Singularity Stairs Following with Limited Numbers of Hidden Units

Seiya Satoh and Ryohei Nakano
Department of Computer Science, Chubu University, 1200 Matsumoto-cho, Kasugai 487-8501, Japan

Keywords: Multilayer Perceptron, Learning Method, Singular Region, Reducibility Mapping.

Abstract: In a search space of a multilayer perceptron having J hidden units, MLP(J), there exist flat areas called singular regions that cause serious stagnation of learning. Recently a method called SSF1.3 utilizing singular regions has been proposed to systematically and stably find excellent solutions. SSF1.3 starts search from a search space of MLP(1), increasing J one by one. This paper proposes SSF2 that performs MLP search by utilizing singular regions with J changed bidirectionally within a certain range. The proposed method was evaluated using artificial and real data sets.

1 INTRODUCTION

In a multilayer perceptron (MLP) search space, there exist flat areas called singular regions where gradients are zero and the corresponding input-output maps are the same (I-O equivalent) (Sussmann, 1992). Since most learning methods get stuck in singular regions, a method (Amari, 1998) was once proposed so as to avoid singular regions, but there is no guarantee that avoiding singular regions will provide an excellent solution.

Statistical models having singular regions are called singular models, and many useful models such as MLPs, RBFs, Gaussian mixtures, and HMMs are singular models (Watanabe, 2009). Learning theory of singular models has been studied intensively (Watanabe, 2008; Watanabe, 2009); however, empirical studies of singular models have been scarcely done. As partial knowledge, we know that an MLP search space has extensive flat areas and troughs (Hecht-Nielsen, 1990), or most points along a search route have huge condition numbers (Nakano et al., 2011). Moreover, an MLP search space has many equivalent points due to equivalence relations (Sussmann, 1992) caused by permutations of hidden units or the nature of an activation function. Even after excluding such equivalences, an MLP search space may have local optima (Duda et al., 2001).

A completely new learning method SSF (Singularity Stairs Following) (Nakano et al., 2011) making good use of singular regions was proposed to stably and successively find excellent solutions of an MLP. SSF1.2 (Satoh and Nakano, 2013) extended the original so as to cover all kinds of singular regions and to reduce the number of starting points. SSF1.3 (Satoh and Nakano, 2014) accelerates SSF1.2 by introducing search pruning without deteriorating solution quality.

This paper proposes a new method called SSF2 which performs MLP search by making good use of singular regions with the number of hidden units J changed bidirectionally within a certain range. The method was evaluated using artificial and real data.

2 SINGULAR REGIONS OF MULTILAYER PERCEPTRON

2.1 Reducibility Mapping and Singular Regions

Let MLP(J) be an MLP having J hidden units. This section explains how we can generate singular regions in MLP(J) search space by applying reducibility mappings to the optimum of MLP($J-1$) (Fukumizu and Amari, 2000).

We consider MLP(J) having one output unit. The output for input x is defined as below. Here parameters $\theta_J = \{w_0, w_j, j = 1, \cdots, J\}$ and $g(h)$ is an activation function, where w_j is a vector of weights from all input units to hidden unit j, and w_j is a weight from hidden unit j to the single output.

$$f_J(x;\theta_J) = w_0 + \sum_{j=1}^{J} w_j z_j, \;\; z_j \equiv g(w_j^{\mathrm{T}} x) \qquad (1)$$

180 Satoh S. and Nakano R..
Singularity Stairs Following with Limited Numbers of Hidden Units.
DOI: 10.5220/0005075601800186
In *Proceedings of the International Conference on Neural Computation Theory and Applications* (NCTA-2014), pages 180-186
ISBN: 978-989-758-054-3

Given data $\{(x^\mu, y^\mu), \mu = 1, \cdots, N\}$, we try to find MLP($J$) which minimizes the following:

$$E_J \equiv E(\theta_J) = \frac{1}{2}\sum_{\mu=1}^{N}(f_J^\mu - y^\mu)^2, \quad (2)$$
$$f_J^\mu \equiv f_J(x^\mu; \theta_J).$$

