Research on Point-wise Gated Deep Networks
Nan Zhang, Shifei Ding, Jian Zhang, Yu Xue

Abstract

Stacking Restricted Boltzmann Machines (RBM) to create deep networks, such as Deep Belief Networks (DBN) and Deep Boltzmann Machines (DBM), has become one of the most important research fields in deep learning. DBM and DBN provide state-of-the-art results in many fields such as image recognition, but they don't show better learning abilities than RBM when dealing with data containing irrelevant patterns. Point-wise Gated Restricted Boltzmann Machines (pgRBM) can effectively find the task-relevant patterns from data containing irrelevant patterns and thus achieve satisfied classification results. For the limitations of the DBN and the DBM in the processing of data containing irrelevant patterns, we introduce the pgRBM into the DBN and the DBM and present Point-wise Gated Deep Belief Networks (pgDBN) and Point-wise Gated Deep Boltzmann Machines (pgDBM). The pgDBN and the pgDBM both utilize the pgRBM instead of the RBM to pre-train the weights connecting the networks’ the visible layer and the hidden layer, and apply the pgRBM learning task-relevant data subset for traditional networks. Then, this paper discusses the validity that dropout and weight uncertainty methods are developed to prevent overfitting in pgRBM, pgDBNs, and pgDBMs networks. Experimental results on MNIST variation datasets show that the pgDBN and the pgDBM are effective deep neural networks learning.

1. Introduction

Restricted Boltzmann Machines (RBM) [1] are effective unsupervised learning methods, which not only can learn representations retaining the information about unlabeled data but also can reconstruct unlabeled data form learned representations. However, if the unlabeled data contains irrelevant patterns then the learned representations will retain the information about irrelevant patterns. Indeed, feature selection methods can distinguish useful representations from irrelevant representations [2]. Sohn et al. proposed Point-wise Gated Restricted Boltzmann machines (pgRBM) which combine RBMs and feature selection methods [3]. The pgRBM not only learn useful representations from irrelevant representations but also can get the “cleaned” data through a switch layer. But the pgRBM cannot learn higher-level representations because it is a shallow learning model.

Restricted Boltzmann Machines (RBM) can be stacked to create deep networks, such as Deep Belief Networks (DBN) [1] and Deep Boltzmann Machines (DBM) [4]. However, the DBN and the DBM don't show better learning abilities than the RBM when dealing with data containing irrelevant patterns. Assume that irrelevant patterns in data lead to bad intermediate level representations and affect the performance of the DBN and the DBM, this paper introduces the pgRBM into the DBN and the DBM and presents Point-wise Gated Deep Belief Networks (pgDBN) and Point-wise Gated Deep Boltzmann Machines (pgDBM). The pgDBN and the pgDBM both utilize the pgRBM instead of the RBM to learn task-relevant patterns in data.

The overfitting problem is a familiar problem in neural networks, so it is in deep networks based on RBM. At present, weight decay, dropout [5], and weight uncertainty [6,7] methods are developed to prevent overfitting in deep networks. Dropout and weight uncertainty can effectively solve the overfitting problem in the RBM, but their performance is truly dissatisfactory when they deal with data containing irrelevant patterns. Introducing dropout and weight uncertainty methods into pgRBM, this paper puts forward Point-wise Gated Dropout Restricted Boltzmann Machines (pgRBM) and Point-wise Gated Weight Uncertainty Restricted Boltzmann Machines (pgwRBM), and then...
discusses the effectiveness of these two algorithms. Next, this paper explores the validities and practicalities of these two methods in pgDBN and pgDBM networks.

The remainder of the paper is organized as follows. Section 2 introduces the related work, such as RBMs, DBNs, DBMs, and pgRBMs. Section 3 describes Point-wise Gated Deep Belief Networks and Point-wise Gated Deep Boltzmann Machines. Section 4 details the application of dropout and weight uncertainty in pgRBMs, pgDBNs, and pgDBMs. In Section 5, experiment results prove the effectiveness of pgDBNs and pgDBMs, whilst we discuss the validities and practicalities of dropout and weight uncertainty in pgRBMs, pgDBNs, and pgDBMs networks. Finally, some conclusions and the intending work are given in the last section.

2. Related work

The proposed models in this paper are primarily based on Restricted Boltzmann Machines (RBM) and the RBM variants. In this section, we first give a brief history of RBMs, and then describe related models.

2.1. A brief history of restricted boltzmann machines

General Boltzmann machines provide a powerful tool for representing dependency structure between random variables. A Boltzmann Machine (BM) has a layer of visible units and a layer of hidden units with any pattern of connectivity between the units. The gradient of the log likelihood in the BM with respect to a connection weight can be expressed simply as the difference of the data-dependent statistic and the model-dependent statistic [8,9]. In 1986, Smolensky first proposed Restricted Boltzmann Machines (RBM) on the foundation of BMs [10]. The RBM prohibits connections between visible units and connections between hidden units, which makes it easy to compute data-dependent and model-dependent statistics exactly. Sequential Gibbs sampling, simulated annealing, and persistent Markov chains are traditional methods of computing data-dependent and model-dependent statistics. However, they perform poorly on large datasets. Contrastive Divergence (CD), proposed by Hinton in 2002 [11], gives very biased estimates of the model-dependent statistics on large datasets. In 2006, Hinton and Salakhutdinov provided an effective way to create deep networks called Deep Belief Networks [1], which opened the research upsurge of deep learning.

Since then, many scholars have been working on the theoretical research of RBM and its applications in each field. Taking image processing as an example, RBM was originally only suitable for addressing binary images. In order to deal with real images, a series of RBM variants are put forward, such as Gaussian RBMs (GRBM), covariance RBMs (cRBM), mean and covariance RBMs (mcRBM), and spike and slab RBMs (ssRBM). The GRBM is not able to effectively learn the edge features of images, and the cRBM can well describe the covariance structure of images [12]. mcRBM and ssRBM are both improved on the basis of the first two models [13,14]. Both of them show better abilities in real images modeling. The main difference between the two algorithms is that the mcRBM adopts the hybrid Monte Carlo (HMC) method to compute the states of visible units, while the states of hidden units are given and the ssRBM uses block Gibbs sampling. These RBM variants convert images into one-dimensional vectors when they learn features. Considering two-dimensional structure information of images, Lee et al. combined convolutional neural network with RBM and put forward convolutional RBM [15]. Robust RBMs (RoBM) and Point-wise gated RBMs (pgRBM) adjust the network model and the energy function of the RBM to better realize image recognition and classification tasks. Adding noise on a certain part of images, the RoBM can learn the structure of noise to achieve better results in the face recognition task [16]. In order to address data containing irrelevant patterns, the pgRBM divides hidden units into task-relevant units and task-irrelevant units and thus finds the task-relevant patterns from data [3]. Of course, there are many other RBM variants, such as Discrete RBMs [17] and Conditional RBMs [18]. In the view of gradient approximation, Persistent Contrastive Divergence (PCD) and Fast Persistent Contrastive Divergence (FPCD) were proposed on the basis of CD. The PCD makes use of alternating Gibbs sampling to compute the model-dependent statistic [19]. In light of the PCD, the FPCD divides the weights connecting visible units and hidden units into regular weights and fast weights, where regular weights are used to compute the data-dependent statistic and the sum of regular weights plus fast weights can be utilized to compute the model-dependent statistic [20]. In addition, the update of fast weights can allow a much higher learning rate. Recently, Carlson et al. utilized the Stochastic Spectral Descent method (SSD) instead of the stochastic gradient descent method (SGD) to update the weights in RBM and obtained good results in the assessment of the log likelihood function on handwritten datasets [21].

