Training Restricted Boltzmann Machines with Overlapping Partitions

Hasari Tosun and John W. Sheppard

Montana State University,
Department of Computer Science, Bozeman, Montana, USA

Abstract. Restricted Boltzmann Machines (RBM) are energy-based models that are successfully used as generative learning models as well as crucial components of Deep Belief Networks (DBN). The most successful training method to date for RBMs is the Contrastive Divergence method. However, Contrastive Divergence is inefficient when the number of features is very high and the mixing rate of the Gibbs chain is slow. We propose a new training method that partitions a single RBM into multiple overlapping small RBMs. The final RBM is learned by layers of partitions. We show that this method is not only fast, it is also more accurate in terms of its generative power.

Keywords: Restricted Boltzmann Machine, Machine Learning.

1 Introduction

The Restricted Boltzmann Machine was introduced by Hinton *et al.* as a parallel network for constraint satisfaction [1]. Since computing the partition function in the Boltzmann distribution is not tractable, training was initially inefficient, and RBMs did not gain popularity for seventeen years until Hinton *et al.* developed Contrastive Divergence, a method based on Gibbs Sampling [8]. Since then, RBMs are used as basic components of deep learning algorithms [3,7,9]. RBMs have also been successfully applied to classification tasks [5,10,12]. Moreover, RBMs have been applied to many other learning tasks including Collaborative Filtering [14].

As RBMs became popular, research on training them efficiently increased. Tieleman modified the Contrastive Divergence method by making Markov chains persistent [15]. Thus, the Markov chain is not reset for each training example. This has been shown to outperform Contrastive Divergence with one step, *CD-1*, with respect to classification accuracy. However, it does not address the problem of training speed. Brekal *et al.* introduced an algorithm to parallelize training RBMs using parallel Markov chains [4]. Resulting Markov chains need to share messages and the gradient is estimated by averaging chains.

In the context of Deep Belief Networks (DBN), the DistBelief model and data parallelization framework was developed by [6]. Here, the DBN model is partitioned into parts. Overlapping parts then exchange messages. Moreover, models are replicated in different computation nodes and trained on different subsets of data to provide data parallelization.

T. Calders et al. (Eds.): ECML PKDD 2014, Part III, LNCS 8726, pp. 195–208, 2014.

In this paper, we propose a novel algorithm, *RBM-Partition*, for training RBMs that splits a single RBM into multiple partitions. Each partition then trains on a subsection of the data instance. We explore the effects of permitting these partitions to overlap to improve training across the boundaries of the partitions. We then investigate the generative power of model and find that this training process improves training performance of CD-1 in terms of both generative power and speed.

The rest of this paper is organized as follows. In Section 2 we briefly introduce the Boltzmann Distribution and RBMs. We describe our training method in Section 3 and show experimental results in Section 4. Finally, we discuss future work in Section 5.

2 Restricted Boltzmann Machines

In statistical mechanics, the Boltzmann distribution is the probability of a random variable that realizes a particular energy level (Equation 1) [11]. Here $\beta = \frac{1}{kT}$ where T is temperature and k is the Boltzmann constant. In machine learning, β is usually set to 1, except in the context of algorithms such as simulated annealing. Z is the partition function, which is generally intractable to compute. However, when Z is computable, all other properties of the system such as entropy, temperature, etc. can be calculated. The equation for Z, which summarizes over the micro states of the system, is shown in Equation 2.

$$p(\mathbf{x}) = \frac{e^{-\beta E(\mathbf{x})}}{Z} \tag{1}$$

$$Z = \sum_i \left(e^{-\beta E(x_i)} \right) \tag{2}$$

As a type of Hopfield Network, an RBM is a generative model with visible and hidden nodes as shown in Figure 1. There are no dependencies between hidden nodes, or between visible nodes, thus an RBM forms a bipartite graph. The model represents a Boltzmann energy distribution [11], where the probability distribution of the RBM with visible (x) and hidden node (h) is given in Equation 3.

