A New Method for Learning RBF Networks by Utilizing Singular Regions

Seiya Satoh[1] and Ryohei Nakano[2]

[1] National Institute of Advanced Industrial Science and Technology
2–4–7 Aomi, Koto-ku, Tokyo, 135-0064 Japan
seiya.satoh@aist.go.jp
[2] Chubu University
1200 Matsumoto-cho, Kasugai, Aichi, 487–8501 Japan
nakano@cs.chubu.ac.jp

Abstract. The usual way to learn radial basis function (RBF) networks consists of two stages: first, select reasonable weights between input and hidden layers, and then optimize weights between hidden and output layers. When we learn multilayer perceptrons (MLPs), we usually employ the stochastic descent called backpropagation (BP) algorithm or 2nd-order methods such as pseudo-Newton method and conjugate gradient method. Recently new learning methods called singularity stairs following (SSF) methods have been proposed for learning real-valued or complex-valued MLPs by making good use of singular regions. SSF can monotonically decrease training error along with the increase of hidden units, and stably find a series of excellent solutions. This paper proposes a completely new method for learning RBF networks by introducing the SSF paradigm, and compares its performance with those of existing learning methods.

Keywords: neural networks, RBF networks, learning method, singular region, reducibility mapping

1 Introduction

In the search space of a multilayer perceptron, whether it is real-valued or complex-valued, there exist singular regions throughout which the gradient is zero; thus any search methods using gradient information cannot move in the regions [5] [8]. There exist singular regions in the search space of radial basis function (RBF) networks as well.

It is common for learning RBF networks to employ two-stage learning; first, select appropriate weights between input and hidden layers, and then optimize weights between hidden and output layers [2]. Moreover, the EM algorithm [3] was used considering RBF networks as a mixture model [6].

In the learning of MLPs, however, whole weights are updated all at once by using the backpropagation algorithm or pseudo-Newton method. Recently new learning methods called singularity stairs following (SSF) methods have been

proposed for learning real-valued or complex-valued MLPs by making good use of singular regions [9] [10]. SSF can decrease training error monotonically as the number of hidden units increases, and can stably find a series of solutions better than those obtained by existing learning methods.

This paper proposes a completely new method for learning RBF networks by applying the SSF paradigm to the RBF framework and compares its performance with those of the two-stage learning and pseudo-Newton method.

2 Background

2.1 RBF Networks

RBF(J) denotes an RBF network with J hidden units and one output unit. In RBF(J) model, let $\boldsymbol{w}_j^{(J)} = (w_{j1}^{(J)}, \cdots, w_{jK}^{(J)})^{\mathrm{T}}$ be weights between input and hidden unit j $(= 1, \cdots, J)$, and let $v_j^{(J)}$ be a weight between hidden unit j and the single output unit. When Gaussian basis function is adopted and μ-th data point $\boldsymbol{x}^\mu = (x_1^\mu, \cdots, x_K^\mu)^{\mathrm{T}}$ is given as input, the output of RBF(J) can be defined as below. Here σ_j is a parameter of Gaussian basis function at hidden unit j.

$$f_J^\mu = v_0^{(J)} + \sum_{j=1}^{J} v_j^{(J)} \exp\left(-\frac{||\boldsymbol{x}^\mu - \boldsymbol{w}_j^{(J)}||^2}{2(\sigma_j^{(J)})^2}\right) \tag{1}$$

The whole parameter vector of RBF(J) is given below:
$\boldsymbol{\theta}^{(J)} = \left(v_0^{(J)}, \cdots, v_J^{(J)}, \left(\boldsymbol{w}_1^{(J)}\right)^{\mathrm{T}}, \cdots, \left(\boldsymbol{w}_J^{(J)}\right)^{\mathrm{T}}, \sigma_1^{(J)}, \cdots, \sigma_J^{(J)}\right)^{\mathrm{T}}$.
Given training data $\{(\boldsymbol{x}^\mu, y^\mu),\ \mu = 1, \cdots, N\}$, the target function of RBF(J) learning is given as follows.

$$E_J = \frac{1}{2}\sum_{\mu=1}^{N} \left(\delta_J^\mu\right)^2, \qquad \delta_J^\mu \equiv f_J^\mu - y^\mu \tag{2}$$

2.2 Existing Methods for Learning RBF Networks

Most ways to learn RBF networks can be classified into two kinds: one is two-stage learning and the other is one-stage learning.