We also consider MLP($J-1$) having parameters $\theta_{J-1} = \{u_0, u_j, u_j, j = 2, \cdots, J\}$. The output is defined as below.

$$f_{J-1}(x; \theta_{J-1}) = u_0 + \sum_{j=2}^{J} u_j v_j, \;\; v_j \equiv g(u_j^{\mathrm{T}} x) \quad (3)$$

Now we consider the following three reducibility mappings α, β, and γ. Let $\widehat{\Theta}_J^\alpha$, $\widehat{\Theta}_J^\beta$, and $\widehat{\Theta}_J^\gamma$ be the regions obtained by applying mappings α, β, and γ respectively to the optimum $\widehat{\theta}_{J-1}$.

$$\widehat{\theta}_{J-1} \xrightarrow{\alpha} \widehat{\Theta}_J^\alpha, \quad \widehat{\theta}_{J-1} \xrightarrow{\beta} \widehat{\Theta}_J^\beta, \quad \widehat{\theta}_{J-1} \xrightarrow{\gamma} \widehat{\Theta}_J^\gamma$$

$$\widehat{\Theta}_J^\alpha \equiv \{\theta_J |\; w_0 = \widehat{u}_0,\; w_1 = 0, \quad (4)$$
$$w_j = \widehat{u}_j, w_j = \widehat{u}_j, j = 2, \cdots, J\}$$
$$\widehat{\Theta}_J^\beta \equiv \{\theta_J |\; w_0 + w_1 g(w_{10}) = \widehat{u}_0, \quad (5)$$
$$w_1 = [w_{10}, 0, \cdots, 0]^{\mathrm{T}},$$
$$w_j = \widehat{u}_j, w_j = \widehat{u}_j, j = 2, \cdots, J\}$$
$$\widehat{\Theta}_J^\gamma \equiv \{\theta_J |\; w_0 = \widehat{u}_0,\; w_1 + w_m = \widehat{u}_m, \quad (6)$$
$$w_1 = w_m = \widehat{u}_m,$$
$$w_j = \widehat{u}_j, w_j = \widehat{u}_j, j \in \{2, \cdots, J\} \setminus \{m\}\}$$

From the above, the following two kinds of singular regions are formed in MLP(J) search space. One is $\widehat{\Theta}_J^{\alpha\beta}$, the intersection of $\widehat{\Theta}_J^\alpha$ and $\widehat{\Theta}_J^\beta$. The parameters are as follows, and only w_{10} is free.

$$w_0 = \widehat{u}_0,\; w_1 = 0,\; w_1 = [w_{10}, 0, \cdots, 0]^{\mathrm{T}}$$
$$w_j = \widehat{u}_j,\; w_j = \widehat{u}_j, \;\; j = 2, \cdots, J \quad (7)$$

The other is $\widehat{\Theta}_J^\gamma$, having the following restriction.

$$w_1 + w_m = \widehat{u}_m \quad (8)$$

2.2 SSF1.3 (Singularity Stairs Following ver. 1.3)

The search method SSF1.3 (Satoh and Nakano, 2014) finds solutions of MLP(J) successively from J=1 until J_{max} making good use of singular regions of each MLP(J). Since gradients are zero at points in a singular region, we employ eigenvector descent (Satoh and Nakano, 2012) only at a starting point, and employ BPQ (BP based on Quasi-Newton) (Saito and Nakano, 1997), a kind of quasi-Newton, from then on. Eigenvector descent calculates the Hessian matrix H, and then as a search direction pick up the eigenvector corresponding to each negative eigenvalue of H.

SSF1.3 has the following characteristics.

(1) Since most points in singular regions are saddles (Fukumizu and Amari, 2000), SSF1.3 can descend toward a better solution, thus guaranteeing monotonic decrease of training error.

(2) SSF1.3 does not have to be repeated since random numbers are not used in SSF1.3.

(3) SSF1.3 finds a set of solutions for successive J = $1, \cdots, J_{max}$, which can be quite useful for model selection.