$$p(\mathbf{x}, \mathbf{h}) = \frac{e^{-E(\mathbf{x},\mathbf{h})}}{Z} \tag{3}$$

If we marginalize over the hidden variables, we obtain the probability of the visible variables $p(\mathbf{x}) = \sum_h \left(\frac{e^{-E(\mathbf{x},h)}}{Z} \right)$. Inspired from statistical mechanics, we write $p(\mathbf{x})$ in terms of Free Energy (A) as follows:

$$A(\mathbf{x}) = -\log \left(\sum_h e^{-E(\mathbf{x},h)} \right) \tag{4}$$

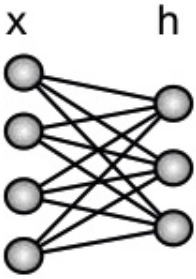

Fig. 1. Restricted Boltzmann Machine

Thus, rewriting $p(\mathbf{x})$ results in

$$p(\mathbf{x}) = \frac{e^{-A(\mathbf{x})}}{Z} \tag{5}$$

The partition function, $Z = \sum_x e^{-A(x)}$. The energy function of an RBM is given in the following equation.

$$E(\mathbf{x}, \mathbf{h}) = -b\mathbf{x} - c\mathbf{h} - \mathbf{h}\mathbf{W}\mathbf{x}.$$

If θ represents the model parameters, then the gradient of the log-likelihood is calculated as in Equation 6. The gradient contains two terms that are referred as the *positive* and the *negative* terms respectively. The first term increases the probability of the training data by decreasing free energy while the second term decreases the probability of a sample generated by the model. Computing the expectation over the first term is tractable; however, for the second term it is not. Thus, Hinton introduced the Contrastive Divergence algorithm that uses Gibbs sampling to estimate the second term [8].

$$\frac{-\partial \log(p(x))}{\partial \theta} = \frac{\partial A(x)}{\partial \theta} - \sum_{\tilde{x}} p(\tilde{x}) \frac{\partial A(\tilde{x})}{\partial \theta} \tag{6}$$

We provide a more detailed description of the CD algorithm in Section 4. An alternative description of Contrastive Divergence algorithm is given by Bengio [2].

CD provides a reasonable approximation to the likelihood gradient. The CD-1 algorithm (i.e, Contrastive Divergence with one step) is usually sufficient for many applications; for CD-k, resetting the Markov chain after each parameter update is inefficient because the model has already changed [15,2].

3 Partitioned Restricted Boltzmann Machines

We propose a training method for RBMs that partitions the network into several overlapping subnetworks. With our method, training involves several partition steps. In each step, the RBM is partitioned into multiple RBMs as shown in Figure 2. In this figure, the partitions do not overlap; we discuss the version with overlap later in this section These partitioned RBMs are trained in parallel with a corresponding partition of training data using CD-1. In other words, the feature vector is also partitioned, and each individual RBM is trained on a section of that feature vector. Once all partitions are trained, we generate another set of RBMs with fewer splits. For example, in Figure 2, we initially generate four RBMS. In the second step, we generate two, and final training occurs on the full RBM. It should be noted that the training process in all steps is over the same weight vector.

The motivation behind our approach is that when RBMs are small, they can be trained with more training epochs. However, as we reduce the number of splits, training requires fewer epochs and therefore less time to train. The pseudo-code for our training procedure is given in Algorithm 1. Since the details for overlapping partitions is omitted, we added notes where overlapping partitions will need different logic.

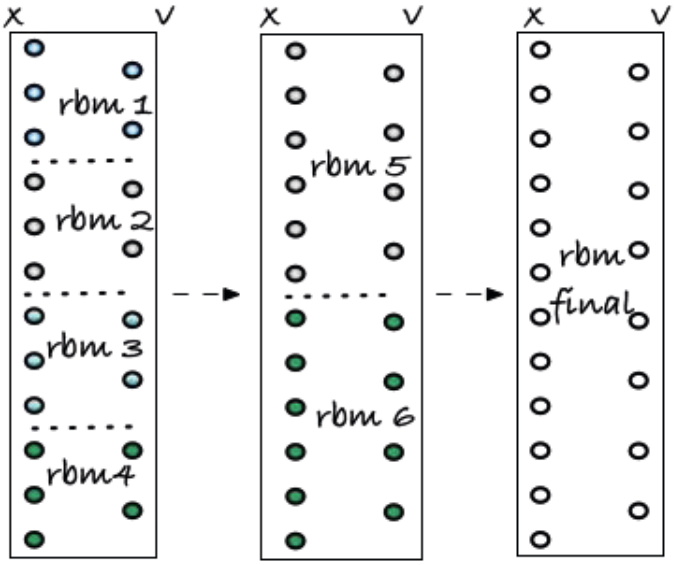

Fig. 2. RBM Partitions

Algorithm 1. Training with Partitions

partition_configurations: a list of configurations that describe splits and training instances for each training step
$W \leftarrow$ Create and initialize weight vector
$vbias \leftarrow$ Create and initialize bias vector for visible layer
$hbias \leftarrow$ Create and initialize bias vector for hidden layer
for each $configuration$ in $partition\_configurations$:
    $train \leftarrow$Training instances
    train_partitions($configuration, train, visible, hidden, W, vbias, hbias$)

