XXXXXXXXXXXXXXXX
doi.XX.XXXX/xxxxxxxxx/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

Subject Section

Bearing Fault Diagnosis Based on Deep Belief Network

Wujie Sun [1,*]

[1] School of Software Engineering, South China University of Technology, Guangzhou 510006, P.R. China.

[*] To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Bearing failures will bring huge economic and time loss. Therefore, it is important to judge whether the bearing is faulty and the possible fault location based on the acceleration of the bearing. However, the acceleration of the bearing is affected by noise, and it is difficult to rely on the observation to find its law. The problem can be solved by using wavelet packet decomposition and deep belief network. In this paper, I use wavelet packet decomposition for noise reduction, and extract features from the noise-reduced data and raw data, and then use the deep belief network (DBN) for training and classification. The experimental results show its effectiveness.

Keywords: Deep belief network, fault diagnosis, wavelet packet decomposition, feature extraction
Contact: wjsunscut@163.com

1 Introduction

Mechanical bearing failures are common in daily life, and the economic and time loss they cause is very serious. Therefore, it is very important to find it early in the event of a failure. Accelerometer data collected from actual operating environment is an important indicator to determine whether the bearing is faulty. However, the acceleration of the bearing is affected by noise, and it is difficult to rely on observation to find the problem. Therefore, we need a method that can efficiently determine whether the bearing is faultly and its fault type through the accelerometer data.

Since the accelerometer data is affected by noise, we need to perform noise reduction. Wavelet packet decomposition [1], [2] is widely used for noise reduction and has achieved remarkable results.

In recent years, deep learning, as an emerging method in the field of machine learning, has achieved brilliant results in the fields of image and speech recognition with its powerful capabilities. The deep learning method has the following advantages compared with the traditional fault diagnosis methods: 1) Deep learning has powerful feature extraction ability, which can automatically extract features from a large amount of data, and reduces the needs of the expert experience and signal processing technology. It reduces the uncertainty of feature extraction and fault diagnosis caused by human in traditional methods; 2) By establishing a deep model, it can well represent the complex mapping relationship between signal and health status, which is very suitable for big data background. The need for diagnostic analysis of diverse, nonlinear, and high-dimensional health monitoring data. Therefore, applying deep learning to the field of fault diagnosis has certain timeliness, practicability and versatility.

As one of the classical algorithms of deep learning, deep belief network [3] successfully solves problems such as information retrieval, dimension reduction, fault classification and so on with its excellent feature extraction and training algorithms. In addition, deep belief network has better scalability and mapping capabilities than other machine learning algorithms such as support vector machine [4] and back propagation neural network [5].

For the input data of the deep belief network, some choose to use the time domain signal as the input [6], and some choose to use the frequency domain signal as the input [7]. In this paper, I choose to extract features from the time domain [8] as the input to the deep belief network. Documents and code can be found on https://github.com/wjsunscut/Intelligent-Software-Project-Training.

2 Wavelet Packet Decomposition and Feature Extraction

Wavelet packet decomposition uses an analysis tree to represent wavelet packets, that is, using multiple iterations of wavelet transform to analyze the details of the input signal. From the perspective of function theory, wavelet packet decomposition is to map the signal into the space formed by the wavelet packet basis function. From the perspective of signal

processing, it is to pass the signal through a series of filters with different center frequencies but the same bandwidth.

Since the wavelet decomposition [9] is poor in frequency resolution in the high frequency band and time resolution in the low frequency band, wavelet packet decomposition is proposed to overcome this problem. It is a more sophisticated method of signal analysis that improves the time domain resolution of the signal. Fig. 1 shows the difference between them.

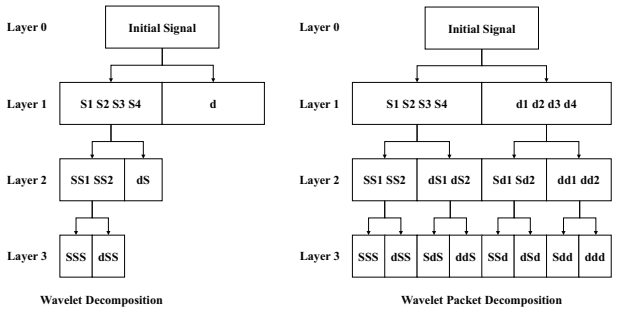

Fig. 1. Difference between wavelet decomposition and wavelet packet decomposition.