(4) During each search process, the possibility of search pruning is checked if the current search will merge into any existing search route. If a positive result is obtained, the search is pruned.

3 PROPOSED METHOD: SSF2

As stated above, SSF1.3 finds excellent solutions successively for $J = 1, \cdots, J_{max}$. However, when we are interested in a large J and solutions for a smaller J are not necessary, we can omit search for a smaller J. This paper proposes a new version of SSF called SSF2, which performs MLP search by utilizing singular regions with J changed bidirectionally within a certain range. How to change J bidirectionally is not defined here, which is to be investigated.

3.1 General Flow of SSF2

The procedure of SSF2 is described below. Here $[J_{min}, J_{max}]$ is a range of J to examine, and up is a flag where up = 1 means J is in an increase phase and up = 0 means a decrease phase.

SSF2 (Singularity Stairs Following 2):

1. Set values for J, J_{min}, J_{max} and up.

2. Find solutions of MLP(J) from random starting points, and keep the best as the optimum of MLP(J), $\widehat{\theta}_J$.

3. Repeat the following:

3.1. if up = 1 then do (a), otherwise do (b).

(a) Apply reducibility mapping to $\widehat{\theta}_J$ to get singular regions, and find solutions of MLP(J+1) by starting from the regions. Store the best as the optimum $\widehat{\theta}_{J+1}$.

$J \leftarrow J + 1$. if J=$J_{\max}$, then $up \leftarrow 0$.

(b) Find solutions of MLP($J-1$) by starting

from $\widehat{\theta}_J$ and using the decrement method described in the next section, and store the best as the optimum $\widehat{\theta}_{J-1}$.
$J \leftarrow J-1$. if J=$J_{\min}$, then $up \leftarrow 1$.

3.2. if one of stopping criteria is satisfied, then stop.

3.2 Decrement Method

Decrement method decreases the number of hidden units, which is required in step 3.1 (b) stated above. Given $\widehat{\theta}_J$, the optimum of MLP(J), we want to find $\widehat{\theta}_{J-1}$, the optimum of MLP($J-1$).

One may consider just deleting one of the hidden units from $\widehat{\theta}_J$ and then starting search. However, such simple deletion is too naive to obtain an excellent solution, as shown in our experiments. Here we consider a method which guides search of MLP(J) from $\widehat{\theta}_J$ into a singular region, and then deletes the redundant hidden unit. Note that we have two kinds of singular regions.

First, we consider guiding search into singular region $\widehat{\Theta}_J^{\alpha\beta}$. Let m be the hidden unit to delete. Perform learning of MLP(J) with w_m and w_m fixed as follows.

$$w_m \leftarrow a \times \widehat{w}_m, \quad w_m \leftarrow a \times \widehat{w}_m \tag{9}$$

Here a is changed gradually from 1 to 0, guiding search into $\widehat{\Theta}_J^{\alpha\beta}$. After finishing search with a=0, we delete hidden unit m to get $\widehat{\theta}_{J-1}$. As for m, we can select a small number of candidates in ascending order of $|w_m|$ or try all cases.

Next, we consider guiding search into singular region $\widehat{\Theta}_J^{\gamma}$. Let m and n be the hidden units to merge. Perform learning of MLP(J) with w_m and w_n fixed as follows.

$$w_m \leftarrow b \times \widehat{w}_m + (1-b) \times \widehat{w}_n, \tag{10}$$
$$w_n \leftarrow b \times \widehat{w}_n + (1-b) \times \widehat{w}_m \tag{11}$$

Here b is changed from 1 to 0.5, guiding search into $\widehat{\Theta}_J^{\gamma}$. After finishing search of MLP(J) with b=0.5, update w_m as $w_m \leftarrow w_m + w_n$, delete hidden unit n, and then perform learning of MLP($J-1$) to get $\widehat{\theta}_{J-1}$. As for m and n, there are $J \times (J-1)/2$ combinations; thus, trying all cases would be unrealistic. We can select a small number of pairs in ascending order of distance $\|w_m - w_n\|$.