Algorithm 2. create_rbm_partitions(configuration, W, vbias, hbias)

//for overlapping, *visible_nodes* and *hidden_nodes* will increase based
    on percentage of overlap
$visible\_nodes \leftarrow configuration.visible/configuration.splits$
$hidden\_nodes \leftarrow configuration.hidden/configuration.splits$
rbms: RBM list
for i in $configuration.splits$:
    //Each RBM will operate on a region of the visible vector
      and hidden vector.
    //For overlapping partitions, *vbase* and *hbase* will change based
      on overlap percentage
    vbase $\leftarrow$base index in visible vector
    hbase $\leftarrow$base index in hidden vector
  $rbm(i) \leftarrow$RBM($W, vbias, hbias, vbase, hbase, visible\_nodes, hidden\_nodes$)
return rbms

Algorithm 3. train_partitions(configuration, train, visible, hidden, W, vbias, hbias)

$rbm\_list \leftarrow$***create_rbm_partitions***($configuration, W, vbias, hbias$)
for each instance in *train*:
    //for overlapping, partition will change according to
      $configuration.overlap$ percentage
    splits $\leftarrow$split instance into number of configuration.splits partitions
    for *rbm* in *rbms_list*
        $rbm(i).contrastive\_divergence(splits(i))$

Based on the intuition that neighboring RBMs may share some features (nodes), for overlapping partitions, we define similar partitions as described above. However, in this model, each partition has some percent of its nodes overlap with its neighboring partitions. As shown in Figure 3, the RBMs are

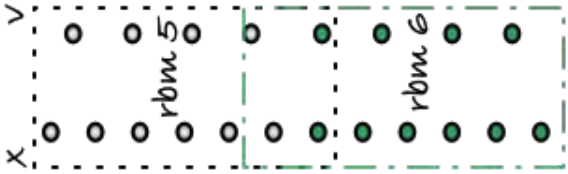

Fig. 3. RBM With Overlapping Partitions

sharing two hidden and two visible nodes. Since nodes are shared, partitioned RBMs cannot be trained concurrently without some kind of message passing. It should be noted that when trained sequentially, message passing between the partitions is not required.

4 Experimental Results

The MNIST dataset is used for our experiments due to its wider association with RBMs. MNIST has 60,000 training samples and 10,000 test samples of images. Each image is 28×28 pixels corresponding to handwritten digits from 0 to 9. Some sample images are presented in Figure 4. We measure performance of our method using reconstruction error, which is defined to be the average pixel differences between the original and reconstructed images (Equation 7). For each epoch, we use a batch size of 10 images from the training samples. Thus, for 6,000 epochs, 60,000 samples are used for training. Unless stated otherwise, we use CD-1 for all training steps. The unpartitioned RBM has 500 hidden nodes and $28 \times 28 = 784$ visible nodes. To have a fair comparison in terms of performance, rather using CPU time, we applied the following method. CD-1 training for one sample is carried out as follows:

- For all hidden nodes, find the probability of hidden node h_i as $\sigma(c_i + \sum_j W_{ij}x_j)$ and sample h_{i1} from a binomial distribution given h_i.
- For all visible nodes, find the probability of visible node x_j as $\sigma(b_j + \sum_i W_{ij}h_{i1})$ and sample x_{j1} from a binomial distribution given x_j.
- For all hidden nodes, find the probability of hidden node h_{i2} as $\sigma(c_i + \sum_j W_{ij}x_{j1})$.
- Calculate the gradient:
  - $W = W + \epsilon(h_{i1}x_j - h_{2i}x_{j1})$ where ϵ is the learning rate.
  - $b = b + \epsilon(x_j - x_{j1})$
  - $c = c + \epsilon(h_{i1} - h_{i2})$

where $\sigma(x) = \frac{1}{1+e^{-x}}$. Since operations at each step of CD involves $visible\_nodes \times hidden\_nodes$ updates, we estimate that the total number of Markov chain calculations is

$$ChainOperations = visible\ nodes \times hidden\ nodes \times samples$$

Fewer chain operations translate into less CPU time.

