Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 55 (2015) 990 – 999

Procedia Computer Science

Information Technology and Quantitative Management (ITQM 2015)

Time Series Prediction using Restricted Boltzmann Machines and Backpropagation

Rafael Hrasko[a], André G. C. Pacheco[a], Renato A. Krohling[a,b]

[a]*Graduate Program in Computer Science, PPGI, UFES - Universidade Federal do Espirito Santo, Av. Fernando Ferrari 514, Vitória 29075-910, Brazil*
[b]*Production Engineering Department, UFES - Universidade Federal do Espirito Santo, Av. Fernando Ferrari 514, Vitória 29075-910, Brazil*

Abstract

Time series prediction appear in many real-world problems, e.g., financial market, signal processing, weather forecasting among others. The underlying models and time series data of those problems are generally complex in a way that reasonable accurate estimation cannot be easily achieved, thus requiring more advanced techniques. Statistical models are the classical approaches for tackling this problem. Many works extended different architectures of Artificial neural networks to work with time series prediction, such as Feedforward, Boltzmann Machines and Deep Belief Network. A Deep Belief Network based on hybridization between Gaussian-Bernoulli Restricted Boltzmann Machine and the Backpropagation algorithm is presented. The hybrid algorithm is tested on three time series databases showing promising results.

Keywords: Time Series Prediction, Gaussian-Bernoulli, Restrict Boltzmann Machine, Deep Belief Network

1. Introduction

Time series, from a general point of view, consist in a sequence of measurements corresponding to a phenomenon observed along the time and therefore chronologically ordered. In the analysis of time series, one wants first to model the phenomenon, describe the behavior of the series, make estimates and evaluate the factors that may have influenced behavior. The problem of estimating a time series is also known as time series prediction.

Time series prediction has been broadly studied in Statistics over the last decades and several models have been developed, e.g. Autoregressive Moving Average Models (ARMA) and Autoregressive Integrated Moving Average Models (ARIMA) [1]. Before that, time series prediction was mainly used in linear problems by means of regression methods. Since the late 1990s machine learning methods have been used in time series prediction. These methods have demonstrated to be powerful non-linear estimators applicable to many real-world problems [2]. Various Machine learning paradigms have approached the time series prediction problem, e.g. Support Vector Machines [3] and Artificial neural networks (ANN) [4].

ANN is a computational paradigm inspired by the neural structures from intelligent organism that acquire knowledge through experience. The most important property of neural networks is the capacity to learn [5]. For

*Corresponding Author. Tel.: +55-27-33374162.
E-mail address: rhrasko@gmail.com (Rafael Hrasko).

doi:10.1016/j.procs.2015.07.104

that learning process to happen, there are many training algorithms like *Gradient descent backpropagation* [6] and *Levenberg-Marquardt backpropagation* [7]. ANN, along with learning algorithms, are usually applied to a variety of computational problems, e.g. regression [8], classification [9] and time series prediction [4].

While the field is maturing, the concept of *Deep Belief Network* (DBN) is becoming more and more important. DBN is a special architecture of ANN, which uses multiple layers of a ANN variation, a concept named *Restricted Boltzmann Machine* (RBM) [10], [11].

In the next section, some background on artificial neural networks, Restricted Boltzmann Machines (RBM) and Deep Belief Network (DBN) is presented. It is also described how the learning algorithm for RBM works. In section 3, the hybrid algorithm is developed. In section 4, the results for 3 benchmarks are presented. The paper ends up with some conclusions in section 5.

2. Background on neural networks

2.1. Artificial neural networks

Artificial neural networks are inspired by biological neural networks. They are interconnected information processing units, called neurons, which have the ability to recognize patterns [5]. All knowledge acquired by an ANN is performed by a learning algorithm, which main function is to modify in an orderly manner the weights of the connections between neurons of the network. The weights of the connections is also known as synaptic weight.

The artificial neuron is the basic processing unit of the neural network, receiving input signal and producing the output to other connected neurons. The most known model of a artificial neuron is the *perceptron*, depicted in figure 1. The output of a neuron is calculated by:

$$y = \theta(\sum_{i=1}^{m} x_i w_i + b) \tag{1}$$

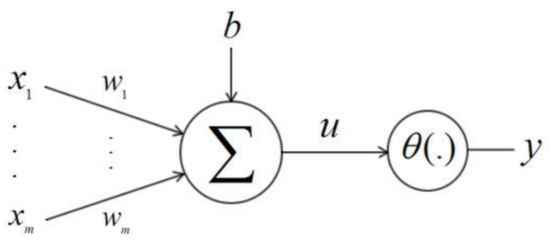

Fig. 1: *Perceptron*

where b is the bias, y is the output, and m is the number of inputs $(x_1, ..., x_m)$ and weights $(w_1, ..., w_m)$.

