and investigate the error with changing decreasing value of "k" at iteration from 0.0006 to 0.0015. Table VIII shows results.

Table 8: The error with changing decreasing value

value	0.0006	0.0007	0.0008	0.0009	0.001
ave	0.01035	0.00906	0.01177	0.01044	0.01118
value	0.0011	0.0012	0.0013	0.0014	0.0015
ave	0.01198	0.01273	0.01303	0.01672	0.01688

The smallest error is 0.0906 with the decreasing value = 0.007. We obtain smaller error than the original method.

5. Conclusion

Our learning method is changing gradient of sigmoid functions by decreasing "k" at iteration for back propagation in feed-forward neural network. In these simulations, we obtain smaller error than the usual method. However, if the "k" is small and starting decreasing time is early, our method makes error very large. So, this method requires attention to set parameters.

In the future work, we resolution that why the error becomes small with our method.

References

[1] A. Hiraiwa, N Uchida, K Shimohara, N Sonehara "EMG Recognition with a Neural Network Model for Cyber Finger Control," Transactions of the Society of Instrument and control Engineers, vol. 30, no.2, pp.216-224, 1994.

[2] S. Kitayama, M. Sasaki, S. Ito, "Limb motion estimation by Bereitschaffs Potentian," Journal of the Japan Society of Applied Electromagnetics and Mechanics, vol. 22, no. 2, pp. 318-323, 2014.

Table 3: The error with changing "k"

		k									
starting decreasing time		0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5
iteration	0	0.33406	0.19977	0.18239	0.01315	0.01150	0.01282	0.01576	0.01674	0.01780	0.01747
	100	0.22713	0.17337	0.21337	0.01244	0.01225	0.01301	0.01509	0.01622	0.01732	0.01743
	200	0.13594	0.16410	0.17844	0.01165	0.01143	0.01303	0.01492	0.01591	0.01598	0.01707
	300	0.10816	0.09496	0.08314	0.01304	0.01106	0.01465	0.01575	0.01514	0.01659	0.01690
	400	0.02578	0.03072	0.05036	0.01180	0.01395	0.01356	0.01587	0.01697	0.01644	0.01697
	500	0.01394	0.01325	0.01122	0.01071	0.01101	0.01301	0.01325	0.01556	0.01617	0.01584
	600	0.10579	0.02672	0.01150	0.01072	0.01148	0.01219	0.01589	0.01560	0.01574	0.01639
	700	0.02457	0.02290	0.03057	0.01090	0.01183	0.01378	0.01642	0.01610	0.01682	0.01624
	800	0.01371	0.01937	0.02613	0.01386	0.01336	0.01409	0.01471	0.01558	0.01644	0.01664
	900	0.01378	0.01228	0.01194	0.01336	0.01250	0.01498	0.01735	0.01528	0.01610	0.01648

Table 7: The error with changing "k"

		k									
starting decreasing time		1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
iteration	0	0.02962	0.02434	0.02279	0.01834	0.01829	0.01684	0.01526	0.01552	0.01547	0.01291
	100	0.02528	0.02261	0.02145	0.02059	0.01926	0.01744	0.01485	0.01508	0.01467	0.01205
	200	0.02424	0.02661	0.02240	0.01832	0.01699	0.01657	0.01470	0.01329	0.01146	0.01097
	300	0.03176	0.02379	0.02006	0.01779	0.01650	0.01441	0.01495	0.01199	0.01204	0.01307
	400	0.02925	0.02362	0.02213	0.02002	0.02130	0.01562	0.01487	0.01322	0.01237	0.01241
	500	0.03032	0.02692	0.02504	0.02203	0.01798	0.01749	0.01547	0.01422	0.01321	0.01307
	600	0.03175	0.03012	0.02393	0.02003	0.01791	0.01752	0.01601	0.01521	0.01520	0.01392
	700	0.03351	0.02595	0.02590	0.02278	0.01905	0.01876	0.01899	0.01753	0.01568	0.01540
	800	0.03773	0.03035	0.02523	0.02360	0.01987	0.02429	0.02236	0.01889	0.01892	0.01642
	900	0.06038	0.04008	0.03369	0.02365	0.02586	0.02658	0.02741	0.02393	0.02168	0.01897