We used reconstruction error to compare our algorithms. For reconstruction error, we first obtain the binary representation of the original image and the reconstructed image. 30 is chosen as the threshold for converting pixel values [0-255] to binary 0 or 1. Thus, pixel values greater than or equal to 30 are set to 1 while values less than 30 are set to 0. Then, the reconstruction error is calculated as in Equation 7.

$$error = \frac{\sum_i (image(i) \neq reconstructedImage(i))}{total\ pixels} \tag{7}$$

Table 1 shows the results of our first experiment where the learning rate is set to 0.1 for all RBMs. *Single RBM* represents a fully connected RBM that is used as a baseline for comparison. The training sample for each RBM is equal to the number of epochs times the batch size (10). For instance, the Single RBM algorithm is trained on 60,000 images. Moreover, we use each image sample once. Unless stated otherwise, for following experiments, we ran training algorithms on samples for one iteration only—at the most, each sample is used only once. Each RBM-X represents a step with X partitions. Samples chosen for RBM-X are always from first N samples of total images. RBM-1 represents the final model. For these experiments, partitions are trained sequentially. Thus, if we train them concurrently, the total *ChainOperations* will be lower. As compared to Single RBM, RBM-1 has significantly lower reconstruction error. The total *ChainOperations* for partitioned RBMs is also less than Single RBM. In the table, using a t-test, significant results with 99% confidence are shown in bold. RBM-Partition after training on 20 partitions, significantly outperformed the Single RBM. Furthermore, the total number of chain operations for RBM-Partition is substantially less than for Single RBM.

Since we want fast convergence in the first step, in the following experiment we varied the learning rate to enable this. Results are shown in Table 2.

Table 1. Training Characteristics

Configuration	Number of RBMs	Epochs (batch=10)	Learning Rate	Reconstruction Error (%)	Chain Operations (10^9)
Single RBM	1	6000	0.1	3.85	23.52
RBM-28	28	6000	0.1	4.76	0.84
RBM-20	20	5000	0.1	4.03	0.98
RBM-15	15	4000	0.1	**3.19**	1.05
RBM-10	10	3000	0.1	**2.67**	1.18
RBM-5	5	2500	0.1	**2.29**	1.96
RBM-2	2	2000	0.1	**2.33**	3.92
RBM-1	1	2000	0.1	**2.36**	7.84
Total					**17.77**

Table 2. Training Characteristics wrt Learning Rate

Configuration	Number of RBMs	Epochs (batch=10)	Learning Rate	Reconstruction Error (%)	Chain Operations (10^9)
RBM-28	28	6000	0.3	**3.23**	0.84
RBM-20	20	5000	0.3	**2.93**	0.98
RBM-15	15	4000	0.3	**2.66**	1.05
RBM-10	10	3000	0.25	**2.30**	1.18
RBM-5	5	2500	0.20	**2.08**	1.96
RBM-2	2	2000	0.10	**2.10**	3.92
RBM-1	1	2000	0.10	**2.10**	7.84
Total					**17.77**

Table 3. Training Characteristics wrt Learning Rate

	$lr = 0.3$	$lr = 0.1$	$lr = 0.05$	$lr = 0.005$	$lr = 0.0005$
Single RBM	4.43	3.85	4.22	9.40	23.15
RBM-1	**3.45**	**2.10**	**1.95**	**1.83**	**1.92**

Table 4. Overlapping Partitions

Configuration	Number of RBMs	Epochs (batch=10)	Learning Rate	Reconstruction Error (%)	Chain Operations (10^9)
RBM-28	28	6000	0.3	**3.11**	0.90
RBM-20	20	5000	0.3	**2.76**	1.10
RBM-15	15	4000	0.3	**2.50**	1.18
RBM-10	10	3000	0.25	**2.19**	1.35
RBM-5	5	2500	0.20	**1.95**	2.27
RBM-2	2	2000	0.10	**1.92**	4.30
RBM-1	1	2000	0.10	**2.08**	7.84
Total					**18.94**

The RBM-Partition with 99% confidence outperforms the Single RBM in all steps including the first partition, RBM-28 (with 28 partitions). Moreover, reconstruction errors are even lower compared to our previous experiment.