In the former, under the condition that $\sigma_1^{(J)}, \cdots, \sigma_J^{(J)}$ are fixed to a certain common constant such as $1/\sqrt{2}$ throughout learning, first, select reasonable $\boldsymbol{w}_1^{(J)}, \cdots, \boldsymbol{w}_J^{(J)}$, and then, optimize $v_0^{(J)}, \cdots, v_J^{(J)}$ [2].

In the latter, whole weights are optimized at the same time by using gradient-based methods or the EM algorithm. Gradient-based methods can be classified into the stochastic descent and 2nd-order method such as pseudo-Newton method.

Two-Stage Learning At the first stage of two-stage learning, one can apply clustering to explanatory data $\{\boldsymbol{x}^\mu\}$ to get J clusters, and then treat all the centroids as $\boldsymbol{w}_1^{(J)}, \cdots, \boldsymbol{w}_J^{(J)}$.

This paper, however, employs function newrb supported in Neural Network Toolbox of MATLAB R2015b. The newrb algorithm gradually expands RBF networks by increasing the number J of hidden units one by one. At each cycle of model expansion, data point $\boldsymbol{x}^\mu$ that has the largest value of output error $|f_{J-1}^\mu - y^\mu|$ is added as $\boldsymbol{w}_J$. Basis function parameters $\{\sigma_j\}$ are not optimized in this learning.

The general flow of the newrb algorithm is shown below. Let $J_{\max}$ be the maximum number of hidden units.

newrb algorithm

$v_0 \leftarrow (1/N) \sum_{\mu=1}^N y^\mu$
for $J = 1, \cdots, J_{\max}$ **do**
  Compute outputs $f_{J-1}^\mu,\ \mu = 1, \cdots, N$.
  $i \leftarrow \mathrm{argmax}_{\mu \in \{1,\cdots,N\}} |f_{J-1}^\mu - y^\mu|$
  $\boldsymbol{w}_J^{(J)} \leftarrow \boldsymbol{x}^i$
  With $\boldsymbol{w}_1^{(J)}, \cdots, \boldsymbol{w}_J^{(J)}$ fixed, optimize $v_0^{(J)}, \cdots, v_J^{(J)}$.
end for

One-Stage Learning One-stage learning can be classified into gradient-based and mixture-based [6]. Among many gradient-based learning methods, this paper employs the pseudo-Newton method with the BFGS update [4] since it has more powerful learning ability than the stochastic descent. Here we consider two kinds of methods: one is simply called BFGS, and the other is called BFGS(σ). The former optimizes all the weights except $\{\sigma_j\}$, while the latter optimizes both all the weights and $\{\sigma_j\}$ at the same time.

2.3 Singular Regions of RBF Networks

The search space of MLP, whether it is real-valued or complex-valued, has a continuous region where input-output equivalence holds and the gradient is zero [5] [8]. Such regions can be generated by reducibility mapping $\alpha\beta$ or γ [9] [10]. We call such a region a singular region. The search space of RBF networks has also singular regions, which are generated only by γ reducible mapping.

Below we explain how to generate a singular region of RBF(J) based on the optimal solution of RBF($J-1$). The optimal solution must satisfy the following, where E_{J-1} denotes the target function of RBF($J-1$) learning, and $\boldsymbol{\theta}^{(J-1)}$ denotes the whole parameter vector of RBF($J-1$).

$$\frac{\partial E_{J-1}}{\partial \boldsymbol{\theta}^{(J-1)}} = \boldsymbol{0} \tag{3}$$

This can be broken down element-wise as follows, where $j = 1, \cdots, J-1$.