The function θ is known as the activation function or transfer function. One of the most used activation function is the *sigmoid*, defined by:

$$\theta(u) = \frac{1}{1 + e^{-\lambda u}} \tag{2}$$

where λ is the steepness of the curve.

A Neural Network architecture defines the neurons connectivity structure. Networks such as the one presented in figure 2 are commonly called Feedforward neural networks, because their graph is a directed acyclic graph. It is very common in a feedforward architecture to call the first layer the input layer, the last one the output layer, and all other layers in between hidden layers. Hidden layers are feature analysers of the input layer, while the output process those features. Neurons in the input layer do not process information, since they are used only as signal to hidden layers. The information in such architecture only propagates in one way: from input to hidden to output. The architecture is shown in figure 2.

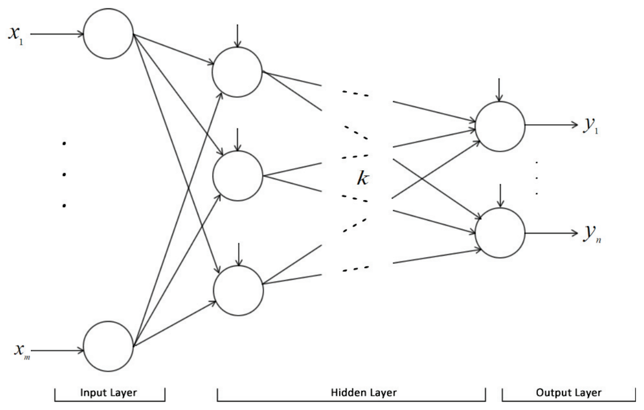

Fig. 2: An illustration of a Feedforward Neural Network

where m is the number of inputs in the input layer, n are the outputs of the output layer and k is the number of hidden layers. k is usually defined empirically and varies with the problem at hand.

Once decided the architecture, a learning algorithm must be chosen. If input and output patterns are used in the training of the network, the learning is called supervised learning. The most known supervised learning algorithm is the *backpropagation* [12]. It updates the weight values using the output and the estimated output using gradient descent [6]. The error E is calculated by [11]:

$$E = \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{n}(d_j^i - y_j^i)^2 \tag{3}$$

where, p is the number of patterns used in the training, n is the number of outputs, d is the desired output and y is the estimated output.

Once calculated the error E, the weight update is given by:

$$W = W - \alpha\frac{\partial E}{\partial W} \tag{4}$$

where α is known as the learning rate.

2.2. *Restricted Boltzmann Machine*

RBM is a stochastic network of neurons made up of two layers: the visible and the hidden layer. The visible layer represent the collected data while the hidden layer try to learn features from the visible layer aiming to represent a probabilistic distribution of the data [13].

The network is called restricted because the neurons in a layer have connections only to the neurons in other layer. Connections between layers are symmetric and bidirectional, allowing information transfer in both directions. RBM's have been used for unsupervised learning, but some works have been adapting RBM for supervised learning [14] [15]. An illustration of a RBM is shown in figure 3.

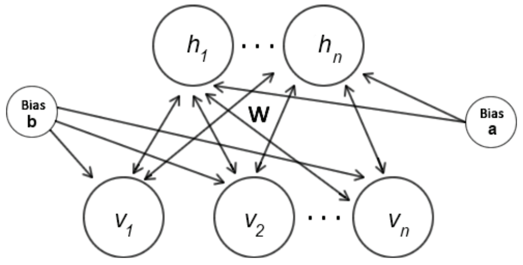

Fig. 3: An illustration of a Restricted Boltzmann Machine

where m is the number of neurons in the visible layer $(v_1, ..., v_m)$, n is the number of neurons in the hidden layer $(h_1, ..., h_n)$, **a** and **b** are *bias* vectors and **W** is the weight matrix.