Using the same configuration above, we varied the learning rate (denoted lr in the results) for the Single RBM and RBM-1. Learning rates for other RBM-X are fixed as in the configuration given in Table 2. Reconstruction errors for different learning rates are given in Table 3. Results demonstrate that RBM-Partition is less sensitive to different learning rates as compared to the Single RBM with 99% confidence.

We also wanted to determine if overlapping partitions would affect the results. We ran our experiment with 5% overlap, which means that each RBM shares 5% of its neighbor's nodes (5% from the left neighbor and 5% from the right neighbor). We ran overlapping partitions sequentially. As shown in Table 4, reconstruction errors are even lower with only a modest increase in overhead in terms of *ChainOperations*.

Comparing overlapping with non-overlapping RBM-Partition algorithms using the t-test, results show that the overlapping algorithm outperforms the non-overlapping algorithm with 99% confidence in almost every stage. However, in the last stage, the results were not significantly different, as shown in Table 5. We hypothesize that since overlapping partitions have more connections in each partition, they will require more training samples.

Table 5. Non-overlapping vs. Overlapping Partitions

Configuration	Number of RBMs	Epochs (batch=10)	Learning Rate	Overlapping Reconstruction Error(%)	NonOverlapping Reconstruction Error(%)	Overlapping Chain Operations (10^9)	NonOverlapping Chain Operations (10^9)
RBM-28	28	6000	0.3	**3.11**	3.23	0.90	0.84
RBM-20	20	5000	0.3	**2.76**	2.93	1.10	0.98
RBM-15	15	4000	0.3	**2.50**	2.66	1.18	1.05
RBM-10	10	3000	0.25	**2.19**	2.30	1.35	1.18
RBM-5	5	2500	0.20	**1.95**	2.08	2.27	1.96
RBM-2	2	2000	0.10	**1.92**	2.10	4.30	3.92
RBM-1	1	2000	0.10	2.08	2.10	7.84	7.84
Total						**18.94**	**17.77**

Table 6. 10-Fold Cross Validation Results

Configuration	Number of RBMs	Samples	Learning Rate	**No Overlap**: Average Reconstruction Error (%)	**Overlap**: Average Reconstruction Error(%)	Chain Operations per fold (10^9) No-Overlap/Overlap
Single RBM	1	60000	0.1	4.06		21.12
RBM-28	28	60000	0.3	**3.32**	**3.32**	0.76/0.81
RBM-20	20	50000	0.3	**3.07**	**2.92**	0.88/1.00
RBM-15	15	40000	0.3	**2.68**	**2.62**	0.94/1.06
RBM-10	10	30000	0.25	**2.35**	**2.29**	1.06/1.21
RBM-5	5	25000	0.20	**2.12**	**2.09**	1.76/2.04
RBM-2	2	20000	0.10	**2.15**	**2.08**	3.53/3.88
RBM-1	1	20000	0.10	**2.18**	**2.14**	7.06/7.06
Total						**16.00/17.06**

Fig. 4. Original vs. Reconstructed Images

Finally, 10-fold cross validation results are given in Table 6. Rather than using the provided training and test data sets. we pooled all of the data and split samples into 10 equal size subsamples. One subsample was used as the validation data for testing and the remaining 9 subsamples were used for training. We repeated this process 10 times. It should be noted that the numbers of samples for partitioned RBMs are not equal (Table 6) because we wanted to keep the total time complexity of RBM-Partition to be no worse than the Single RBM. RBM-Partition outperforms Single RBM with 99% confidence. Moreover, overlapping RBMs have lower average reconstruction error as compared to non-overlapping ones.

To visually compare the original images with the some of our reconstructed images, we present some examples in Figure 4.

Learning behavior with respect to the number of training samples is given in Figure 5. We compare RBM-10 with RBM Single. After each training cycle where we add 10,000 more images, we tested the algorithms on 10,000 images. RBM-10 outperforms RBM Single with 99% confidence on all training steps.