2.2. Restricted boltzmann machines and deep networks based on RBMs

An Restricted Boltzmann Machine (RBM) is a generative stochastic network and it contains a layer of visible units \( v = [v_i]_{i=1}^{D} \) and a layer of hidden units \( h = [h_i]_{i=1}^{J} \) with the parameters \( \theta \). The parameters \( \theta \) consist of the weights connecting visible units and hidden units \( \mathbf{W} \in \mathbb{R}^{D,J} \), bias terms of the visible layer \( \mathbf{c} = [c_i]_{i=1}^{D} \), and bias terms of the hidden layer \( \mathbf{b} = [b_i]_{i=1}^{J} \). The energy function and the likelihood function of the RBM are expressed as:

\[
\begin{align*}
E(v, h; \theta) &= -\sum_{i=1}^{D} v_i W_{ij} h_j - \sum_{j=1}^{J} b_j h_j - \sum_{i=1}^{D} c_i v_i, \\
P(v; \theta) &= \sum_{h} P(v, h; \theta) = \sum_{h} \left( \frac{1}{Z(\theta)} \exp(-E(v, h; \theta)) \right) = \frac{1}{Z(\theta)} \sum_{h} \exp(-E(v, h; \theta)),
\end{align*}
\]

(1)

(2)

where \( v_i \in \{0, 1\} \), \( h_j \in \{0, 1\} \), \( Z(\theta) = \sum_{h} \sum_{v} \exp(-E(v, h; \theta)) \) is the partition function, and \( P(v; \theta) \) also can be called the marginal distributions of \( P(v, h; \theta) \). The conditional probabilities of the RBM can be given by:

\[
P(v_i = 1 | h) = \sigma(\sum_{j} W_{ij} h_j + c_i),
\]

(3)
where $\sigma(x) = \frac{1}{1+\exp(-x)}$. By maximizing the log likelihood function $\sum_{k=1}^{N} \ln P(\mathbf{v}^k)$, the RBM utilizes the stochastic gradient ascent algorithm to update the weights $\theta$ when a train sample set $\mathbf{X} = \{\mathbf{v}_i^k\}_{i=1}^{N}$ is given. We can use the Contrastive Divergence (CD) or other algorithms to estimate these gradients [11].

A Deep Belief Network (DBN) is a probabilistic generative deep model and it can be created by stacking RBMs. The DBN first makes use of the data without labels and the RBM pre-training layer by layer to initialize the weights of the network. After the pre-training, the DBN utilizes the data with labels and the gradient descent method to fine-tune the weights.

Similarly to the DBN model, the DBM also uses the RBM pre-training layer by layer to initialize the weights of the network and the details see the literatures [4,22]. The difference is that the DBM is an undirected graph model and the outputs of a hidden layer are determined by the upper and lower two layers (the ratio of the two layers is consistent in general). Compared with the DBN, the DBM shows not only outstanding classification abilities, but also powerful image reconstruction abilities. By taking a DBM having two hidden layer as an example, the energy function can be expressed as:

$$E(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}; \theta) = -\sum_{i=1}^{D} \sum_{j=1}^{I} v_i W_{ij}^{(1)} h_j + b_i^{(1)} - \sum_{j=1}^{J} \sum_{j'=1}^{J'} h_j^{(1)} W_{jj'}^{(2)} h_{j'}^{(2)} - \sum_{j=1}^{J} b_{j}^{(1)} h_j^{(1)} - \sum_{j'=1}^{J'} b_{j'}^{(2)} h_{j'}^{(2)} - \sum_{i=1}^{D} c_i v_i.$$  

At the same time, the conditional probabilities of the RBM can be given by:

$$P(h_j^{(1)} = 1 | \mathbf{v}, \mathbf{h}^{(2)}; \theta) = \sigma(\sum_i v_i W_{ij}^{(1)} + \sum_{j'} h_j^{(2)} W_{jj'}^{(2)} + b_j^{(1)}),$$  

$$P(h_j^{(2)} = 1 | \mathbf{h}^{(1)}; \theta) = \sigma(\sum_{j'} h_j^{(1)} W_{jj'}^{(2)} + b_j^{(2)}),$$  

$$P(v_i = 1 | \mathbf{h}^{(1)}; \theta) = \sigma(\sum_{j} h_j^{(1)} W_{ij}^{(1)} + c_i).$$  

The gradients in the DBM are estimated by the Contrastive Divergence (CD) or other algorithms, too. We should utilize the mean field method to compute the probability distribution of the second hidden layer $Q(\mathbf{h}^{(2)})$ before the gradient descent method is used to fine-tune the weights of the network. Then, the original data $\mathbf{v}$ is combined with $Q(\mathbf{h}^{(2)})$ to form the inputs of the network. Finally, we random initialize the weights connecting the second hidden layer and the output layer and then use the gradient descent method fine-tune the weights of the network.

### 2.3. Point-wise gated restricted boltzmann machines

Compared with the RBM, the pgRBM has a layer of visible units, a layer of switch units, and two layers of hidden units, and the network structure of pgRBM is shown in Fig. 1. The pgRBM divides hidden units into task-relevant units and task-irrelevant units. Then, the energy function of the pgRBM can be expressed as:

$$E(\mathbf{v}, \mathbf{z}, \mathbf{h}; \theta) = -\sum_{i=1}^{D} \sum_{j=1}^{I} \sum_{r=1}^{2} z_{ir}^z v_i W_{ij} h_r^h - \sum_{j=1}^{J} \sum_{r=1}^{2} z_{jr}^z h_j - \sum_{i=1}^{D} \sum_{r=1}^{2} c_i^z (z_{ir}^z v_i),$$  

s.t. $\sum_{r=1}^{2} z_{ir}^z = 1, i = 1, \cdots, D.$
where the weights \( \{W^1, c^1, b^1\} \) correspond to the foreground hidden layer \( h^1 \) and the weights \( \{W^2, c^2, b^2\} \) correspond to the background hidden layer \( h^2 \). The conditional probabilities of the pgRBM can be given by:

\[
P(h^1_i = 1|v, \mathbf{v}) = \sigma(\sum_j (c^1_j v_j) W^1_{ji} + b^1_i),
\]

\[
P(v = 1|h^1, \mathbf{h}^2) = \sigma(\sum_{j} \sum_{r} (\sum_{i} W^2_{ji} h^2_j + c^2_{jr})) \exp(v_i(\sum_{j} W^2_{ji} h^2_j + c^2_{jr})),
\]

\[
P(z^1_i = 1|v, \mathbf{h}^1, \mathbf{h}^2) = \frac{\exp(v_i(\sum_{j} W^1_{ji} h^1_j + c^1_i))}{\sum_s \exp(v_i(\sum_{j} W^1_{ji} h^1_j + c^1_s))}.
\]