RBM is an energy-based model that uses a layer of hidden variables to model a probabilistic distribution over visible variables [16]. The visible units constitute the first layer and correspond to the components of an observation, i.e. one visible unit for each feature of an input pattern. The hidden units model dependencies between the components of observations, i.e. the dependencies between features. Being energy-based model means that the probability distribution over the variables **v** e **h** are defined by an entropy function. This function is described by (5) and (6) [11], first in its vectorial form in bold and secondly in extended form.

$$\mathbf{E}(\mathbf{v}, \mathbf{h}) = -\mathbf{h}^T\mathbf{W}\mathbf{v} - \mathbf{a}^T\mathbf{v} - \mathbf{b}^T\mathbf{h} \tag{5}$$

$$E(v, h) = -\sum_{i,j=1}^{m,n} v_i h_j w_{ij} - \sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j \tag{6}$$

Using the entropy function, it is possible to assign a probability for each pair of neurons of the network, one in the visible layer and the other in the hidden layer, giving a probabilistic distribution described by [16]:

$$p(v, h) = \frac{e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}} \tag{7}$$

The probability of the vector from the visible layer, v, is given as the sum of all probabilities of the vector from the hidden layer, described by [16]:

$$p(v) = \frac{\sum_h e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}} \tag{8}$$

Since Restricted Boltzmann Machines do not have connections between neighbouring neurons of the same layer, the events are independents. This facilitates the calculations of the conditional probabilities, which are [16]:

$$p(h|v) = \prod_j p(h_j|v) \tag{9}$$

$$p(v|h) = \prod_i p(v_i|h) \tag{10}$$

The first versions of RBM were developed to solve problems with binary data. Considering that, equations (9) and (10) could be generalized using the correct probability distribution for binary data, given by [16]:

$$p(h_j = 1|v) = sigm(b_j + \sum_{i=1}^{m} v_i w_{ij}) \tag{11}$$

$$p(v_i = 1|h) = sigm(a_i + \sum_{j=1}^{n} h_j w_{ij}) \tag{12}$$

where $sigm(x)$ is the sigmoid function.

Using RBM for only binary data obviously limit its potential for problem solving. There are some works which present new variations for dealing with continuous data. A known approach is to change the probability distribution of the visible layer to a Gaussian distribution. This variation is called Gaussian-Bernoulli RBM (GBRBM) [8]. The equations for the probability distribution are described by [8]:

$$p(h_j = 1|v) = sigm(b_j + \frac{1}{\sigma^2}\sum_{i=1}^{m} v_i w_{ij}) \tag{13}$$

$$p(v_i = 1|h) = N(a_i + \sum_{j=1}^{n} h_j w_{ij}, \sigma^2) \tag{14}$$

where $N(x)$ stands for the Gaussian distribution and σ^2 is the standard deviation. In this case, the deviation is also known as noise level, and it is usually set to 1.

2.2.1. Learning in Restricted Boltzmann Machine

The training of a RBM consists in minimizing the negative log-likelihood given by:

$$\Delta w_{ij} = \epsilon \frac{\partial log p(V)}{\partial w_{ij}} = \epsilon \left\langle v_i h_j \right\rangle_d - \left\langle v_i h_j \right\rangle_m \tag{15}$$

where ϵ is the learning rate, $\langle . \rangle_d$ and $\langle . \rangle_m$ are used to represent the expected values of the data and the model, respectively.

The expectation can be obtained by equations (11) and (12) for binary data, while equations (13) and (14) gives expectations for continuous data. Expectations from the model is harder to be obtained, however, there is a good stochastic approximation known as Contrastive Divergence (CD) algorithm [17]. This algorithm replaces the model expectation for an estimation using Gibbs Sampling with a limited number of iterations. Even when only one iteration of the Gibbs Sampling is used, the CD algorithm is capable to provide good results [18]. Updates for the weight and bias for GBRBM are given by:

$$\Delta w_{ij} = \epsilon \left\langle \frac{v_i h_j}{\sigma^2} \right\rangle_d - \left\langle \frac{v_i h_j}{\sigma^2} \right\rangle_m \tag{16}$$

$$\Delta a_i = \epsilon \left\langle \frac{v_i}{\sigma^2} \right\rangle_d - \left\langle \frac{v_i}{\sigma^2} \right\rangle_m \tag{17}$$

$$\Delta b_j = \epsilon \left\langle \frac{h_j}{\sigma^2} \right\rangle_d - \left\langle \frac{h_j}{\sigma^2} \right\rangle_m \tag{18}$$

2.2.2. Deep Belief Network

RBM's are made up of two layers. Stacking multiple RBM's is called a Deep Belief Network (DBN) as shown in figure 4. In a DBN the hidden layer of one RBM is connected to the visible layer of the next RBM, except for the last one. This provides the ability to train the layers of the network in stages, one RBM at a time using the CD algorithm.