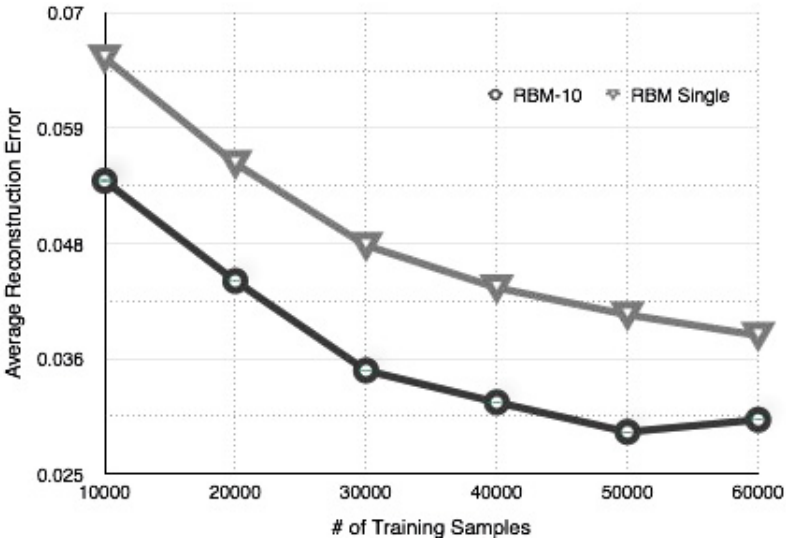

Fig. 5. Reconstruction Error vs. Training Samples

As we described at the begining of the Section 4, so far we ran these experiments for one iteration only. To see how our learning method will behave with additional iterations, we ran RBM-Partition and Single RBM for 15 iterations. Results are shown in Table 7. Starting with RBM-10, RBM-Partition significantly outperforms Single RBM with 99% confidence. For RBM-Partition, on average, the error is approximately 5 pixels out of 28×28 pixels, whereas it is 10 pixels for Single RBM.

The *Special Database 19* dataset from the National Institute of Standards and Technology (NIST) is the official training dataset for handprinted document and character recognition from 3600 writers, including 810K character images and 402K handwritten digits. Unlike the MNIST dataset, images are 128 by 128 pixels. We selected 62K images for training and testing. The dataset consists of 62 types of images for lowercase and uppercase letters, and numbers. Thus, in our dataset each type has 1,000 images. We used 10% for testing and 90% for training. 1-fold validation results are shown in Table 8. Based on the

Table 7. Training Iterations

Configuration	Number of RBMs	Samples	Learning Rate	Reconstruction Error (%)
Single RBM	1	60000	0.05	1.29
RBM-28	28	60000	0.3	1.75
RBM-20	20	30000	0.3	1.45
RBM-15	15	20000	0.3	1.35
RBM-10	10	20000	0.25	**1.08**
RBM-5	5	20000	0.2	**0.94**
RBM-2	2	20000	0.1	**0.75**
RBM-1	1	30000	0.05	**0.67**

Table 8. Training Characteristics with NIST dataset

Configuration	Number of RBMs	Training Samples	Learning Rate	Reconstruction Error (%)	Chain Operations (10^9)
Single RBM	1	62000	0.1	4.82	507.90
RBM-28	28	62000	0.3	**3.74**	19.92
RBM-20	20	50000	0.3	**3.65**	24.09
RBM-15	15	40000	0.3	**3.70**	25.19
RBM-10	10	30000	0.25	**3.69**	28.70
RBM-5	5	25000	0.20	**3.63**	47.78
RBM-2	2	20000	0.10	**3.67**	90.14
RBM-1	1	20000	0.10	**3.74**	163.8
Total					**399.62**

Table 9. Reconstruction Error per Character

Uppercase	Error	Lowercase	Error	Number	Error
A	3.93	a	3.55	0	2.95
B	**6.24**	b	4.14	1	*1.95*
C	2.8	c	2.68	2	3.55
D	5.22	d	4.49	3	4.01
E	3.34	e	2.79	4	3.67
F	3.61	f	4.14	5	4.07
G	5.68	g	5.15	6	3.35
H	4.25	h	3.35	7	3.2
I	*1.47*	i	*1.93*	8	4.49
J	4.47	j	3.2	9	3.97
K	4.89	k	4.15		
L	3.08	l	1.97		
M	4.26	m	4.69		
N	3.64	n	2.83		
O	2.58	o	2.69		
P	4.62	p	3.78		
Q	**6.62**	q	4.5		
R	3.53	r	2.37		
S	3.24	s	2.98		
T	3.08	t	3.48		
U	3.56	u	3.09		
V	3.18	v	2.87		
W	**6.92**	w	4.17		
X	4.38	x	3.19		
Y	3.75	y	3.37		
Z	5.04	z	3.57		

t-test results, RBM-Partition significantly outperforms Single RBM, again with substantially fewer chain operations.