Since wavelet packet decomposition is not the focus point of my project training, I will not describe it too much in this paper. Wavelet packet decomposition can be easily implemented using functions such as wavedec, appcoef, detcoef, wbmpen, and wdencmp in MATLAB (The specific code associated with the wavelet packet decomposition is in $preprocessing.m$ in the $Code$ folder).

Through the wavelet packet decomposition, we obtained the noise-reduced data. Next, we should extract features from the data. We need to set a sliding window to read the data by its length. For example, if there are 50,000 data and the sliding window length is 1000 and the number of statistical features is 100, then we can get 50 data with each containing 100 parameters. Table 1 shows how to calculate the statistical features. N is the length of sliding window and x_i is the i^{th} accelerometer data in the sliding window. More discriminating parameters can help to improve the accuracy of the classification. Limited to the paper space, only 18 statistical features related to the time domain are provided in the paper.

Table 1. Statistical Features

Name	Formula
Peak-to-Peak	$\mathrm{MAX}\lvert x_i\rvert - \mathrm{MIN}\lvert x_i\rvert$
Peak	$\mathrm{MAX}\lvert x_i\rvert$
Mean	$\frac{1}{N}\sum_i^N x_i$
Mean Square	$\frac{1}{N}\sum_i^N x_i^2$
Root Mean Square	$\sqrt{\frac{1}{N}\sum_i^N x_i^2}$
Mean Amplitude	$\frac{1}{N}\sum_i^N \lvert x_i\rvert$
Square Mean Root	$\left(\frac{1}{N}\sum_i^N \lvert x_i\rvert^{1/2}\right)^2$
Variance	$\frac{1}{N}\sum_i^N (x_i-\overline{x})^2$
Standard Deviation	$\sqrt{\frac{1}{N}\sum_i^N (x_i-\overline{x})^2}$
Skewness	$\frac{1}{N}\sum_i^N x_i^3$
Kurtosis	$\frac{1}{N}\sum_i^N \lvert x_i\rvert^4$
Skewness Factor	$\frac{1}{N}\sum_i^N \lvert x_i\rvert^3 / \left(\sqrt{\frac{1}{N}\sum_i^N x_i^2}\right)^3$
Kurtosis Factor	$\frac{1}{N}\sum_i^N \lvert x_i\rvert^4 / \left(\sqrt{\frac{1}{N}\sum_i^N x_i^2}\right)^4$
Peak Factor	$\mathrm{MAX}\lvert x_i\rvert / \sqrt{\frac{1}{N}\sum_i^N x_i^2}$
Impulsive Factor	$\mathrm{MAX}\lvert x_i\rvert / \frac{1}{N}\sum_i^N \lvert x_i\rvert$
Clearance Factor	$\mathrm{MAX}\lvert x_i\rvert / \left(\frac{1}{N}\sum_i^N \lvert x_i\rvert^{1/2}\right)^2$
Waveform Factor	$\sqrt{\frac{1}{N}\sum_i^N x_i^2} / \frac{1}{N}\sum_i^N \lvert x_i\rvert$
Waveform Entropy [10]	$\frac{1}{N}\sum_i^N x_i \log x_i$

3 Deep Belief Network

A common DBN model [11] proposed by Hinton consists of a classifier and multiple restricted Boltzmann machines (RBMs) as shown in Fig. 2. If we want to figure out the principles of the DBN, first we need to master the knowledge related to RBM.

3.1 Restricted Boltzmann Machine

RBM [12] has a visible layer, a hidden layer, but there is no connection within the layer, and is fully connected between the layers. And each layer unit commonly takes a value of 0 or 1. It removes the intra-layer connection of the Boltzmann machine (BM), which greatly reduces the amount of calculation. The structure of the RBM is shown in Fig. 3.