In the literature [3], it is proved that if the weights of the pgRBM are initialized by the RBM and the feature selection methods instead of randomly then the pgRBM will perform better. As shown in Fig. 1, the number of the foreground and background hidden layers is \( J_1 \) and \( J_2 \) (\( J_1 \) is equal to \( J_2 \) in the literature [3]). At the same time, the number of the hidden layer units in RBM is slightly larger than the sum \( J_1 \) plus \( J_2 \) when the weights of the pgRBM are initialized by the RBM and the feature selection methods. Like the RBM, the gradients in the pgRBM are estimated by the Contrastive Divergence (CD) or other algorithms.

3. Point-wise Gated Deep Networks

3.1. Point-wise Gated Deep Belief Networks

DBN don’t show better learning abilities than RBM when dealing with data containing irrelevant patterns. The irrelevant patterns in data affect the performance of the DBN, but the pgRBM can divide hidden units into task-relevant units and task-irrelevant units and thus find the task-relevant patterns from data. Therefore, this paper introduces pgRBM into DBN and presents Point-wise Gated Deep Belief Networks (pgDBN) which utilizes pgRBM instead of RBM to learn task-relevant patterns in data. By taking a pgDBN having two hidden layer as an example, Fig. 2 shows the pre-training process of the pgDBN. The pgDBN first utilizes pgRBM to pre-train the weights connecting the visible layer, the switch layer, and the first hidden layer, and then uses RBMs to obtain the weights between the hidden layers. After the pre-training, the weights connecting the last hidden layer and the output layer are randomly initialized and then the pgDBN uses the gradient descent method fine-tune the weights of the network. In the whole learning process of the pgDBN, learning the weights connect the visible layer and the first hidden layer by the pgRBM takes the longest time. If a DBN and a pgDBN have the same network structure, the network of the RBM used in the pgDBN pre-training process is different from the pgRBM used in the pgDBN when the weights connect the visible layer and the first hidden layer is leaned. The number of the pgRBM hidden layer is usually double the RBM. The learning procedure of the pgRBM is more complex than the RBM, and the weights of the pgRBM are initialized by a RBM which has a little more hidden layer units than the pgRBM. Therefore, a pgDBN takes several times longer than a DBN when their networks are the same.

3.2. Point-wise Gated Deep Boltzmann Machines

Similarly to the pgDBN model, this paper utilizes pgRBM instead of RBM to learn the weights and presents Point-wise Gated Deep Boltzmann Machines (pgDBM). By taking a pgDBM having two hidden layer as an example, Fig. 3 shows the pre-training process of the pgDBM. Compared with pgDBNs, pgDBMs need to use higher-level knowledge to resolve uncertainty about intermediate level features in the approximate inference procedure. So when we compute the weights connecting the visible layer, the switch layer, and the first hidden
layer in the greedy pre-training process of the pgDBM, we need to copy the visible layer and the switch layer of the pgRBM. This copying insinuates that the outputs of the first hidden layer are determined jointly by the visible layer and the second hidden layer. The conditional probabilities of the modified pgRBM can be given by:

\[ P(h^1 | z, v, z', v') = \sigma(\sum_z (\tilde{c}_i^1 v_i + z_i^2 v_i') W^1_{ji} + b_j^1 + b_j'^1), \]

\[ P(v' = 1 | z, z', h^1, h^2) = P(v_i = 1 | z, z', h^1, h^2) = \sigma(\sum_t z^1_t (\sum_j W^2_{jt} h^2_t + c_j^2)), \]

\[ P(v' = 1 | v, v', h^1, h^2) = P(z^1_t = 1 | v, v', h^1, h^2) = \frac{\exp(v_i (\sum_j W^2_{jt} h^2_t + c_j^2))}{\sum_s \exp(v_i (\sum_j W^2_{jt} h^2_t + c_j^2))}, \]

As shown in Fig. 3, the learned weights \( \{W^1, c^1, b^1, z^1\} \) of the modified pgRBM can be used in pgDBM. We find that if the weights of the modified pgRBM are initialized by the regular pgRBM and the feature selection methods such as the \( t \)-test instead of randomly then the pgDBM will perform better. If the learned weights of the regular pgRBM are \( \{\tilde{W}^1, \tilde{c}^1, \tilde{b}^1, \tilde{W}^2, \tilde{c}^2, \tilde{b}^2\} \) then the weights of the modified pgRBM are initialized using \( \{\tilde{W}^1 / 2, \tilde{c}^1 / 2, \tilde{b}^1 / 2, \tilde{W}^2 / 2, \tilde{c}^2, \tilde{b}^2\} \). Then, the outputs of the first hidden layer can be expressed as \( P(h^1 = 1 | z^1, v, z', v') \). Next, similar to DBM, pgDBM uses the modified RBM to learn the weights between the hidden layers. The conditional probabilities of the modified RBM can be given by:

\[ P(h_j^{(1)} = 1 | h_j^{(2)}, h_i^{(2)}) = \sigma(\sum_{f} W^{2}_{ij} h_{i}^{(2)} + c_j^{(2)} + \sum_{f} W^{2}_{if} h_{f}^{(2)} + c_j^{(2)}), \]

\[ P(h_j^{(2)} = 1 | h_i^{(1)}) = P(h_j^{(2)} = 1 | h_i^{(1)}) = \sigma(\sum_{f} h_{i}^{(1)} W^{2}_{if} + b_j^{(2)}). \]