4 EXPERIMENTS

We evaluated the proposed SSF2 for the following sigmoidal MLP using artificial and real data. Here $\sigma(h) = 1/(1+\exp(-h))$.

$$f = w_0 + \sum_{j=1}^{J} w_j z_j, \quad z_j = \sigma(w_j^{\mathrm{T}} x) \tag{12}$$

For comparison, we used a quasi-Newton called BPQ and SSF1.3. The initial weights for BPQ are randomly selected from the range $[-1,+1]$ with the exception $w_0 = \overline{y}$. BPQ was repeated 100 times for each J.

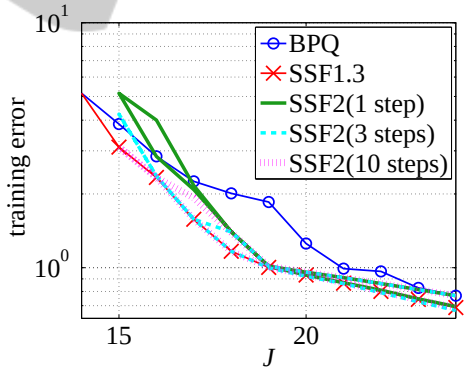
For SSF1.3 and the increase phase of SSF2, we set free parameters of the singular regions as follows. For $\widehat{\Theta}_J^{\alpha\beta}$ we set as w_{10}=0 and for $\widehat{\Theta}_J^{\gamma}$ we set q in the following as q=0.5, 1.0, and 1.5.

$$w_1 \leftarrow q \times \widehat{u}_m, \quad w_m \leftarrow (1-q) \times \widehat{u}_m \tag{13}$$

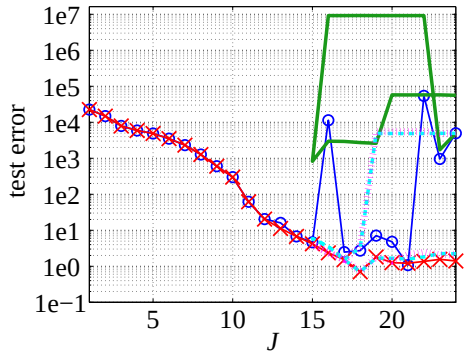
For SSF1.3 and SSF2, search pruning check was done for each 100 points during search.

We started SSF2 with J=J_{max} and up=0. The best solution obtained by running BPQ 20 times was selected as the optimum $\widehat{\theta}_J$ at the initial J (=J_{max}). In our experiments, SSF2 went in one cycle with J decreased from J_{max} to J_{min}, then increased from J_{min} to J_{max}, and stopped.

Each search was stopped when the iteration exceeded 10,000 sweeps, or the search step got smaller than 10^{-16}. Test error for artificial data was calculated



(a) training error



(b) test error

Figure 1: Solution quality for Experiment 1a.

using test data of 1,000 data points without noise, generated independently of training data. Test error for real data was evaluated using one segment among 10 segments; the remaining 9 segments were used for training.

4.1 Experiment using Artificial Data

Our artificial data set was generated using MLP having weights shown in the following equations. Values of variables $x_1, \cdots, x_{10}$ were randomly selected from the range $(0,1)$. Note that five variables $x_1, \cdots, x_5$ contribute to y, but the other five variables $x_6, \cdots, x_{10}$ are irrelevant. We included irrelevant variables to make the learning harder. Values of y were generated by adding small Gaussian noise $\mathcal{N}(0, 0.05^2)$ to MLP outputs. The sample size was 1,000 (N=1,000). We set J_{min}=15 and J_{max}=24, while the original J=22.