In RBM, the activation conditions of each hidden layer units are independent when the visible layer units' state is given. Similarly, the activation conditions of the visible layer units are independent when the hidden layer units' state is given.

The essence of RBM is a useful tool for unsupervised learning, which can be used for dimensionality reduction, feature extraction, autoencoder, deep belief network, and so on.

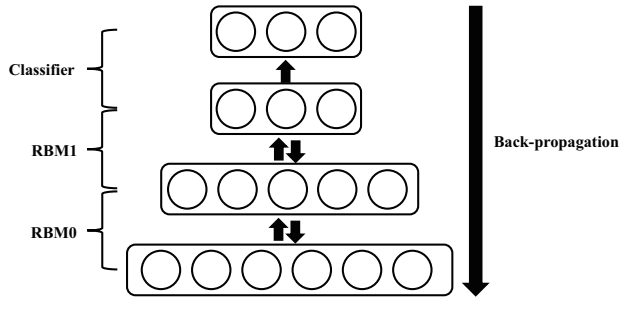

Fig. 2. A simple DBN model with two RBMs and one classifier.

Next I will formulate the formula according to [13]. Assume the RBM has n visible layer units and m hidden layer units. We use the vectors $\mathbf{v}$ and $\mathbf{h}$ to represent the states of its visible layer and hidden layer where v_i is the i^{th} visible layer unit and h_j is the j^{th} hidden layer unit. Therefore,

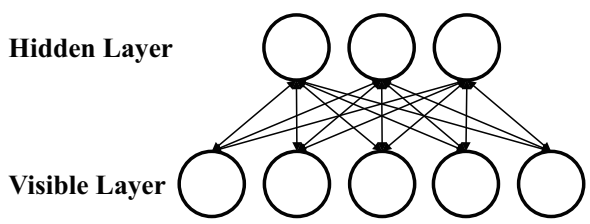

Fig. 3. A simple RBM model with three hidden layer units and five visible layer units.

for a given state $(\mathbf{v}, \mathbf{h})$, the energy of the RBM can be calculated as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} v_i h_j - \sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j \qquad (1)$$

where w_{ij} is the weight, a_i is the visible layer unit bias, b_j is the hidden layer unit bias, and $\boldsymbol{\theta} = \{w_{ij}, a_i, b_j\}$.

After that, the joint probability distribution can be calculated as

$$P(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta}) = \frac{\mathrm{e}^{-E(\mathbf{v},\mathbf{h}|\boldsymbol{\theta})}}{Z(\boldsymbol{\theta})}, \quad Z(\boldsymbol{\theta}) = \sum_{\mathbf{v},\mathbf{h}} \mathrm{e}^{-E(\mathbf{v},\mathbf{h}|\boldsymbol{\theta})} \qquad (2)$$

and the likehood function can be calculated as

$$P(\mathbf{v}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \sum_{\mathbf{h}} \mathrm{e}^{-E(\mathbf{v},\mathbf{h}|\boldsymbol{\theta})} \qquad (3)$$

By formula derivation, we can get the following activation probability formula

$$P\left(h_j = 1|\mathbf{v}, \boldsymbol{\theta}\right) = \sigma\left(b_j + \sum_i v_i w_{ij}\right) \qquad (4)$$

$$P\left(v_i = 1|\mathbf{h}, \boldsymbol{\theta}\right) = \sigma\left(a_i + \sum_j w_{ij} h_j\right) \qquad (5)$$

where $\sigma(x) = 1/(1 + e^{-x})$.

3.2 Contrastive Divergence Algorithm

The purpose of using RBM is to find the suitable value of $\boldsymbol{\theta}$ to fit the given training data. However, obtaining results based on methods such as Gibbs sampling is cumbersome and time consuming [13]. Fortunately, Hinton proposed an algorithm called Contrastive Divergence (CD) [14] which is easy to calculate. Algorithm is shown as Alg. 1. By using the CD algorithm, the suitable $\boldsymbol{\theta}$ value can be calculated very conveniently and quickly.