$$\frac{\partial E}{\partial v_0^{(J-1)}} = \sum_{\mu=1}^{N} \delta_{J-1}^{\mu} = 0 \tag{4}$$

$$\frac{\partial E}{\partial v_j^{(J-1)}} = \sum_{\mu=1}^{N} \delta_{J-1}^{\mu} \exp\left(-\frac{||\boldsymbol{x}^{\mu} - \boldsymbol{w}_j^{(J-1)}||^2}{2(\sigma_j^{(J-1)})^2}\right) = 0 \tag{5}$$

$$\frac{\partial E}{\partial \boldsymbol{w}_j^{(J-1)}} = \sum_{\mu=1}^{N} \delta_{J-1}^{\mu} v_j^{(J-1)} \exp\left(-\frac{||\boldsymbol{x}^{\mu} - \boldsymbol{w}_j^{(J-1)}||^2}{2(\sigma_j^{(J-1)})^2}\right) \frac{\boldsymbol{x}^{\mu} - \boldsymbol{w}_j^{(J-1)}}{\left(\sigma_j^{(J-1)}\right)^2} = \boldsymbol{0} \tag{6}$$

$$\frac{\partial E}{\partial \sigma_j^{(J-1)}} = \sum_{\mu=1}^{N} \delta_{J-1}^{\mu} v_j^{(J-1)} \exp\left(-\frac{||\boldsymbol{x}^{\mu} - \boldsymbol{w}_j^{(J-1)}||^2}{2(\sigma_j^{(J-1)})^2}\right) \frac{||\boldsymbol{x}^{\mu} - \boldsymbol{w}_j^{(J-1)}||^2}{\left(\sigma_j^{(J-1)}\right)^3} = 0 \tag{7}$$

Let the optimal solution $\widehat{\boldsymbol{\theta}}^{(J-1)}$ of RBF$(J-1)$ be
$\left(\widehat{v}_0^{(J-1)}, \cdots, \widehat{v}_J^{(J-1)}, \ \left(\widehat{\boldsymbol{w}}_1^{(J-1)}\right)^{\mathrm{T}}, \cdots, \left(\widehat{\boldsymbol{w}}_J^{(J-1)}\right)^{\mathrm{T}}, \widehat{\sigma}_1^{(J-1)}, \cdots, \widehat{\sigma}_J^{(J-1)}\right)^{\mathrm{T}}$.

Now apply reducibility mapping γ to the optimal solution $\widehat{\boldsymbol{\theta}}^{(J-1)}$ to get singular region $\widehat{\boldsymbol{\Theta}}_{\gamma}^{(J)}$. Here $m \in \{2, \cdots, J\}$.

$$\widehat{\boldsymbol{\theta}}^{(J-1)} \quad \xrightarrow{\gamma} \quad \widehat{\boldsymbol{\Theta}}_{\gamma}^{(J)}$$

$$\begin{aligned}
\widehat{\boldsymbol{\Theta}}_{\gamma}^{(J)} \equiv \{\boldsymbol{\theta}^{(J)} |\ & v_0^{(J)} = \widehat{v}_0^{(J-1)}, v_1^{(J)} + v_m^{(J)} = \widehat{v}_{m-1}^{(J-1)}, \\
& \boldsymbol{w}_1^{(J)} = \boldsymbol{w}_m^{(J)} = \widehat{\boldsymbol{w}}_{m-1}^{(J-1)}, \sigma_1^{(J)} = \sigma_m^{(J)} = \widehat{\sigma}_{m-1}^{(J-1)}, \\
& v_j^{(J)} = \widehat{v}_{j-1}^{(J-1)}, \boldsymbol{w}_j^{(J)} = \widehat{\boldsymbol{w}}_{j-1}^{(J-1)}, \sigma_j^{(J)} = \widehat{\sigma}_{j-1}^{(J-1)}, \\
& \text{for } j \in \{2, \cdots, J\} \backslash \{m\}, \quad m = 2, \cdots, J\}
\end{aligned} \tag{8}$$

Note that $v_1^{(J)}$ and $v_m^{(J)}$ cannot be determined uniquely since they only have the following constraint.

$$v_1^{(J)} + v_m^{(J)} = \widehat{v}_{m-1}^{(J-1)} \tag{9}$$

This equation can be rewritten using parametric variable q.

$$v_1^{(J)} = q\ \widehat{v}_{m-1}^{(J-1)}, \qquad v_m^{(J)} = (1-q)\ \widehat{v}_{m-1}^{(J-1)} \tag{10}$$

3 SSF for RBF Networks

A new search paradigm called Singularity Stairs Following (SSF) stably finds a series of excellent solutions by making good use of singular regions. So far, two series of SSF methods have been proposed: The latest versions are SSF1.4 [9] for real-valued MLPs and C-SSF1.3 [10] for complex-valued MLPs.

By applying the SSF paradigm to RBF networks, we have a completely new learning method called RBF-SSF. RBF-SSF optimizes $\{\sigma_j\}$ as well.