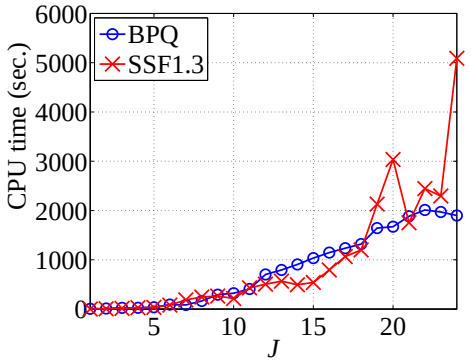
$$
\begin{aligned}
&[w_0, w_1, \cdots, w_{22}] \\
&\quad = [-11, -12, -10, -6, 4, 20, 18, -3, 12, -18, -17, 17, -1, \\
&\qquad 13, -9, 9, -3, 14, 1 - 18, 15, 12, 13], \\
&[w_1, w_2, \cdots, w_{22}] \\
&= \begin{bmatrix}
3 & -7 & -2 & 6 & -8 & 7 & -3 & 10 & -5 & 10 & -3 \\
-3 & 3 & -6 & 2 & -7 & -8 & 9 & 2 & 3 & 1 & -5 \\
9 & 1 & 4 & -1 & -4 & -3 & -8 & 2 & -4 & 1 & -9 \\
-8 & -8 & 3 & -6 & -7 & -10 & 1 & 5 & 0 & 2 & -4 \\
3 & -4 & 5 & 7 & -9 & -4 & -4 & 0 & 4 & 6 & -8 \\
-10 & -6 & -1 & 2 & 4 & 5 & -10 & -9 & -5 & -3 & 2
\end{bmatrix} \\
&\quad \begin{bmatrix}
-1 & 1 & 10 & -6 & -7 & 1 & -1 & 3 & -3 & 7 & -6 \\
0 & 0 & -8 & 2 & -7 & -4 & 3 & -3 & -7 & 3 & 10 \\
-5 & -10 & -8 & 9 & -7 & -9 & -2 & -2 & 1 & 4 & -1 \\
5 & 8 & 6 & -1 & 9 & -6 & 3 & -1 & 5 & -2 & 0 \\
6 & 2 & -2 & 2 & 10 & 4 & -10 & 8 & -9 & -5 & 3 \\
2 & -2 & -5 & 9 & -8 & 5 & 4 & -3 & 7 & 6 & 5
\end{bmatrix}
\end{aligned}
$$

In our experiments using artificial data, we examine how the performance of SSF2 is influenced by our decrement method. Below we examined in two ways 1a and 1b.

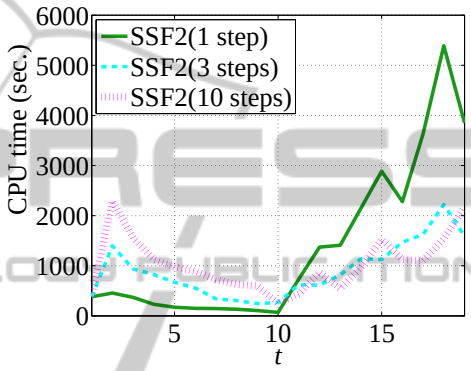
(1) Experiment 1a

First, in the decrement method, only singular region $\widehat{\Theta}_J^{\alpha\beta}$ was considered, and every hidden unit was tested as a candidate to delete. In eq. (9), we changed the number of guiding steps in three ways: one step, three steps ($a = 2/3, 1/3, 0$), and ten steps ($a = 9/10, 8/10, \cdots, 1/10, 0$). They are referred to as SSF2(1 step), SSF2(3 steps), and SSF2(10 steps) respectively. Note that SSF2(1 step) is nothing but simple deletion.

Figure 1 shows the solution quality of each method, showing the best training errors and corresponding test errors. Each SSF2 method made a round trip, at first decreasing J from 24 until 15, and then increasing J until 24, while SSF1.3 went on increasing J. SSF1.3 outperformed BPQ both in training and test. In each SSF2 method, the second half



(a) BPQ, SSF1.3



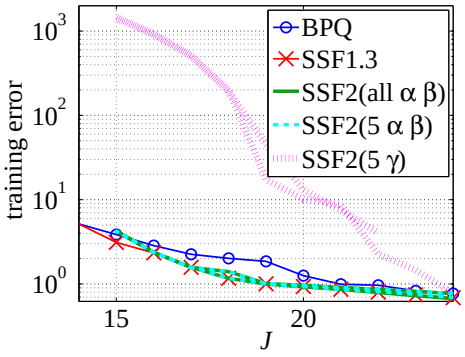
(b) SSF2

Figure 2: CPU time for Experiment 1a.