If the learned weights of the RBM are \( \{W^{(2)}, c^{(2)}, b^{(2)}\} \) then the weights of the pgDBM are initialized using \( \{z^1, c, W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\} \). And the states of the switch layer \( z^1 \) remain unchanged when the weights of the pgRBM are fine-tuned. So we use the data \( v \) and \( z^1 \) compute a new data \( \tilde{v} = v \times z^1 \) (where \( \times \) denotes element-wise multiplication) and the energy function of pgDBM can be expressed as:

\[ E(\tilde{v}, h^1, h^2; \theta) = -\sum_{i=1}^{D} \sum_{j=1}^{F} \tilde{v}_i W^{(1)}_{ij} h_j^{(1)} - \sum_{j=1}^{F} \sum_{f=1}^{D} h_j^{(1)} W^{(2)}_{jf} h_f^{(2)} - \sum_{j=1}^{F} b_j^{(1)} h_j^{(1)} - \sum_{j=1}^{F} b_j^{(2)} h_f^{(2)} - \sum_{i=1}^{D} c_i \tilde{v}_i. \]

After the greedy pre-training procedure, the pgDBM uses mean field method and the Contrastive Divergence-\( k \) (CD-\( k \)) or other gradient estimate algorithms to fine-tune the weights of the network, and the details are given in Algorithm 1. Next, we utilize the new weights \( \{c, W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\} \), and the mean field method to compute the probability distribution of the second hidden layer \( Q(h^{(2)}) \) which is combined with the data \( \tilde{v} \) to form the inputs of the classified network. In the classified network, the weights of the first two layers are \( \{[c \tilde{b}^{(2)}], [W^{(1)}; (W^{(2)})^\top], b^{(1)}, W^{(2)}, b^{(2)}\} \). Finally, we randomly initialize the weights connecting the second hidden layer and the output layer and then use the gradient descent method fine-tune the weights of the network. In the whole learning process of the pgDBM, learning the weights connect the visible layer and the first hidden layer by the modified pgRBM takes the longest time. The number of the modified
pgRBM hidden layer is usually double the modified RBM used in the DBM, and the weights of the modified pgRBM are initialized by a regular pgRBM which also takes a long time. In the whole learning process of the pgDBM, the fine-tuning the whole weights procedure also takes a long time. So a pgDBM also takes several times longer than a DBM when their networks are the same.

Algorithm 1 Fine-tuning the weights of a two-layer pgDBM for classification

1. Given: a train set of $N$ data vectors $(v^{(t)})^N_{t=1}$, and the number of Gibbs sampling $K$.

2. // Use the learned weights of the modified pgRBM $(W^i, c^i, b^i, W^2, c^2, b^2)$ to learn a new data:

3. for each training sample $v^{(t)}$, $n = 1$ to $N$ do

4. define a vector $z^{(t)} = 1$, run mean field updates until convergence:

   $h^{(i)} = \sigma((v^{(t)} \times z^{(t)})(2W^i) + 2b^i), \quad \times \text{ denotes element-wise multiplication.}$

   $h^{(2)} = \sigma((v^{(t)} \times (1-z^{(t)}))(2W^2) + 2b^2), \quad \sigma(x) = 1/(1+\exp(-x)).$

   $z = \sigma(v^{(t)} \times (h^{(1)}(W^1)^T + c^1) - (h^{(2)}(W^2)^T + c^2)).$

5. Set $z^{t} = z$.

6. end for

7. Compute the new data $\bar{v} = v \times z$.  // $(z^{(t)})^N_{t=1}$ is related to the foreground hidden layer.

8. Obtain the learned weights $(c, W^{(0)}, b^{(0)}, W^{(2)}, b^{(2)})$ in the greedy pre-training procedure.

9. for $t = 1$ to $T$ (the number of iterations) do

10. // Variational Inference:

11. for each new sample $\bar{v}^{(t)}$, $n = 1$ to $N$ do

12. define $h^{(0)} = \sigma((\bar{v}^{(t)})(2W^{(0)}) + b^{(0)}), \quad h^{(2)} = \sigma((h^{(0)})(W^{(2)}) + b^{(2)}),$ and run mean field updates until convergence:

   $h^{(0)} = \sigma((\bar{v}^{(t)}W^{(0)} + h^{(2)}(W^{(2)})^T + b^{(0)}), \quad h^{(2)} = \sigma((h^{(0)})(W^{(2)}) + b^{(2)}).$

13. Set $h^{(0)} = h^{(0)}, \quad h^{(2)} = h^{(2)}$.

14. end for

15. // Stochastic Approximation:

16. Initialize M samples: $(\bar{v}^{(0)}, h^{(0)}(0), h^{(2)}(0)), \ldots, (\bar{v}^{(K)}, h^{(0)}(K), h^{(2)}(K)).$

17. Make use of $(\bar{v}^{(k)}, h^{(0)}(k), h^{(2)}(k))$ to obtain binary vectors $(h^{(0)}(k), h^{(2)}(k))$ and $\bar{v} = \bar{v}$.

18. for $k = 1$ to $K$ do

19. for each new sample $\bar{v}^{(t)}$, $n = 1$ to $N$ do

20. Sample $(\bar{v}^{(t)}, h^{(0)}(k), h^{(2)}(k))$ given $(\bar{v}^{(t-1)}, h^{(0)}(k-1), h^{(2)}(k-1))$ by running a Gibbs sampler (Eqs. 6-8).

21. end for

22. end for

23. // Parameter Update:

24. $W^{(0)} = W^{(0)} + \alpha_a \left( \sum_{m=1}^{N} \bar{v}^{(t)} (p_m^{(0)})^T - \sum_{m=1}^{N} \bar{v}^{(t)} (h^{(0)}(m))^T \right) / N.$

25. $W^{(2)} = W^{(2)} + \alpha_c \left( \sum_{m=1}^{N} p_m^{(2)} (p_m^{(2)})^T - \sum_{m=1}^{N} h^{(2)}(m)^T \right) / N.$

26. $c = c + \alpha_c \left( \sum_{m=1}^{N} \bar{v}^{(t)} - \sum_{m=1}^{N} h^{(2)}(m) \right) / N.$

27. $b^{(0)} = b^{(0)} + \alpha_a \left( \sum_{m=1}^{N} p_m^{(0)} - \sum_{m=1}^{N} h^{(0)}(m) \right) / N.$

28. $b^{(2)} = b^{(2)} + \alpha_c \left( \sum_{m=1}^{N} p_m^{(2)} - \sum_{m=1}^{N} h^{(2)}(m) \right) / N.$