3.3 Train DBN

Now we construct DBN as shown in Fig. 2. First, we input the data into RBM0 and run CD algorithm. Then we should consider the output of RBM0 as the input of RBM1 and run CD algorithm. After that, we can see the whole network as back propagation neural network and train it. The $\boldsymbol{\theta}$ trained by RBMs should be the initial value of the back propagation neural network.

So what is the advantage of DBN compared to the traditional back propagation neural network? One problem with traditional multi-layer perception or neural network is that the back propagation may always result in local minima. Because when the error surface contains multiple grooves, the deepest groove may cannot be found when using the gradient descent. The deep belief network can solve the problem of local minimum through additional pre-training, i.e. RBM. Pre-training is done before back propagation so that the optimal point is not so far from the initial point. Then we can slowly approach the optimal point through gradient descent.

Algorithm 1 Contrastive Divergence

Training sample x_0, learning rate ϵ, max epoch T;
$\mathbf{v_1} = x_0$, $\mathbf{w}$, $\mathbf{a}$, and $\mathbf{b}$ are random small values;
for $epoch = 1, 2, ..., T$ **do**
  for $j = 1, 2, ..., m$ **do**
    Calc $P\left(h_{1j} = 1|\mathbf{v_1}\right) = \sigma\left(b_j + \sum_i v_{1i} w_{ij}\right)$;
    Randomly select $h_{1j} \in \{0, 1\}$;
  end for
  for $i = 1, 2, ..., n$ **do**
    Calc $P\left(v_{2i} = 1|\mathbf{h_1}\right) = \sigma\left(a_i + \sum_j h_{1j} w_{ij}\right)$;
    Randomly select $v_{2i} \in \{0, 1\}$;
  end for
  for $j = 1, 2, ..., m$ **do**
    Calc $P\left(h_{2j} = 1|\mathbf{v_2}\right) = \sigma\left(b_j + \sum_i v_{2i} w_{ij}\right)$;
  end for
  $\triangle\mathbf{w} = P(h_{1.} = 1|\mathbf{v_1})\mathbf{v_1^T} - P(h_{2.} = 1|\mathbf{v_2})\mathbf{v_2^T}$;
  $\triangle\mathbf{a} = \mathbf{v_1} - \mathbf{v_2}$;
  $\triangle\mathbf{b} = P(h_{1.} = 1|\mathbf{v_1}) - P(h_{2.} = 1|\mathbf{v_2})$;
  $\mathbf{w} \leftarrow \mathbf{w} + \epsilon\triangle\mathbf{w}$;
  $\mathbf{a} \leftarrow \mathbf{a} + \epsilon\triangle\mathbf{a}$;
  $\mathbf{b} \leftarrow \mathbf{b} + \epsilon\triangle\mathbf{b}$;
end for

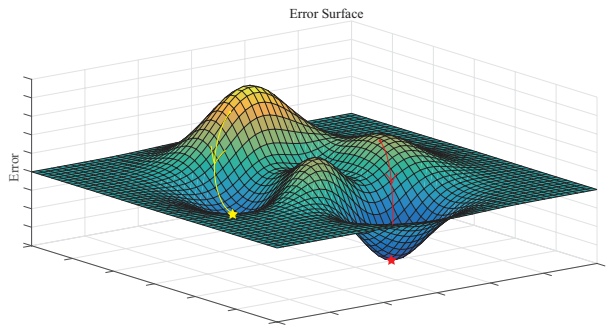

Fig. 4. Error surface with multiple grooves. The yellow pentagram indicates the local minimum point, and the red pentagram indicates the global minimum point. The yellow dot indicates the starting point when RBM is not used. The red dot indicates the starting point when RBM is used. The yellow and red curves indicate the gradient descent routes. Obviously, the red starting point is more helpful for training to get a better classification model.

This is the most basic DBN model and we can optimize it for better classification accuracy and performance. In [8], the authors improved the activation function by combining Sigmoid with LReL to achieve the goal. Authors of [15] proposed the semi-supervised DBN (SSDBN) to improve the structure of the DBN to achieve better performance.

In this project training, I mainly reproduced these two papers. Since there is no suitable optimization method at present, in the DBN and experiment part, only the optimization methods of the published papers are involved. In addition, I found some problems in the process of reading and reproducing the papers, which I will explain and correct below.