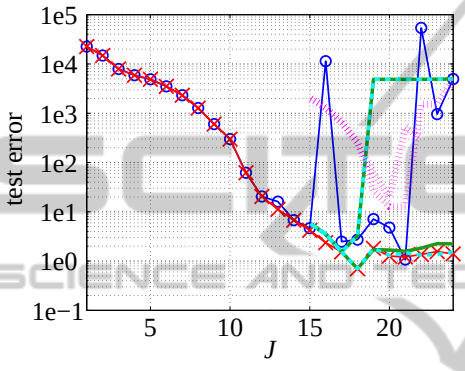
(increase phase) worked better than the first half (decrease phase) in both training and test. SSF2(1 step) worked rather poorly especially in test. However, SSF2(3 steps) and SSF2(10 steps) worked in much the same manner, and their increase phases were almost equivalent to that of SSF1.3. SSF2(3 steps), SSF2(10 steps) and SSF1.3 indicate J=18 is the best model, while BPQ indicates J=21 is the best.

Figure 2 shows CPU time required by each method. The horizontal axis t in Fig. 2(b) indicates how many times J was changed; thus, t=1,$\cdots$,10 corresponds to the decrease phase from J=24 until 15, and t=10,$\cdots$,19 means the increase phase from J=15 until 24. Each method has a tendency to require longer CPU time as J increases. Among SSF2 methods, SSF2(1 step) spent the longest because it required the largest number of search routes in its increase phase. The total CPU time of BPQ, SSF1.3, SSF2(1 step), SSF2(3 steps), and SSF2(10 steps) were 5h28m, 6h30m, 7h20m, 5h1m, and 6h1m respectively. Hence, both SSF2(3 steps) and SSF2(10 steps) were faster than SSF1.3.

Based on the results of Experiment 1a, we considered SSF(3 steps) as the most promising when using only singular region $\widehat{\Theta}_J^{\alpha\beta}$.

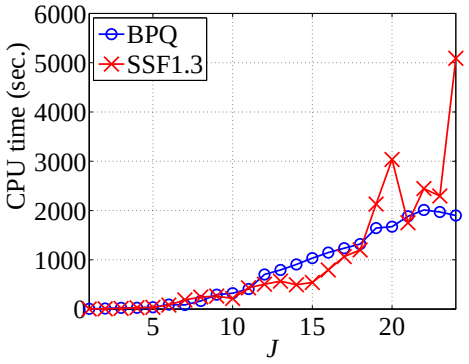


(a) training error

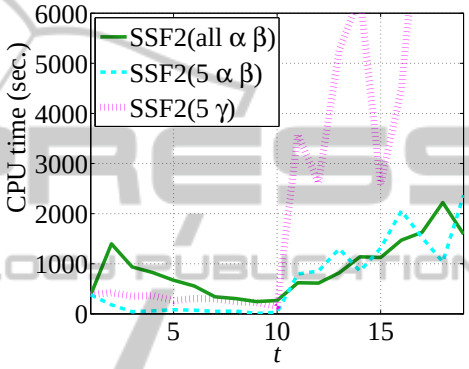


(b) test error

Figure 3: Solution quality for Experiment 1b.



(a) BPQ, SSF1.3



(b) SSF2

Figure 4: CPU time for Experiment 1b.

(2) Experiment 1b

Next, we examined two possibilities: i) as for singular region $\widehat{\Theta}_J^{\alpha\beta}$, what if we use only top five hidden units instead of using all, and ii) what if we use only top five for singular region $\widehat{\Theta}_J^{\gamma}$ instead of using $\widehat{\Theta}_J^{\alpha\beta}$. They are referred to as SSF2(5 αβ) and SSF2(5 γ) respectively, and will be compared with SSF2(all αβ) which is equivalent to SSF2(3 steps) in Experiment 1a.