3.1 General Flow of RBF-SSF

The general flow of RBF-SSF is shown below. Let $J_{\max}$ be the maximum number of hidden units.

General flow of RBF-SSF

1: Get the best solution of RBF(J =1) by repeating search with different initial values.
2: **for** $J = 2, \cdots, J_{\max}$ **do**
3: Select starting points from the singular region obtained by applying reducibility mapping γ to the best solution of RBF($J-1$).
4: Calculate Hessian matrix at each starting point, calculate eigenvalues and eigenvectors of the Hessian, and select eigenvectors corresponding to each negative eigenvalue.
5: Repeat search predefined times from the starting points in the direction and in the opposite direction of a negative eigenvector selected at Step 4, and get the best solution of RBF(J).
6: **end for**

In the following experiments, the starting points from the singular region at the above Step 3 are obtained by setting q to $0.5, 1.0, 1.5$ in eq.(10). These three points correspond to interpolation, boundary, and extrapolation. Since m is changed in the range of $2, \cdots, J$, the total number of starting points in the search of RBF(J) amounts to $3 \times (J-1)$.

3.2 Techniques for Making SSF Faster

As described above, the number of starting points of RBF(J) gets large as J gets large. Moreover, the dimension of the search space also gets large as J gets large. Thus, if we perform search for every negative eigenvalue for large J, the number of searches gets very large, and consequently the processing time becomes huge. Hence, we introduce two accelerating techniques into RBF-SSF: one is search pruning and the other is to set upper limit to the number of searches.

Search pruning is an accelerating technique which discards a search if the search is found to proceed along much the same route experienced before [9].

Setting upper limit to the number of searches is an accelerating technique which selects the predefined number of starting points based on the preference.

Here preference is given to smaller negative eigenvalues since such eigenvalues indicate bigger drop from a search point. In the following experiments the upper limit is set to be 100.

4 Experiments

The performance of RBF-SSF was compared with three other learning methods. Tabel 1 shows the settings of these four learning methods. BFGS and BFGS(σ) are repeated 100 times changing initial values of weights. Note that both BFGS(σ) and RBF-SSF adapt $\{\sigma_j\}$ through learning.

Table 1. Learning methods.

Learning method	Description ($j = 1, \cdots, J$)
newrb	a method of two-stage learning included in Neural Network Toolbox version 9.1 $\sigma_j^{(J)}$: fixed to be $1/\sqrt{2}$
BFGS	$v_j^{(J)}, \boldsymbol{w}_j^{(J)}$: optimized by BFGS the initial value of $v_j^{(J)}$: 0 the initial value of $\boldsymbol{w}_j^{(J)}$: randomly selected from $\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(N)}$ $\sigma_j^{(J)}$: fixed to be $1/\sqrt{2}$
BFGS(σ)	$v_j^{(J)}, \boldsymbol{w}_j^{(J)}, \sigma_j^{(J)}$: optimized by BFGS(σ) the initial value of $v_j^{(J)}$: 0 the initial value of $\boldsymbol{w}_j^{(J)}$: randomly selected from $\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(N)}$ the initial value of $\sigma_j^{(J)}$: $1/\sqrt{2}$
RBF-SSF	$v_j^{(J)}, \boldsymbol{w}_j^{(J)}, \sigma_j^{(J)}$: optimized by RBF-SSF

The following three datasets were used: engine behavior dataset, building energy dataset, and Parkinsons telemonitoring dateset [7]. The first two are included in MATLAB Neural Network Toolbox. The third was obtained from UCI ML Repository [1]. Each dataset was normalized as follows, where y_{mean} and y_{std} are the average and standard deviation of $\{y^\mu\}$ respectively.

$$\widetilde{x}_k^\mu \leftarrow \frac{x_k^\mu}{\max_\mu(|x_k^\mu|)}, \qquad \widetilde{y}^\mu \leftarrow \frac{y^\mu - y_{\text{mean}}}{y_{\text{std}}} \tag{11}$$

4.1 Experiments Using Engine Behavior Dataset

The problem for this dataset is to estimate an engine torque from two variables: fuel use and speed. The number N of data points is 1199; 90 % was used for training, and the remaining 10 % was used for test. The number J of hidden units for newrb was changed as $10, 20, \cdots, 200$, while J of BFGS, BFGS(σ), and RBF-SSF was changed as $1, 2, \cdots, 40$.