3. A hybrid approach to Time Series Prediction

RBM's are originally used for unsupervised learning. The usual approach for supervised learning is to connect an additional ANN that specializes in supervised problems. The idea is that the hidden units extract relevant features from the observations. These features can serve as input to another RBM. By stacking RBMs in this way, one can learn features in the expectation of arriving at a high-level representation. Hence, the features extracted by single or stacked RBMs can serve as input to a supervised learning algorithm. A common ANN used as a supervised learning system is the feedforward neural network (FNN). In this case, the hidden layer of the last

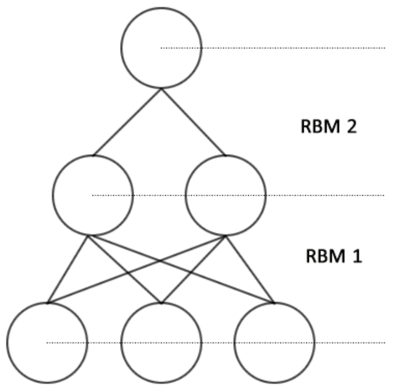

Fig. 4: An example of a Deep Belief Network with two layers

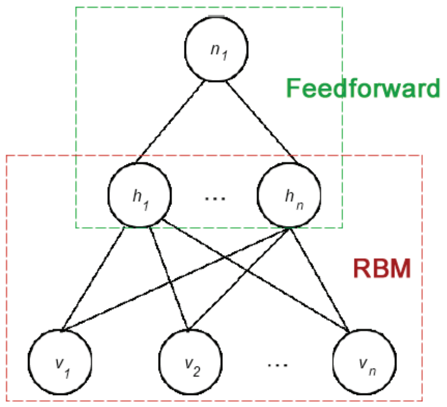

Fig. 5: Deep Belief Network with a Feedforward Network

connected RBM is the input of the FNN as shown in figure 5. The combination of RBM with FNN has been proposed with good results for classification [9] and time series prediction [4].

The use of DBN's for time series prediction is relatively recent. Training a DBN requires a certain amount of practical experience to decide how to set the values of numerical parameters such as the learning rate, the momentum, the weight-cost, the initial values of the weights, the number of hidden units and the number of layers [19]. To determine those parameters, including a delay and interval between delays of the time series, Kuremoto et al. (2014) [4] proposed a DBN to predict time series. In their method a single output layer is fine tuned with Backpropagation (BP) and the parameters are found using the Particle Swarm Optimization (PSO) algorithm.

The time series needed to be reconstructed as input data for the DBN are given as follows [4]:

- Assume the time series data as $x(t), t = 1, 2..., T$, where T is the number of samples of the time series.
- Reconstruct $x(t)$ as $x(t - \tau), x(t - 2\tau), ..., x(t - \mu\tau)$, where τ is a positive integer representing the interval between delays and μ is the delay, also called dimension of the input space.
- Set $v_i = x(i)$, where $i = 1, 2, ..., \mu$ and v_i is the i-th input of the first RBM.

The pseudo-code for the hybrid Gaussian-Bernoulli Restrict Boltzmann Machine with Backpropagation is presented in algorithm 1. The interested reader in the Backpropagation algorithm is referred to [6].

Algorithm 1: Gaussian-Bernoulli Restrict Boltzmann Machine with FNN

```
begin
    Initialization
    Initialize all weights with random numbers
    Set $\langle v_i \rangle_d$ = reconstructed time series
    Set $y$ = time series output
    repeat
        for every layer $l$ do
            for every input $k$ in training dataset do
                for all hidden units $j$ do
                    update $\langle h_j \rangle_d$ according to equation (13)
                end
                for all visible units $i$ do
                    update $\langle v_i \rangle_m$ according to equation (14)
                end
                for all hidden units $j$ do
                    update $\langle h_j \rangle_m$ according to equation (13)
                end
                update weights $W$ according to equation (16)
                update bias $a$ according to equation (17)
                update bias $b$ according to equation (18)
            end
        end
    until until stop criteria is met;
    for every input $k$ in training data do
        for all layers $i$ do
            for all hidden units $j$ do
                update $h_{j_i}$ according to equation (13)
            end
            Set $v_{i+1} = h_i$
        end
    end
    Set output to Backpropagation($h_{lastlayer}, y$)
    return output
end
```

In this paper, we use the source code implemented and described by Tanaka et. al (2014) [20]. However, we modified their code to use the inference described by equations (13) and (14). The modified source code is available from the authors upon request.