Finally, RBM-Partitioned reconstruction error for each character is given in Table 9. The average reconstruction error is lowest for I, i, and 1 and it is highest for W,Q and B.

5 Conclusions and Future Work

We showed that our RBM-Partition training algorithm with small RBM partitions outperforms training full RBMs using CD-1. In addition to having superior results in terms of reconstruction error, RBM-Partition is also faster as compared to the single, full RBM. The reason that RBM-Partition is faster is due to having

fewer connections in each training step. However, the reasons for the superior generative characteristics in terms of reconstruction error is not that obvious. We hypothesize that it is because in each training step, fewer nodes are involved and a small partition RBM settles in a low energy configuration more rapidly. As we move to other stages with less partitions, fewer training instances are needed to modify the energy configuration to obtain lower energy in the full network. Furthermore, in spatial data like an image, only neighboring nodes are involved in representing a feature. Therefore, a fully connected RBM is not optimal for training spatial datasets.

Our algorithm also has similarities to transfer learning. Since in each stage we learn some weights and those weights are used as a base configuration for the next stage, in way it corresponds to feature representation transfer [13]. One interesting direction of future work is to investigate whether other methods of transfer learning can be used during training or not.

Moreover, our approach opens the door to many potential applications. Since training is done on partitioned small RBMs, we believe the method will learn multi-model data, that is data from multiple sources, more accurately. Thus, other directions for future work include: 1) carrying out additional experiments to demonstrate that this training method can be applied to other domains with a high volume of features; 2) investigating if the training layers can be used in the form of a Deep Belief Network (i.e., the process will still require partitions as we described; however, instead of training each layer independently, a layer-wise training may produce more accurate results); and 3) investigating the discriminative power of the model by running it on classification tasks.

References

1. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for boltzmann machines. Cognitive science 9(1), 147–169 (1985)
2. Bengio, Y.: Learning deep architectures for ai. Foundations and trends in Machine Learning 2(1), 1–127 (2009)
3. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al.: Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing Systems, vol. 19, p. 153 (2007)
4. Brakel, P., Dieleman, S., Schrauwen, B.: Training restricted boltzmann machines with multi-tempering: Harnessing parallelization. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part II. LNCS, vol. 7553, pp. 92–99. Springer, Heidelberg (2012)
5. Dahl, G.E., Adams, R.P., Larochelle, H.: Training restricted boltzmann machines on word observations. In: Proceedings of the 29th International Conference on Machine Learning, pp. 679–686. ACM (2012)
6. Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q.V., Mao, M.Z., Ranzato, M., Senior, A.W., Tucker, P.A., et al.: Large scale distributed deep networks. In: Advances in Neural Information Processing Systems, pp. 1232–1240 (2012)
7. Hinton, G., Salakhutdinov, R.: Discovering binary codes for documents by learning deep generative models. Topics in Cognitive Science 3(1), 74–91 (2011)

8. Geoffrey, E.: Hinton. Training products of experts by minimizing contrastive divergence. Neural Computation 14(8), 1771–1800 (2002)
9. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
10. Larochelle, H., Bengio, Y.: Classification using discriminative restricted boltzmann machines. In: Proceedings of the 25th International Conference on Machine Learning, pp. 536–543. ACM (2008)
11. Lemons, D.S.: A student's guide to entropy. Cambridge University Press (2013)
12. Louradour, J., Larochelle, H.: Classification of sets using restricted boltzmann machines, pp. 463–470. AUAI (2011)
13. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359 (2010)
14. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted boltzmann machines for collaborative filtering. In: Proceedings of the 24th International Conference on Machine Learning, pp. 791–798. ACM (2007)
15. Tieleman, T.: Training restricted boltzmann machines using approximations to the likelihood gradient. In: Proceedings of the 25th International Conference on Machine Learning, pp. 1064–1071. ACM (2008)