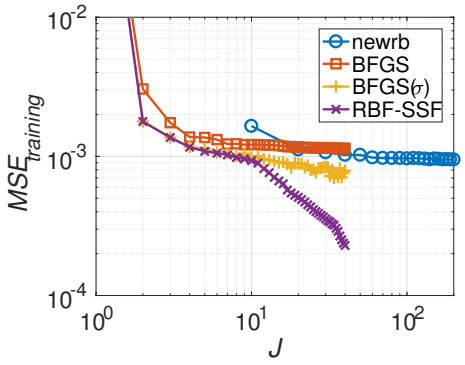

Fig. 1. Training error for engine behavior dataset.

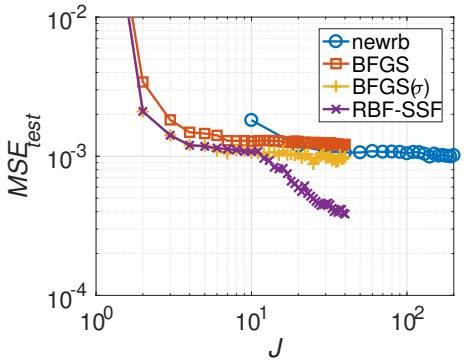

Fig. 2. Test error for engine behavior dataset.

Figures 1 and 2 show minimum training and test errors of each method respectively. Each error is shown in mean squared error (MSE). RBF-SSF and newrb monotonically decreased training error. The final training and test errors of RBF-SSF were the smallest among the four methods, a few times smaller than those of the other three methods. The final training and test errors of BFGS(σ) were smaller than those of BFGS, which means the adaptation of $\{\sigma_j\}$ worked to some extent.

Figure 3 shows the processing time spent by each method at each J. The total time (h:min:sec) spent by newrb, BFGS, BFGS(σ), and RBF-SSF were 00:00:05, 01:30:56, 02:08:15, and 01:48:40 respectively. The fastest was newrb and the slowest was BFGS(σ). BFGS(σ) was 4/3 times slower than BFGS since BFGS(σ) had to optimize $\{\sigma_j\}$ aside from weights.

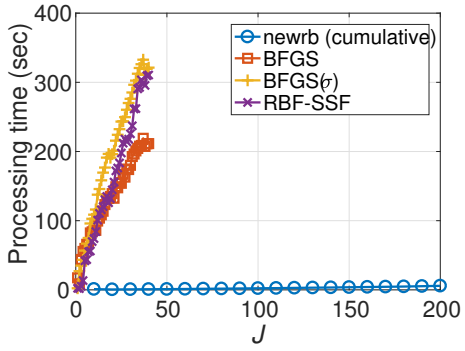

Fig. 3. Processing time for engine behavior dataset.

4.2 Experiment Using Building Energy Dataset

The problem for this dataset is to estimate the energy use of a building from 14 variables such as time and weather conditions. The number N of data points is 4208; 90 % was used for training, and the remaining 10 % was used for test. The number J of hidden units for newrb was changed as $10, 20, \cdots, 400$, while J of BFGS, BFGS(σ), and RBF-SSF was changed as $1, 2, \cdots, 30$.

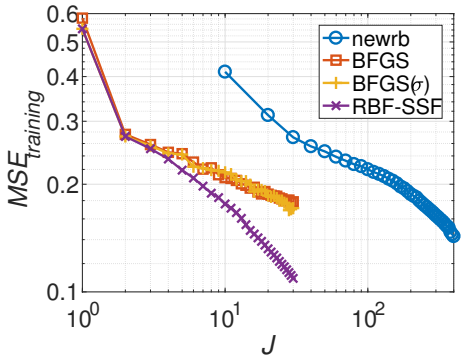

Fig. 4. Training error for building energy dataset.