29. Decrease $\alpha_a$.

30. end for

4. Methods Preventing Overfitting in Point-wise Gated Deep Networks

4.1. Dropout in Point-wise Gated Deep Networks

The Dropout is an effective solution to the overfitting problem in neural networks. The key idea of the dropout is to randomly drop units from the neural network during training. Dropping units out can be interpreted as temporarily removing these units from the network
and the dropped units are resampled for each mini-batch. Srivastava et al. introduce the dropout into the RBM and put forward Dropout Restricted Boltzmann Machines (dRBM). Similar to the Denoising Autoencoder (DAE) [23] and the Extreme Learning Machine based DAE [24], the dropout in dRBM can be interpreted as an effective regularization method. But the difference is that the dRBM adds noise to the hidden layer units and the other two methods adds noise to the input layer units. But the dRBM performs poorly when it deals with data containing irrelevant patterns. Therefore, this paper introduces the dropout into the pgdRBM and proposes Point-wise Gated Dropout Restricted Boltzmann Machines (pgdRBM). The Dropout method is only used in the foreground hidden layer in pgdRBM. That is, the connections between the visible layer, the switch layer, and the foreground hidden layer in pgdRBM can be regarded as the integration of many sub networks. The outputs of the foreground hidden layer are related to a set of binary variables $r = (0, 1)^J$. The value of $r_j$ equals 1 with the probability $p$. If the value of $r_j$ equals 1 then the output of the corresponding foreground unit $h^1_j$ remains unchanged otherwise it is 0. The energy function of the pgdRBM can be expressed as:

$$
E(v, z, h^1, h^2; \theta) = -\sum_{i=1}^{J1} \left( z^1_i v_i \right) W_{ij}^1 \left( r_j h^1_j \right) - \sum_{i=1}^{J1} \sum_{j=1}^{J2} \left( z^2_i v_i \right) W_{ij}^2 h^2_j - \sum_{i=1}^{J1} \sum_{j=1}^{J2} c_i^j (z^1_i v_i)
$$

$$
- \sum_{j=1}^{J1} b_1 \left( r_j h^1_j \right) - \sum_{j=1}^{J2} b_2 h^2_j
$$

$$
s.t. \sum_{r=1}^{J2} z^r_i = 1, i = 1, \ldots, D.
$$

The conditional probabilities of the pgdRBM can be given by:

$$
P(h^1_j = 1 | z^1, v, r_j) = \sigma \left( \sum_i z^1_i v_i W_{ij}^1 + b^1_j \right).
$$

$$
P(h^2_j = 1 | z^1, v, r_j) = \sigma \left( \sum_i z^1_i v_i W_{ij}^2 + b^2_j \right).
$$

$$
P(v_i = 1 | z^1, h^1, r, h^2) = \sigma \left( z^1_i \left( \sum_j W_{ij} h^1_j + c^1_i \right) + z^2_i \left( \sum_j W_{ij}^2 h^2_j + c^2_i \right) \right)
$$

$$
P(z^1 = 1 | v, h^1, r, h^2) = \frac{\exp \left( v_i \left( \sum_j W_{ij} h^1_j + c^1_i \right) \right)}{\exp \left( v_i \left( \sum_j W_{ij} h^1_j + c^1_i \right) \right) + \exp \left( v_i \left( \sum_j W_{ij}^2 h^2_j + c^2_i \right) \right)}.
$$

$$
P(z^2 = 1 | v, h^1, r, h^2) = \frac{\exp \left( v_i \left( \sum_j W_{ij}^2 h^2_j + c^2_i \right) \right)}{\exp \left( v_i \left( \sum_j W_{ij} h^1_j + c^1_i \right) \right) + \exp \left( v_i \left( \sum_j W_{ij}^2 h^2_j + c^2_i \right) \right)}.
$$

We find that if the pgdRBM uses the RBM and the feature selection methods to initialize the weights then it will perform better. If the number of the foreground and background hidden layer units is $J1$ and $J2$ ($J1 \times p$ is equal to $J2$ in this paper), then the number of the hidden layer units in RBM is slightly larger than the sum $J1$ plus $J2$. The gradient estimate procedure of the pgdRBM is similar to the pgRBM. If the number of the pgdRBM background hidden layer units is equal to the pgRBM then the pgdRBM spends a little more time on the gradient estimate procedure.

Just like the pgDBN, the Point-wise Gated Dropout Deep Belief Networks (pgdDBN) can be created by stacking the pgdRBM and RBMs. The pgdDBN first utilizes the pgdRBM to pre-train the weights connecting the visible layer, the switch layer, and the first hidden layer, and then uses dRBM to obtain the weights between the hidden layers. After the pre-training, the pgdDBN utilizes the data with labels and the gradient descent method to fine-tune the weights.

4.2. Weight uncertainty in Point-wise Gated Deep Networks

The weight uncertainty is also a common method to solve the over-fitting problem in neural networks. The weight uncertainty method regards each weight in neural networks as a probability distribution of a possible value instead of the previous fixed value, so this method can learn more robust and useful features. Therefore, we can introduce the weight uncertainty into RBMs and propose Weight Uncertainty Restricted Boltzmann Machines (wRBM) [25–27]. Likewise, the performance of the wRBM is truly dissatisfactory when it deals with data containing irrelevant patterns. In the wRBM, the fluctuations in weights can be also interpreted as the change in the training data. We think
that the fluctuations affects the performance of the algorithm to a certain extent. Therefore, this paper introduces the weight uncertainty into pgwRBMs and propose Point-wise Gated Weight Uncertainty Restricted Boltzmann Machines (pgwRBM). That is, the connections between the visible layer, the switch layer, and the foreground hidden layer in pgwRBM can be regarded as probability distributions, and the rest weights are fixed values. If we assume that these distributions are Gaussian distributions, the connections between the visible layer, the switch layer, and the foreground hidden layer $\mathbf{W}^1$ are variables following Gaussian distribution. Thus the mean values and volatility variances are $\mu^1$ and $\sigma^1 = \log (1 + \exp (\rho^1))$ respectively. The energy function of the pgwRBM can be expressed as:

$$E(\mathbf{v}, \mathbf{z}, \mathbf{h}^1; \theta) = \sum_{i=1}^{D} \sum_{j=1}^{f} (z_{i}^{1} v_{i}) \left( \mu_{ij}^{1} + \log \left( 1 + \exp \left( \rho_{ij}^{1} \right) \right) \right) s_{ij}^{1} h_{j}^{1} - \sum_{i=1}^{D} \sum_{j=1}^{f} (z_{i}^{2} v_{i}) W_{ij}^{2} h_{j}^{2} - \sum_{j=1}^{2} \sum_{i=1}^{f} b_{ij}^{2} h_{j}^{2} - \sum_{i=1}^{f} \sum_{j=1}^{2} c_{ij}^{2} (z_{i}^{j} v_{i}),$$