3.4 Correction of Problems

The 15^{th} formula in [8] is written incorrectly and should be changed to

$$f_I(x) = \begin{cases} \alpha(x-a) + \text{Sigmoid}(a) & x >= a \\ \text{Sigmoid}(x) & -a < x < a \\ \alpha(x+a) + \text{Sigmoid}(-a) & x <= -a \end{cases}$$

The 17^{th} formula in [15] is written incorrectly and should be changed to

$$E(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{h}/\boldsymbol{\psi}) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i=1}^{m}\sum_{j=1}^{n} v_i w_{ij} h_j - \lambda \sum_{k=1}^{m} c_k u_k - \lambda \sum_{k=1}^{m}\sum_{j=1}^{n} u_k p_{kj} h_j$$

and the 18^{th} should be changed to

$$P\left(h_j/\boldsymbol{v}, \boldsymbol{u}, \psi\right) = \sigma\left(\sum_{i=1}^{m} v_i w_{ij} + \lambda \sum_{k}^{m} u_k p_{kj} + b_j\right)$$

In addition, the corresponding derivatives are not given in the paper, so the results are given here. The specific derivation process is in the report.

$$\begin{aligned} \frac{\partial \log P(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{h})}{\partial w_{ij}} =& P(h_j = 1|\boldsymbol{v^{(0)}}, \boldsymbol{u^{(0)}}) v_i^{(0)} \\ & - P(h_j = 1|\boldsymbol{v^{(1)}}, \boldsymbol{u^{(1)}}) v_i^{(1)} \end{aligned} \tag{6}$$

$$\begin{aligned} \frac{\partial \log P(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{h})}{\partial p_{kj}} =& \lambda P(h_j = 1|\boldsymbol{v^{(0)}}, \boldsymbol{u^{(0)}}) u_k^{(0)} \\ & - \lambda P(h_j = 1|\boldsymbol{v^{(1)}}, \boldsymbol{u^{(1)}}) u_k^{(1)} \end{aligned} \tag{7}$$

$$\frac{\partial \log P(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{h})}{\partial a_i} = v_i^{(0)} - v_i^{(1)} \tag{8}$$

$$\frac{\partial \log P(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{h})}{\partial c_k} = \lambda u_k^{(0)} - \lambda u_k^{(1)} \tag{9}$$

$$\begin{aligned} \frac{\partial \log P(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{h})}{\partial b_j} =& P(h_j = 1|\boldsymbol{v^{(0)}}, \boldsymbol{u^{(0)}}) \\ & - P(h_j = 1|\boldsymbol{v^{(1)}}, \boldsymbol{u^{(1)}}) \end{aligned} \tag{10}$$

4 Experiments

In the experiments of this paper, I used the drive end bearing accelerometer data provided by the Electrical Engineering Laboratory of Case Western Reserve University [16].

Since the content only covers the reproducing part, the main goal is to verify whether the various methods speed up the convergence or improve the classification accuracy rate when the parameters are basically consistent, instead of finding the higher classification accuracy rate.

In the classification of faults, I refer to [6] to classify faults into 10 categories. In this experiment, 1500 data were used, that is, 150 data per category, and each data was extracted from 2048 continuous points to form statistical features.

First I use functions of MATLAB to denoise the data, and the waveform comparison before and after noise reduction is shown in the Fig. 5. As we can see, by noise reduction, a large amount of noise is filtered out so that features can be more easily resolved.