Figures 4 and 5 show minimum training and test errors of each method respectively. Again, RBF-SSF and newrb monotonically decreased training error. It is clear that the final training and test errors of RBF-SSF were the smallest among the four. The final training and test errors of BFGS(σ) were much the same as those of BFGS, which means the adaptation of $\{\sigma_j\}$ did not work well here. The final training error at $J = 400$ of newrb was smaller than that of

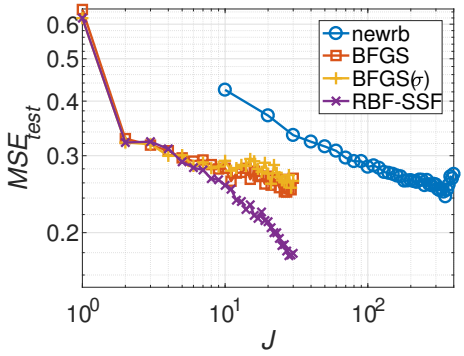

Fig. 5. Test error for building energy dataset.

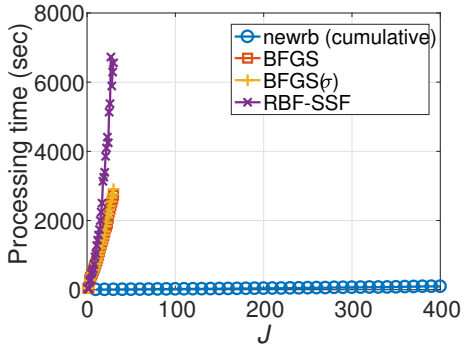

Fig. 6. Processing time for building energy dataset.

BFGS. The best test errors of newrb at $J = 350$ and BFGS(σ) at $J = 29$ were much the same.

Figure 6 shows the processing time spent by each method at each J. The total time (h:min:sec) spent by newrb, BFGS, BFGS(σ), and RBF-SSF Were 00:01:34, 10:19:41, 10:51:10, and 21:56:55 respectively. The fastest was newrb and the slowest was RBF-SSF. RBF-SSF spent the longest time since it had to compute the Hessian repeatedly and the processing time gets even larger as the number of input units gets large.

4.3 Experiment Using Parkinsons Telemonitoring Dataset

The problem for this dataset is to estimate motor UPDRS (unified Parkinson ' s disease rating score) based on 18 variables such as voice measures. The number N of data points is 5875; 90 % was used for training, and the remaining 10 % was used for test. The number J of hidden units for newrb was changed

as $10, 20, \cdots, 350$, while J of BFGS, BFGS(σ), and RBF-SSF was changed as $1, 2, \cdots, 20$.

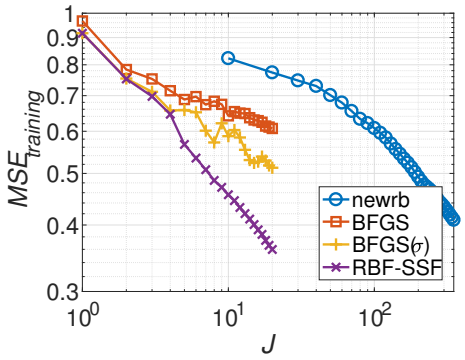

Fig. 7. Training error for Parkinsons telemonitoring dataset.

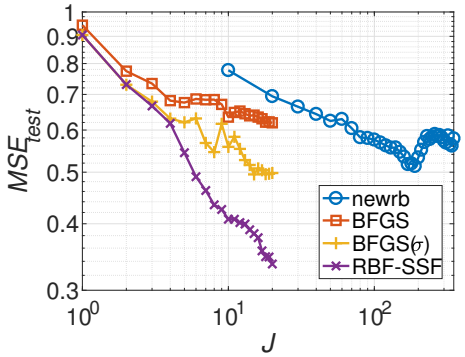

Fig. 8. Test error for Parkinsons telemonitoring dataset.

Figures 7 and 8 show minimum training and test errors of each method respectively. Again, RBF-SSF and newrb decreased training error monotonically. It is clear the final training and test errors of RBF-SSF were the smallest. The final training and test errors of BFGS(σ) were clearly smaller than those of BFGS, which means the adaptation of $\{\sigma_j\}$ worked well here. The final training error at $J = 400$ of newrb was obviously smaller than that of BFGS. The best test errors of newrb at $J = 190$ and BFGS(σ) at $J = 15$ were much the same.

Figure 9 shows the processing time spent by each method at each J. The total time (h:min:sec) spent by newrb, BFGS, BFGS(σ), and RBF-SSF Were

00:03:27, 10:42:19, 10:49:15, 19:03:27 respectively. Again the fastest was newrb, and the slowest was RBF-SSF. RBF-SSF spend the longest time due to the repeated computation of the Hessian.