$$s.t. \sum_{i=1}^{2} z_{i}^{j} = 1, i = 1, \ldots, D. \tag{25}$$

where the weights $\{\mu^1, \rho^1, c^1, b^1\}$ correspond to the foreground hidden layer $\mathbf{h}^1$ and the weights $\{\mathbf{W}^2, \sigma^2, \mathbf{b}^2\}$ correspond to the background hidden layer $\mathbf{h}^2$. We find that if the weights $\{\mu^1, c^1, b^1, \mathbf{W}^2, \sigma^2, \mathbf{b}^2\}$ are initialized using the RBM and the feature selection methods and the weights $\rho^1$ are initialized randomly then it will perform better. The connections between the visible layer, the switch layer, and the foreground hidden layer $\mathbf{W}^1$ can expressed as $\mathbf{W}^1 = \mu^1 + \log (1 + \exp (\rho^1)) \times \mathbf{e}^1$ where $\mathbf{e}^1 \sim N(0, 1)$. Then the conditional probabilities of the pgwRBM can be given by Eq. (6–8). When pgwRBM uses the Contrastive Divergence-k (CD-k) algorithm to estimate the gradients, the gradients of the weights $\{\mu^1, \rho^1, \mathbf{W}^2\}$ can be expressed as:

$$\Delta \mu_{ij}^1 = \alpha_w \left( \mathbb{P}(h_{j}^{1} = 1 | \mathbf{z}_{i}^{1(0), 0}, \mathbf{v}_{i}^{1(0)}(\mathbf{z}_{i}^{1(0), 0})\mathbf{v}_{i}^{1(0)}) - \mathbb{P}(h_{j}^{1} = 1 | \mathbf{z}_{i}^{1(k), 0}, \mathbf{v}_{i}^{1(k)}(\mathbf{z}_{i}^{1(k), 0})\mathbf{v}_{i}^{1(k)}) \right),$$

$$\Delta \rho_{ij}^1 = \Delta \mu_{ij}^1 \times \frac{\mathbf{e}_i^1}{1 + \exp (-\rho_{ij}^1)}, \tag{26}$$

$$\Delta W_{ij}^2 = \alpha_w \left( \mathbb{P}(h_{j}^{2} = 1 | \mathbf{z}_{i}^{2(0), 0}, \mathbf{v}_{i}^{2(0)}(\mathbf{z}_{i}^{2(0), 0})\mathbf{v}_{i}^{2(0)}) - \mathbb{P}(h_{j}^{2} = 1 | \mathbf{z}_{i}^{2(k), 0}, \mathbf{v}_{i}^{2(k)}(\mathbf{z}_{i}^{2(k), 0})\mathbf{v}_{i}^{2(k)}) \right). \tag{27}$$

Like the pgDBN and the pgdDBN, the Point-wise Gated Weight Uncertainty Deep Belief Networks (pgwDBN) are created by stacking the pgwRBM and wRBMs. The pgwDBN first utilizes the pgwRBM to pre-train the weights connecting the visible layer, the switch layer, and the first hidden layer, and then uses wRBMs to obtain the weights between the hidden layers. After the pre-training, the pgwDBN uses the data with labels and the gradient descent method to fine-tune the weights. If a pgwDBN has the same network as a pgDDBN then the training time of pgwDBN is close to the pgDDBN.

We also can stack the pgwRBM and wRBMs to create the Point-wise Gated Weight Uncertainty Deep Boltzmann Machines (pgwDBM). The pre-training process of the pgwDBN is similar to pgDBM. By taking a pgDBN having two hidden layer as an example, if the weights of the modified pgwRBM and the modified RBM are $\{\mu^1, \rho^1, c^1, b^1, \mathbf{z}^1, \mathbf{W}^2, \sigma^2, \mathbf{b}^2\}$ and $\{\mu^2, \rho^2, c^2, b^2\}$ respectively, then the weights of pgwDBM are initialized using $\{\mathbf{z}^1, c = c^1, \mu^1 = \mu^1, \rho^1 = \rho^1, \mathbf{b}^1 = \mathbf{b}^1 + c^2, \mu^2, \rho^2, \mathbf{b}^2\}$. We also use the data $\mathbf{v}$ and $\mathbf{z}^1$ compute a new data $\tilde{\mathbf{v}} = \mathbf{v} \times \mathbf{z}^1$ (where $\times$ denotes element-wise multiplication) and the energy function can be expressed as:

$$E(\tilde{\mathbf{v}}, \mathbf{h}^{1(1)}, \mathbf{h}^{2(2)}; \theta) = \sum_{i=1}^{D} c_{i} \tilde{v}_{i} - \sum_{i=1}^{D} \sum_{j=1}^{f^{(1)}} \left( \mu_{ij}^{(1)} + \log \left( 1 + \exp \left( \rho_{ij}^{(1)} \right) \right) \right) s_{ij}^{(1)} h_{j}^{(1)} - \sum_{j=1}^{f^{(2)}} b_{ij}^{(2)} h_{j}^{(2)} + \sum_{j=1}^{f^{(1)}} \left( \mu_{ij}^{(2)} + \log \left( 1 + \exp \left( \rho_{ij}^{(2)} \right) \right) \right) s_{ij}^{(2)} h_{j}^{(2)} - \sum_{j=1}^{f^{(2)}} b_{ij}^{(2)} h_{j}^{(2)} \tag{28}$$

The pgwDBM also makes use of mean field method and the Contrastive Divergence-k (CD-k) or other gradient estimation algorithms to fine-tune the weights of the network, and the details are given in Algorithm 2. Next, we sample $\mathbf{e}^{(1)}$ and $\mathbf{e}^{(2)}$ and then compute $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$. Similar to the pgDBM, we also need to compute the probability distribution of the second hidden layer $Q(\mathbf{h}^{2})$. Then, the data $\tilde{\mathbf{v}}$ is combined with $Q(\mathbf{h}^{2})$ to form the inputs of the classified network. In the classified network, the weights of the first two layers are $\{\mathbf{c}^{(2)}, \mathbf{W}^{(2)}; (\mathbf{W}^{(2)})^{T}, \mathbf{b}^{(2)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}\}$. Finally, similar to pgDBM, the weights connecting the second hidden layer and the output layer are random initialized and then the gradient descent method is used to fine-tune the weights of this classified network. If a pgwDBM has the same network as a pgDBM then the training time of pgwDBM is close to the pgDBM.
Algorithm 2 Fine-tuning the weights of a two-layer pgwDBM for classification