Then I use the sliding window to segment the data and calculate the features according to Table 1. During the experiment, I found that noise

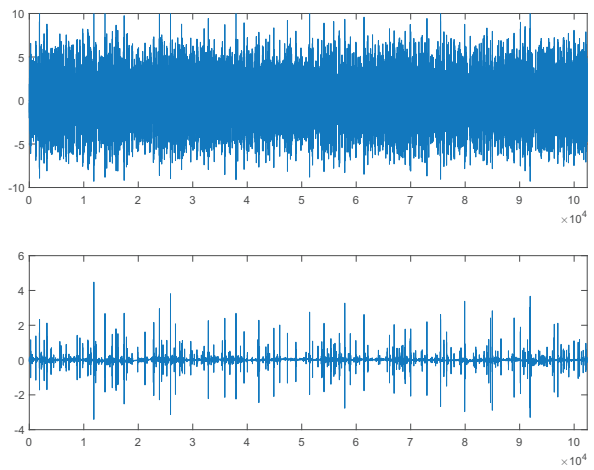

Fig. 5. The above figure is the raw data, and the figure below is the data after noise reduction.

reduction has the potential to remove useful signals, so when calculating features, I separately calculate the features of the raw data and the noise-reduced data, and combine them together. The feature distribution can be seen in Fig. 6. It can be seen that different fault types have distinct feature distributions, so we should extract and use features which can distinguish different fault types easily and classify them. And it requires the help of DBN.

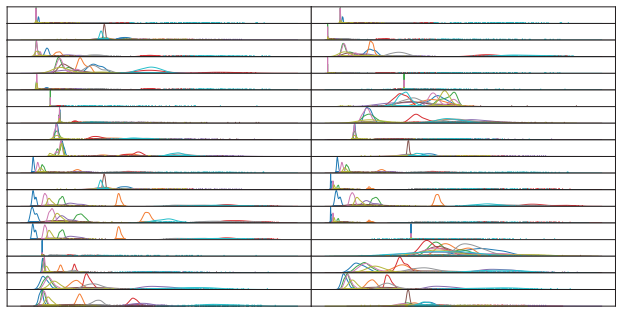

Fig. 6. The figure includes a total of 36 axes. In each axis, curves of different colors represent different fault type data.

After the feature extraction, we can start the training of DBN. In this experiment, three RBMs are used, and the corresponding number of units per layer from bottom to top is 36, 18, 13, and 10. And we should add a classifier which has 10 units at the end of the DBN and compare the result with the label to determine the correctness. MomentumOptimizer is used in the back propagation. For each specific method, I run 50 times and average them for comparison. In the following paragraphs, the black curves always represents the method with theoretically better performance.

4.1 DBN

First I compare the differences between DBN and traditional back propagation neural network. As we can see from Fig. 7, there was no significant improvement in the accuracy rate of the back propagation neural network in the first 200 epoches. The accuracy rate is about the accuracy rate at random selection, i.e. 10%. When using DBN for the training, the

initial accuracy rate can reach 20% and it converges faster. And the final accuracy rate is higher than using the traditional back propagation neural network.

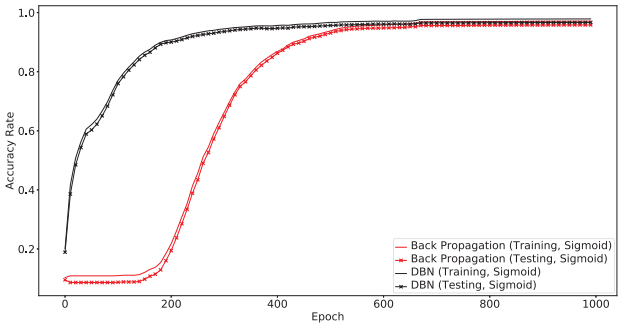

Fig. 7. Comparison of classification accuracy between traditional back propagation neural network and DBN in 1000 epoches.

4.2 Isigmoid

Isigmoid proposed by Yi Qin [8] is capable to relax the vanishing gradient problem and converges faster then Sigmoid. As shown in Fig. 8, since I did not use Isigmoid in RBM (pre-training), their initial accuracy rates are similar. However, when using Isigmoid in the back propagation, the DBN can converge faster and has higher accuracy rate.

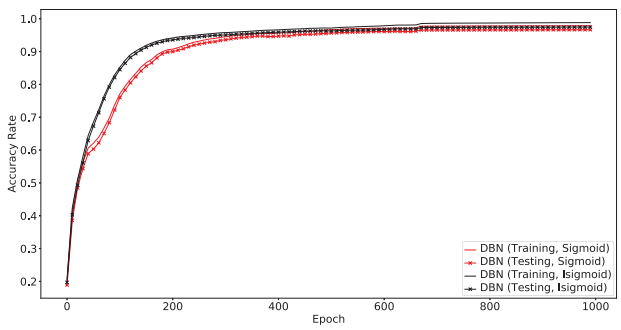

Fig. 8. Comparison of classification accuracy between DBN using Sigmoid and DBN using Isigmoid in 1000 epoches.