4. Simulation Results

In order to evaluate the hybrid algorithm GBRBM+FNN, we apply it to three known time series benchmarks.

4.1. Databases

The databases used to make predictions were obtained in [9].

Australia Energy Production [Energy]: The first series shows the electric energy monthly production from Australia, in GWh, from January 1956 up to August 1995, with 476 samples.

Dollar to Libra Conversion [Dollar]: This series shows monetary information from US$ dollars conversion to Libra. The values represent monthly average from January 1981 up to July 2005, with 295 values.

North Atlantic Oscilation [NAO]: The NAO is the normalized pressure difference between a station on the Azores and one on Iceland. The time series shows monthly data from January 1950 to July 2005, with 667 samples.

Values for μ and τ were empirically searched in the range [2 20]. More advanced approaches are required to find optimal values.

4.2. Parameters Setting

In this paper, we used an exhaustive search to find the parameters. The GBRBM+FNN algorithm and BP algorithm were tested with 100 neurons in the first hidden layer and 50 neurons in the second hidden layer. The parameters μ and τ were set according to each database. The values are shown in table 1

Table 1: μ and τ for each time series

Series	μ	τ
Energy	8	3
Dollar	5	2
NAO	11	4

4.3. Metrics Used

For comparison of the results of both algorithms, it was used the root of the squared error (RMSE), which is calculated by $RMSE = \sqrt{\frac{\sum_i^N (y_i - \widehat{y_i})}{N}}$, where y_i is the desired output corresponding to the input x_i, $\widehat{y_i}$ the value obtained by the algorithm, and N the number of patterns.

4.4. Results and discussion

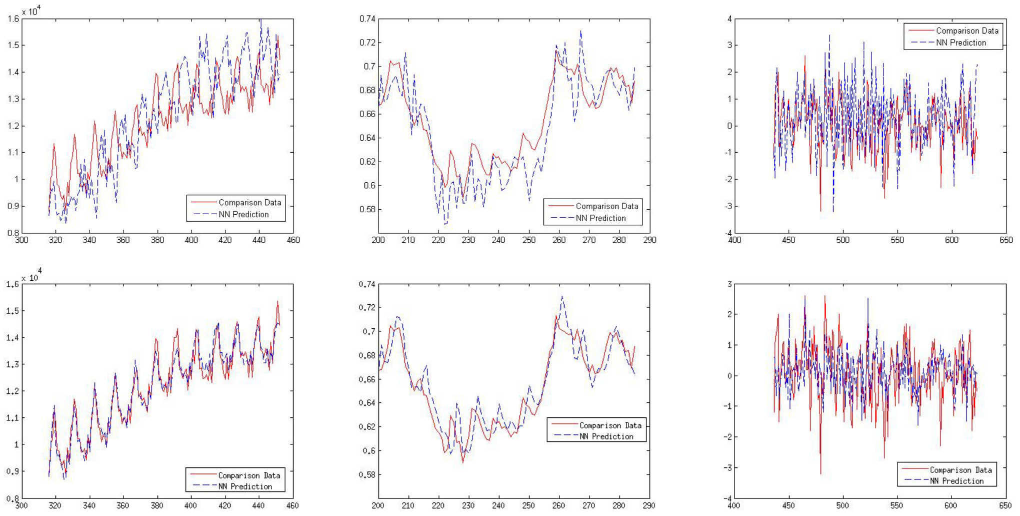

Fig. 6: (a)FNN: Energy Database; (b)FNN: Dollar Database; (c)FNN: NAO Database; (d)GBRBM+FNN: Energy Database; (e)GBRBM+FNN: Dollar Database; (f)GBRBM+FNN: NAO Database

The algorithms were run ten times for each database. We use 70% of the original database for training and 30% for testing. We present the results in terms of mean and standard deviations (stdDev) in table 2. Figure 6 shows the prediction results for each database for both algorithms.