1. Given: a train set of $N$ data vectors $\{v_i\}_{i=1}^N$, and the number of Gibbs sampling $K$.

2. // Use the learned weights of the modified pgwRBM $(\mu', \rho', c', b', W', c, b)$ to learn a new data:
3. Sample $v' \sim N(0, I)$ and compute $W^v = \mu^v + \log(1 + \exp(\rho^v)) \cdot \times v'$.
4. for each training sample $v^n, n = 1$ to $N$
5. define a vector $z = 1$, run mean field updatas until convergence:
   $h^i = \sigma((v' \cdot z)(2W' + 2b'), // \cdot \cdot \cdot$ denotes element-wise multiplication.
   $h^i = \sigma((W' \cdot (1-\cdot z))(2W' + 2b'), // \cdot \cdot \cdot = \sigma(1/(1+\exp(-x))).
   z = \sigma(v' \cdot ((h^i(W')^T + c') - (h^i(W')^T + c'))) // \cdot \cdot \cdot$
6. Set $z' = z$. // \cdot \cdot \cdot is related to the foreground hidden layer.
7. end for
8. Compute the new data $\tilde{v} = v \cdot z$. // \cdot \cdot \cdot is related to the foreground hidden layer.
9. Obtain the learned weights $(c, \mu^{(1)}, \rho^{(1)}, b^{(1)}, \mu^{(2)}, \rho^{(2)}, b^{(2)})$ in the greedy pre-training procedure.
10. for $t = 1$ to $T$ (the number of iterations) do
11. Sample $v^{(1)} \sim N(0, I)$ and compute $W^{(1)} = \mu^{(1)} + \log(1 + \exp(\rho^{(1)})) \cdot \times v^{(1)}$.
12. Sample $v^{(2)} \sim N(0, I)$ and compute $W^{(2)} = \mu^{(2)} + \log(1 + \exp(\rho^{(2)})) \cdot \times v^{(2)}$.
13. // Variational Inference:
14. for each new sample $v^n, n = 1$ to $N$ do
15. define $h^{(1)} = \sigma(v^{(1)}(2W^{(1)}) + b^{(1)}), h^{(2)} = \sigma(h^{(1)}W^{(2)} + b^{(2)})$, and run mean field updatas until convergence:
   $h^{(1)} = \sigma((v^{(1)}W^{(1)} - h^{(2)})(W^{(2)})^T + b^{(2)}), h^{(2)} = \sigma(h^{(1)}W^{(2)} + b^{(2)}$)
16. Set $\mu^n^{(1)}, \mu^n^{(2)} = h^{(2)}$.
17. end for
18. // Stochastic Approximation:
19. Initialize $M$ samples: $(\tilde{v}^{(1)}, h^{(1)}, b^{(1)}), \ldots, (\tilde{v}^{(K)}, h^{(1)}, b^{(1)}).
20. Make use of $\mu^n^{(1)}, \mu^n^{(2)}$ to obtain binary vectors $(h^{(1)}, b^{(2)})$ and $\tilde{v} = \tilde{v}$.
21. for $k = 1$ to $K$ do
22. for each new sample $\tilde{v}^n, n = 1$ to $N$ do
23. Sample $(\tilde{v}^{(1), k}, h^{(1), k}, b^{(2), k})$ given $(\tilde{v}^{(1), k-1}, h^{(1), k-1}, b^{(2), k-1})$ by running a Gibbs sampler (Eqs. 6-8).
24. end for
25. end for
26. // Parameter Update:
27. $\mu^{(1)} = \mu^{(1)} + \alpha \left(\sum_{m=1}^{M} \tilde{v}^{(m)} \cdot (\mu^{(m)} - \sum_{m=1}^{M} \tilde{v}^{(m)} h^{(m)}) / N\right.$.
28. $\rho^{(1)} = \rho^{(1)} + \alpha \left(\sum_{m=1}^{M} \tilde{v}^{(m)} \cdot (\mu^{(m)} - \sum_{m=1}^{M} \tilde{v}^{(m)} h^{(m)}) \times v^{(1)}, \times \sigma(v^{(1)}) / N\right.$.
29. $\mu^{(2)} = \mu^{(2)} + \alpha \left(\sum_{m=1}^{M} \mu^{(m)} \cdot (\mu^{(m)} - \sum_{m=1}^{M} \tilde{v}^{(m)} h^{(m)}) / N\right.$.
30. $\rho^{(2)} = \rho^{(2)} + \alpha \left(\sum_{m=1}^{M} \mu^{(m)} \cdot (\mu^{(m)} - \sum_{m=1}^{M} \tilde{v}^{(m)} h^{(m)}) \times v^{(2)}, \times \sigma(v^{(2)}) / N\right.$.
31. $c = c + \alpha \left(\sum_{m=1}^{M} \tilde{v}^{(m)} - \sum_{m=1}^{M} \tilde{v}^{(m)} / N\right.$.
32. $b^{(1)} = b^{(1)} + \alpha \left(\sum_{m=1}^{M} \mu^{(m)} - \sum_{m=1}^{M} \tilde{v}^{(m)} h^{(m)} / N\right.$.
33. $b^{(2)} = b^{(2)} + \alpha \left(\sum_{m=1}^{M} \mu^{(m)} - \sum_{m=1}^{M} \tilde{v}^{(m)} h^{(m)} / N\right.$.
34. Decrease $\alpha$. // \cdot \cdot \cdot
Table 1
Details of benchmark data sets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Number of samples</th>
<th>Dimension</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST back-image</td>
<td>10000</td>
<td>784</td>
<td>10</td>
</tr>
<tr>
<td>MNIST back-random</td>
<td>10000</td>
<td>784</td>
<td>10</td>
</tr>
<tr>
<td>MNIST rotated + back-image</td>
<td>10000</td>
<td>784</td>
<td>10</td>
</tr>
<tr>
<td>Rectangles-images</td>
<td>10000</td>
<td>784</td>
<td>2</td>
</tr>
</tbody>
</table>

![Image](image_url)

(a) MNIST back-image  
(b) MNIST back-random  
(c) MNIST rotated+back-image  
(d) Rectangles-images

Table 2
Error rates of shallow algorithms on different test data sets.

<table>
<thead>
<tr>
<th>Algorithms/Datasets</th>
<th>RBM</th>
<th>wRBM</th>
<th>dRBM</th>
<th>pgRBM</th>
<th>pgwRBM</th>
<th>pgdRBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST back-image</td>
<td>15.83</td>
<td>15.83</td>
<td>33.22</td>
<td>15.01</td>
<td>14.65</td>
<td>14.29</td>
</tr>
<tr>
<td>MNIST back-random</td>
<td>11.05</td>
<td>10.95</td>
<td>36.74</td>
<td>10.15</td>
<td>10.00</td>
<td>9.95</td>
</tr>
<tr>
<td>MNIST rotated + back-image</td>
<td>47.47</td>
<td>47.34</td>
<td>67.31</td>
<td>44.32</td>
<td>44.34</td>
<td>44.15</td>
</tr>
<tr>
<td>Rectangles-images</td>
<td>22.42</td>
<td>22.64</td>
<td>28.12</td>
<td>25.37</td>
<td>28.75</td>
<td>24.94</td>
</tr>
</tbody>
</table>

5. Experimental results

5.1. Experimental setup and datasets

In order to test the performance of the algorithms, the proposed algorithms are compared with the RBM, wRBM, dRBM, pgRBM, DBN, and DBM algorithms on four datasets containing irrelevant patterns [28]. All these algorithms are carried out in a work station with a core i7 DMIi2-Intel 3.6 GHz processor and 18 GB RAM running MATLAB 2012B. The datasets used in this paper are MNIST back-image, MNIST back-random, MNIST rotated + back-image, Rectangles-images datasets, and each dataset contains training dataset $l$, validation dataset $v$, and testing dataset $t$. The details of four datasets are shown in Table 1, and Fig. 4 gives legends of four data sets in turns.