4.3 SSDBN

Next I compare DBN with SSDBN. Since SSDBN has studied the sample labels at RBM, I thought that the initial classification accuracy rate should be higher than DBN, but the result is not the case. And DBN seems to converge faseter than SSDBN. However, the final classification accuracy rate of SSDBN is higher than DBN.

In order to find out whether SSDBN really improves the performance of DBN, I compare the reconstruction error of each RBM when using DBN and SSDBN. The reconstruction error can be calculated as

$$R - Error = \sum_{s=1}^{N}\sum_{i=1}^{m}(v_{s,i}^{(0)} - v_{s,i}^{(1)})^2 \tag{11}$$

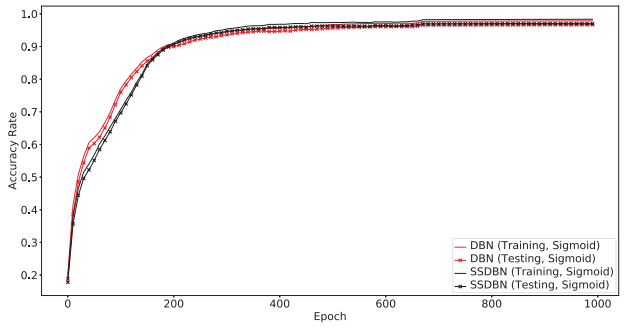

Fig. 9. Comparison of classification accuracy between DBN and SSDBN in 1000 epoches.

where N is the number of input data and m is the number of visible layer units.

The result is shown as Fig. 10. As we can see, in the first two RBMs, the SSDBN does help to reduce the reconstruction error.

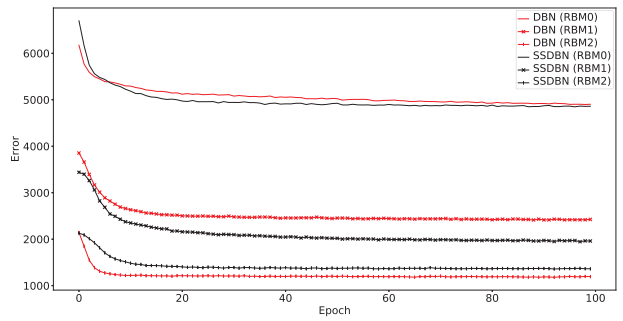

Fig. 10. Comparison of reconstruction error between DBN and SSDBN.

Fig. 11 shows that by combining SSDBN with Isigmoid, we can further improve the classification accuracy.

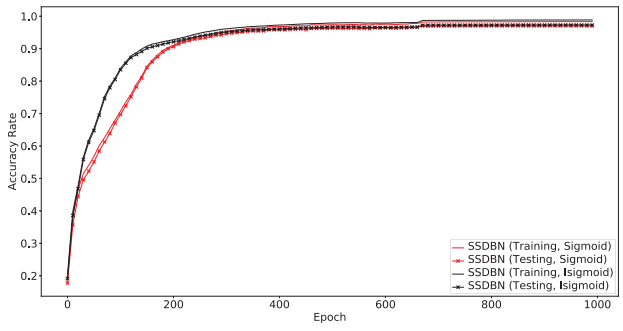

Fig. 11. Comparison of classification accuracy between SSDBN using Sigmoid and SSDBN using Isigmoid in 1000 epoches.

In the traditional back propagation neural network using Sigmoid, the accuracy rate of the training dataset is 97.12%, and the accuracy rate of the testing dataset is 95.83%. In the SSDBN using Isigmoid, the accuracy rate of the training dataset is 98.87%, and the accuracy rate of the testing

dataset is 97.33%. From the above results, we can see that SSDBN and ISigmoid have indeed improved the original method.