Figure 3 shows the best training and test errors for each method. Each SSF2 method has two lines, indicating decrease and increase phases. The training and test errors of SSF2(5 γ) were rather large and quite unsatisfactory. Since it required huge CPU time, the processing was stopped at J=22 in the increase phase. The best training and test errors of SSF2(5 αβ) were much the same as SSF2(all αβ)=SSF2(3 steps) and also much the same as SSF1.3.

Figure 4 compares CPU time required by each method. The horizontal axis t in Fig. 4(b) indicates the same meaning as in Fig. 2. SSF2(5 γ) needed huge CPU time. The total CPU time of SSF2(all αβ), SSF2(5 αβ), and SSF2(5 γ) were 5h1m, 4h45m, and 10h9m respectively. Note that 4h45m of SSF2(5 αβ) was much shorter than 6h30m of SSF1.3.

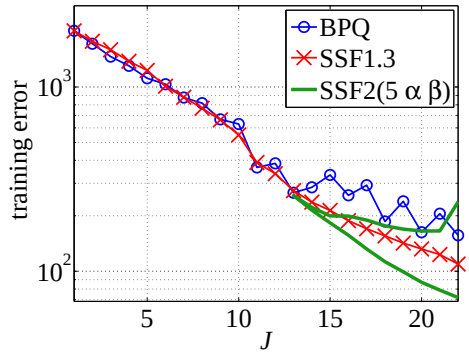
Experiments 1a and 1b showed SSF2(5 αβ) was the best among SSF2 methods we examined. Compared with SSF1.3, SSF2(5 αβ) showed much the same solution quality and was faster.

4.2 Experiment using Real Data

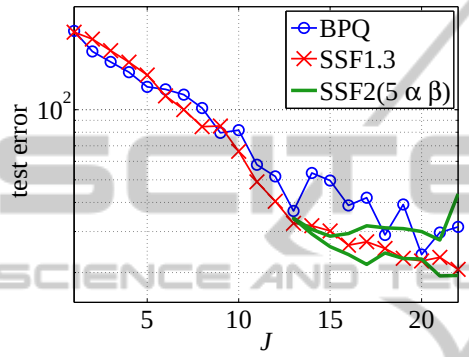
We evaluated SSF2 using Parkinson telemonitoring data (Little el al., 2007) available from UCI ML Repository. The number of variables is 18 (K=18) and the sample size is 5,875 (N=5,875). As for SSF2, we used SSF2(5 αβ), which performed the best in Experiment 1. Note that SSF2(5 αβ) employs only $\widehat{\Theta}_J^{\alpha\beta}$, only top five hidden units to delete, and three steps guiding into the singular region. We set J_{min}=13 and J_{max}=22.

Figure 5 shows the training and test errors for each method. Here again, SSF2(5 αβ) has two lines as in Fig. 3. From Fig. 5(a), we can see the best training errors of SSF2 considerably outperformed the other two. The training errors of SSF2 and SSF1.3 show preferable monotonic decrease, while those of BPQ showed up-and-down movement. From Fig. 5(b), we see that the best test errors of SSF2 also outperformed the other two.