Table 2: Numeric Results

Series	FNN mean(stdDev)	GBRBM+FNN mean(stdDev)
Energy	2.124 (0.971)	0,927 (2.120)
Dollar	0.052 (0.021)	0.093 (0.035)
NAO	31.547 (2.334)	2.663 (0.733)

According to the results we notice that the performance of the GBRBM+FNN algorithm in terms of mean values is better in 2 out of 3 benchmarks compared to the standard FNN. Thus, we observe that the GBRBM+FNN obtained a performance improvement as compared to FNN. The predicted values for the Energy and Dollar series are very close to the true values of the series. It is also noticeable the fact that the best results were obtained for the Dollar series database. The number of inputs, directly related to the delay, seems to have a strong influence on the test results.

5. Conclusion

A hybrid algorithm combining a Gaussian-Bernoulli restricted Boltzmann machine (GBRBM) with a feed-forward neural network (FNN) was presented. Three time series benchmarks were used, i.e. Australia Energy Production, Dollar to Libra Conversion and North Atlantic Oscillation. Preliminary results obtained with the hybrid algorithm GBRBM+FNN seems to be promising. Comparing the hybrid GBRBM+FNN with the BP algorithm turns out that GBRBM+FNN overcomes the BP in 2 out of 3 benchmarks. An extended study with more benchmarks is needed to be carried out in order to draw more general conclusions. In addition, a sensitivity analysis of the key parameters will also be performed in our future study.

References

[1] Box G, Jenkins G M, Reinsel G C. Time series analysis: Forecasting and control. Prentice Hall, New Jersey.

[2] Van Veelen M, Nijhuis J, Spaanenburg B. Neural network approaches to capture temporal information. in: Computing Anticipatory Systems: CASYS'99-Third International Conference Vol. 517 AIP Publishing, 2000 pp. 361–371.

[3] Sapankevych N I, Sankar R. Time series prediction using support vector machines: a survey. Computational Intelligence Magazine 4 (2) (2009) 24–38.

[4] Kuremoto T, Kimura S, Kobayashi K, Obayashi M. Time series forecasting using a deep belief network with restricted Boltzmann machines. Neurocomputing 137 (2014) 47–56.

[5] Haykin S. Neural networks: A comprehensive foundation. Prentice Hall, New Jersey.

[6] Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors. Cognitive modeling 323 (9) (1986) 533–536.

[7] Hagan M T, Menhaj M B. Training feedforward networks with the Marquardt algorithm. Transactions on Neural Networks 5 (6) (1994) 989–993.

[8] Hinton G E, Salakhutdinov R. Reducing the dimensionality of data with neural networks. Science 313 (5786) (2006) 504–507.

[9] Salama M A, Fahmy A A, et al.. Deep belief network for clustering and classification of a continuous data. in: IEEE International Symposium on Signal Processing and Information Technology 2010 pp. 473–477.

[10] Ackley D H, Hinton G E, Sejnowski T J. A learning algorithm for Boltzmann machines. Cognitive science 9 (1) (1985) 147–169.

[11] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets. Neural computation 18 (7) (2006) 1527–1554.

[12] Eberhart R C, Shi Y. Computational intelligence: concepts to implementations Elsevier, 2011.

[13] Memisevic R, Hinton G E. Learning to represent spatial transformations with factored higher-order Boltzmann machines. Neural Computation 22 (6) (2010) 1473–1492.

[14] Xie G, Zhang X, Zhang Y, Liu C. Integrating supervised subspace criteria with restricted Boltzmann machine for feature extraction. in: IEEE International Joint Conference on Neural Networks (IJCNN) 2014 pp. 1622–1629.

[15] Zhou S, Chen Q, Wang X. Active deep learning method for semi-supervised sentiment classification. Neurocomputing 120 (2013) 536–546.

[16] Larochelle H, Bengio Y. Classification using discriminative restricted Boltzmann machines. in: Proceedings of the 25th international conference on Machine learning ACM, 2008 pp. 536–543.

[17] Hinton G E. Training products of experts by minimizing contrastive divergence. Neural computation 14 (8) (2002) 1771–1800.

[18] Carreira-Perpinan M A, Hinton G E. On contrastive divergence learning. in: Proceedings of the tenth international workshop on artificial intelligence and statistics Citeseer, 2005 pp. 33–40.

[19] Hinton G E. A practical guide to training restricted Boltzmann machines. Momentum 9 (1) (2010) 926.
[20] Tanaka M, Okutomi M. A novel inference of a restricted Boltzmann machine. in: Proceedings of the 22nd IEEE International Conference on Pattern Recognition 2014 pp. 1526–1531.