5 Conclusions

This paper is a summary of the project training. In this paper, two optimization methods of DBN are mainly involved, which are optimized from two aspects: activation function and structure. The paper also includes some noise reduction and feature extraction methods.

By using RBM, the DBN reduces the possibility of falling into the local minimum point due to the poor initial value when using the traditional back propagation neural network. The network after pre-training using RBM has better initial weight and bias, which makes neural network training more rapid, and has higher classification accuracy rate.

When using Sigmoid as the activation function, the problem of vanishing gradient easily occurs, and LReL cannot fully play its role in DBN. By combining them to form Isigmoid, the possibility of vanishing gradient problem is reduced, and the convergence is faster, and the advantages can be fully utilized in the DBN.

SSDBN optimizes the DBN structure and introduces semi-supervised learning into the DBN, which improves the classification accuracy. By combining SSDBN and Isigmoid, a higher classification accuracy rate can be achieved.

Unfortunately, in this training, I have not come up with a better optimization method to improve the initial classification accuracy rate, speed up the convergence or improve the final classification accuracy rate. In addition, due to the lack of data and poor computer performance, I am unable to successfully train large data sets such as MNIST to further demonstrate the optimization effects of Isigmoid and SSDBN. At the same time, the parameter adjustment for each method may still cannot make the method achieve its optimal performance.

Acknowledgement

Acknowledgement is made for the measurements used in this work provided through data-acoustics.com Database. And I am grateful to senior apprentices and Ms. Huang for guiding me in the project training.

References

[1]Yi Wang, Guanghua Xu, Lin Liang, and Kuosheng Jiang. Detection of weak transient signals based on wavelet packet transform and manifold learning for rolling element bearing fault diagnosis. *Mechanical Systems & Signal Processing*, 54-55:259–276, 2015.

[2]L. U. Yongle, Yingjun Pan, Chunhua Ren, Yu Liu, and Hui Peng. Zero drift compensation of mems accelerometer based on wavelet packers-neural network. *Piezoelectrics & Acoustooptics*, 37(1):27–31, 2015.

[3]Geoffrey E. Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):599–619, 2012.

[4]Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

[5]M. A. Kramer and J. A. Leonard. Diagnosis using backpropagation neural networksanalysis and criticism. *Computers & Chemical Engineering*, 14(12):1323–1338, 1990.

[6]ZHAO Guangquan, GE Qiangqiang, LIU Xiaoyong, and Peng Xiyuan. Fault feature extraction and diagnosis method based on deep belief network. *Chinese Journal of Scientific Instrument*, 39(7):1946–1953, 2016.

[7]Feng Jia, Yaguo Lei, Jing Lin, Xin Zhou, and Na Lu. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing*, 72:303–315, 2016.

[8]Qin Yi, Wang Xin, and Zou Jingqiang. The optimized deep belief networks with improved logistic sigmoid units and their application in fault diagnosis for planetary gearboxes of wind turbines. *IEEE Transactions on Industrial Electronics*, 2019.

[9]B. Walczak and D. L. Massart. Noise suppression and signal compression using the wavelet packet transform. *Chemometrics & Intelligent Laboratory Systems*, 36(2):81–94, 1997.

[10]Bin Zhang, Shaohui Zhang, and Weihua Li. Bearing performance degradation assessment using long short-term memory recurrent network. *Computers in Industry*, 106:14 – 29, 2019.

[11]Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[12]Geoffrey E Hinton, Terrence J Sejnowski, et al. Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(282-317):2, 1986.

[13]ZHANG Chun-xia, JI Nan-nan, and WANG Guan-wei. Restricted boltzmann machines. *CHINESE JOURNAL OF ENGINEERING MATHEMATICS*, 2015.

[14]G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[15]Gongming Wang, Junfei Qiao, Xiaoli Li, Lei Wang, and Xiaolong Qian. Improved classification with semi-supervised deep belief network. *IFAC-PapersOnLine*, 50(1):4174–4179, 2017.

[16]Electrical Engineering Laboratory of Case Western Reserve University. Bearing data. *Case Western Reserve University Bearing Data Center Website*.