In this paper, all algorithms uses mini-batch learning on four datasets, and the mini-batch size is set to 100. The number of hidden layer units in the RBM and the wRBM is set to 500 or 1000. In the dRBM, $p$ is set to 0.5 or 0.8, and the product of the number of hidden layer units and $p$ is set to 500 or 1000. In pgdRBM, $p$ is set to 0.8. The pgRBM, the pgwRBM, and the pgdRBM all use the RBM having 1200 hidden layer units initialize the weights, and the number of the foreground or background hidden layer units or the product of the number of the foreground hidden layer units and $p$ is set to 500. The hidden layers structures of DBM, pgDBM, and pgwDBM are all set to 500-500. The hidden layers structures of DBN, pgDBN, and pgwDBN select from (500-500-500, 500-500-1000, 500-500-2000). In dDBN and pgdDBN, $p$ in the input layer is set to 0.8 and $p$ in all the hidden layers is set to 0.5. The hidden layers structures of dDBN and pgdDBN select from (625–1000–1000, 625–1000–2000, 625–1000–4000). For all algorithms, backpropagation is used to calculate the gradients and Conjugate gradient optimization was used on minibatches of size 1000 for 100 epochs, and early stopping is used according to the error rate on the validation dataset with a look ahead of 10 iterations.

5.2. Comparisons with related algorithms

The error rates of the shallow algorithms on four datasets are shown in Table 2. By comparing the RBM with the pgRBM, we can see that the pgRBM outperforms the RBM on three MNIST variation datasets and underperforms the RBM on the Rectangles-images. Fig. 5 gives the learning results of the pgRBM on MNIST back-image and Rectangles-images datasets. We can learn that the learning background images on MNIST back-image seem task-irrelevant and the learning background images on Rectangles-images contain the task-relevant information. Therefore, the pgRBM doesn’t perform the ideal results in all the background image datasets, and how to apply pgRBM to the datasets similar to Rectangles-images is our next research direction. From Table 2, we can also find that: 1) the wRBM underperforms the RBM on two datasets and the dRBM underperforms the RBM on all the datasets; 2) the pgwRBM performs slightly worse than the pgRBM on MNIST rotated + back-image and the pgdRBM outperforms the RBM on all the datasets. We can make conclusion that dropout and weight uncertainty methods are efficient for preventing overfitting in the pgRBM.
Table 3 shows the error rates of the deep algorithms related to DBM on MNIST variation datasets. We can find that the pgDBM outperforms the DBM. The pgRBM can effectively find the task-relevant patterns from data containing irrelevant patterns, and the pgDBM stacks the pgRBM and RBMs to create deep networks. Experimental results show that the pgDBM performs better learning abilities than the pgRBM and the DBM when dealing with data containing irrelevant patterns. From Table 3, we can also see that: 1) the pgDBM performs slightly worse than the pgDBM on MNIST back-random; 2) the pgwDBM outperforms the pgwDBM on all the MNIST variation datasets. We can make conclusion that the pgwDBM algorithm is an effective neural network learning algorithm.

Table 4 shows the error rates of the deep algorithms related to DBN on MNIST variation datasets. We can see that the pgDBN outperforms the DBN on three MNIST variation datasets. Experimental results show that the pgDBN is an effective deep network model. From Table 4, we can also see that: 1) the wDBN outperforms the DBN and the dDBN underperforms the DBN on all the datasets; 2) the pgwDBN and the pgdDBN perform slightly worse than the pgDBN on all the datasets. Experimental results show that dropout and weight uncertainty can’t effectively solve overfitting in the pgDBN. But dropout and weight uncertainty in DBNs can perform well on the data without irrelevant patterns, so we think the cause might be the “cleaned” data learned from pgRBM still contains some irrelevant patterns. By comparing the RBM, the DBN, and the DBM, we can see that the DBN and the DBM perform worse than the RBM when dealing with data containing irrelevant patterns. The weights of DBNs and DBMs are both initialized through the data and RBMs layer by layer. The data containing irrelevant patterns can lead to the expected results that not all of the outputs of the first hidden layer are useful features and a substantial part of the outputs contain the information about irrelevant patterns. We realize that irrelevant patterns in data lead to bad intermediate level representations and affect the performance of DBNs and DBMs. We can also learn that the pgRBM and the pgDBN are effective deep neural networks learning methods, which also confirms that if the data is “cleaned” then the DBN and the DBM will perform well. Meanwhile, we find that dropout in both DBNs and RBMs can lead to the high error rates on three MNIST variation datasets. We know that not all of the outputs of hidden layers in RBMs and DBNs are useful features when they deal with data containing irrelevant patterns. And the key idea of the dropout in RBMs and DBNs is to randomly drop hidden units from the networks during training. In other words, the dropout in these two model can be interpreted as adding noise to the hidden layer units which have contained a large amount of invalid information in addition to useful features, which is the most likely reason why dRBM and dDBN perform bad. pgdRBM and pgdDBN handling the data containing irrelevant patterns well seems to prove the point.

6. Conclusions

Point-wise Gated Restricted Boltzmann Machines are effectively shallow learning methods for data containing irrelevant patterns. On this basis, this paper introduces task-relevant patterns in data learned from the pgRBM into the DBN and the DBM and presents Point-wise Gated Deep Belief Networks and Point-wise Gated Deep Boltzmann Machines. Experimental results show that the pgDBN and the pgDBM perform better than the pgRBM on MNIST variation datasets. Then, we introduce dropout and weight uncertainty methods into the pgRBM, the pgDBN, and the pgDBM and discuss the validities of these methods. Experimental results shows that dropout and weight uncertainty methods are efficient for preventing overfitting in the pgRBM. However, weight uncertainty in pgDBMs gets barely satisfactory results.
and dropout and weight uncertainty in pgDBNs fail to achieve the expected results. We think the cause might be that the “cleaned” data learned from pgRBM still contains some irrelevant patterns. And we also find that the pgRBM does not achieve the desired results on all the background image data sets. Therefore, our next research direction is how to let pgRBM learn the more “cleaned” data and apply it to the datasets similar to Rectangles-images.

Acknowledgements

This work is supported by the National Natural Science Foundations of China (Nos.61379101, 61672522), the National Key Basic Research Program of China (No.2013CB329502), the Graduate Student Scientific Research Innovation Project in Jiangsu Province (No. KYZZ16.0215), the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD), and the Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology(CICAEET).

References