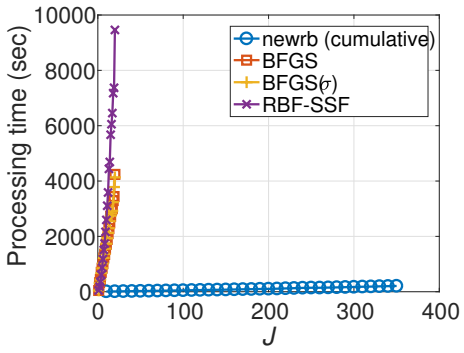

Fig. 9. Processing time for Parkinsons telemonitoring dataset.

4.4 Findings Obtained from Experiments

The following findings can be made from our experiments.
(1) Training Errors

RBF-SSF could decrease training error monotonically and in a striking manner, much smaller than those of the other three methods. BFGS(σ) clearly exceeded BFGS in training error for two datasets out of three, which means the adaptation of $\{\sigma_j\}$ is a meaningful step. The two-stage learning newrb decreased training error monotonically , but the decreasing rate is relatively small, and it needs hundreds of hidden units to seriously decrease training error.
(2) Test Errors

The tendency of test error decreasing curves by four methods were much the same as that of training error. RBF-SSF could decrease test error to a remarkable extent, much smaller than those of the other three methods. RBF-SSF showed a negative gradient at the largest J for every dataset, which indicates the optimal model for each dataset may have $J(> J_{max})$. This may suggest the presumable tendency that RBF networks are unlikely to cause the overfitting. BFGS(σ) clearly exceeded BFGS in test error for two datasets out of three, which means the adaptation of $\{\sigma_j\}$ is a meaningful step. Test error of newrb showed U-shaped curve as J got large for two datasets out of three. This indicates the overfitting happened in this type of learning.
(3) Processing Time

Two-stage learning newrb was overwhelmingly fast since it only solves linear problems repeatedly. On the other hand, the other three methods required considerable amount of time since they use BFGS updates so many times. BFGS(σ)

was slower than BFGS because BFGS(σ) has to adapt $\{\sigma_j\}$ aside from weights. Moreover, RBF-SSF has to compute the Hessian repeatedly; thus RBF-SSF spent the longest time.

5 Conclusion

This paper proposed a completely new method for learning RBF networks by applying the SSF paradigm to the RBF framework, and compared its performance with those of three existing methods. The proposed RBF-SSF could decrease training and test errors to a remarkable extent, much smaller than those of the other three methods. However, it required the longest processing time.

In the future, we plan to accelerate RBF-SSF even more and apply it to applications which strongly require the advantages of RBF networks.

Acknowledgment

This work was supported by Grants-in-Aid for Scientific Research (C) 16K00342.

References

1. UCI Machine Learning Repository. *http://archive.ics.uci.edu/ml/*, 1996.
2. C. M. Bishop. *Neural networks for pattern recognition.* Clarendon Press, 1995.
3. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B*, 39:1–38, 1977.
4. R. Fletcher. *Practical methods of optimization, 2nd edition.* John Wiley & Sons, 1987.
5. K. Fukumizu and S. Amari. Local minima and plateaus in hierarchical structure of multilayer perceptrons. *Neural Networks*, 13(3):317–327, 2000.
6. M. Làzaro, I. Santamarìa, and C. Pantaleòn. A new EM-based training algorithm for RBF networks. *Neural Networks*, 16:69–77, 2003.
7. M.A. Little, P.E. McSharry, S.J. Roberts, D.A.E. Costello, and I.M. Moroz. Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *Biomedical Engineering OnLine*, 6(23), 2007.
8. T. Nitta. Local minima in hierarchical structures of complex-valued neural networks. *Neural Networks*, 43:1–7, 2013.
9. S. Satoh and R. Nakano. Multilayer perceptron learning utilizing singular regions and search pruning. In *Proc. Int. Conf. on Machine Learning and Data Analysis*, pages 790–795, 2013.
10. S. Satoh and R. Nakano. A yet faster version of complex-valued multilayer perceptron learning using singular regions and search pruning. In *Proc. of 7th Int. Joint Conf. on Computational Intelligence (IJCCI)*, volume 3 NCTA, pages 122–129, 2015.