Figure 6 shows CPU time required by each

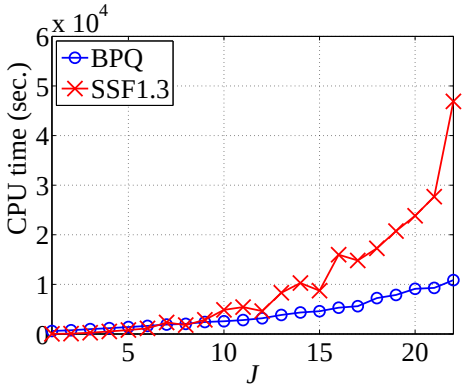


(a) training error

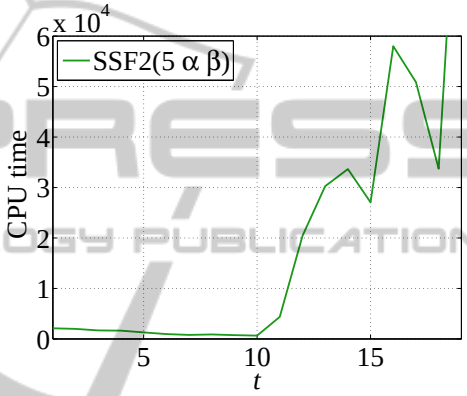


(b) test error

Figure 5: Solution quality for Experiment 2.



(a) BPQ, SSF1.3



(b) SSF2(5 αβ)

Figure 6: CPU time for Experiment 2.

method. The increase phase ($t \geq 10$) of SSF2 needed very long CPU time. The total CPU time of BPQ, SSF1.3, and SSF2 were 24h49m, 60h53m, and 105h16m respectively.

In this experiment, the solution quality of SSF2 was better than BPQ and SSF1.3, but SSF2 required more CPU time than the other two.

5 CONCLUSION

This paper proposed SSF2 which performs MLP search by making good use of singular regions with J changed bidirectionally within a certain range. The method was tuned and evaluated using artificial and real data. Our experiments using artificial data showed the tuned SSF2 showed much the same solution quality as the existing SSF1.3 with much smaller CPU time. In our experiment using real data, SSF2 resulted in better solutions with longer CPU time. In the future we will make the method even faster and examine how the range of J influences the performance.

ACKNOWLEDGEMENTS

This work was supported by Grants-in-Aid for Scientific Research (C) 25330294 and Chubu University Grant 26IS19A.

REFERENCES

Amari, S. (1998). Natural gradient works efficiently in learning. Neural Computation, 10 (2):251–276.

Duda, R.O., Hart, P.E. and Stork, D.G. (2001). *Pattern classification*. John Wiley & Sons, Inc., New York, 2nd edition.

Fukumizu, K. and Amari, S. (2000). Local minima and plateaus in hierarchical structure of multilayer perceptrons. Neural Networks, 13 (3):317–327.

Hecht-Nielsen, R. (1990). *Neurocomputing*. Addison-Wesley Publishing Company, Reading, Massachusetts.

Little, M.A., McSharry, P.E., Roberts, S.J., Costello D.A.E., Moroz, I.M. (2007). Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. BioMedical Engineering OnLine 2007, 6:23.

Nakano, R., Satoh, E. and Ohwaki, T. (2011). Learning method utilizing singular region of multilayer perceptron. Proc. 3rd Int. Conf. on Neural Comput. Theory and Appl., pp.106–111, 2011.

Saito, K. and Nakano, R. (1997). Partial BFGS update and efficient step-length calculation for three-layer neural networks. Neural Compututation, 9 (1): 239–257.

Satoh, S. and Nakano, R. (2012). Eigen vector descent and line search for multilayer perceptron. Proc. Int. Multi-Conf. of Engineers and Comput. Scientists, vol.1, pp.1–6.

Satoh, S. and Nakano, R. (2013). Fast and stable learning utilizing singular regions of multilayer perceptron. Neural Processing Letters, 38 (2): 99–115.

Satoh, S. and Nakano, R. (2014). Search pruning for a search method utilizing singular regions of multilayer perceptrons (in Japanese). IEICE Trans. on Information and Systems, J97-D (2):330–340.

Sussmann, H.J. (1992). Uniqueness of the weights for minimal feedforward nets with a given input-output map. Neural Networks, 5 (4):589–593.

Watanabe, S. (2008). A formula of equations of states in singular learning machines. Proc. Int. Joint Conf. on Neural Networks, pp.2099–2106, 2008.

Watanabe, S. (2009). *Algebraic geometry and statistical learning theory*. Cambridge Univ. Press, Cambridge.