# Restricted Boltzmann Machine and its High-Order Extensions

Rishabh Dugar[1], Martin Renqiang Min[2], and Eric Cosatto[3]

[1]*NEC Laboratories America , EPFL, Switzerland*
[2]*Research Staff Member, NEC Laboratories America*
[3]*Senior Research Staff Member, NEC Laboratories America*

Nov 25, 2013

**Abstract**

Deep Neural Network pre-trained with Restricted Boltzmann Machine (RBM) is widely used in many applications. However, it is quite tricky to extend RBM to have high-order interactions. Its dependence on the choice of parameters and hyper-parameters such as the number of hidden units, learning rate, momentum, sampling methods, number of factors, initialization of factor weights makes it pretty difficult for a novice user to apply it to a new application. Moreover, no attempt has been made to apply high-order semi-RBM for modeling binary data, for which the RBM was introduced in the first place. In this work, we have tried to analyze the above mentioned aspects, and tried to use higher-order interactions similar to mean-covariance RBM (mcRBM) for binary data. The purpose of this work is to help someone new to RBM understand the basic RBM, subtle differences in their variations, and also appreciate the difference in performance resulting from different techniques. Experimental results on many datasets demonstrate the significance of high-order interactions for improving the generative power of RBM.

# Contents

**1 Introduction** **1**

**2 Energy-based Models and RBMs** **2**
2.1 Boltzmann Machines . . . . . 2
2.2 Restricted Boltzmann Machine . . . . . 2
2.3 A Simple Example . . . . . 4

**3 Training RBMs** **5**
3.1 Sampling Methods . . . . . 6

**4 RBMs for Continuous Data** **8**
4.1 Issues with mcRBM . . . . . 11
4.2 Even Higher-Order Correlations . . . . . 12

**5 Higher-order Correlations in Binary Data** **13**
5.1 Lateral Connections . . . . . 13
5.2 cRBM for binary data . . . . . 14

**6 Experiments and Results** **17**
6.1 Datasets . . . . . 17
6.2 Comparing Algorithms . . . . . 17
6.3 Analysis of algorithms . . . . . 18
6.3.1 CD/PCD/CD-k/PCD-k: . . . . . 19
6.3.2 Lateral Connections: . . . . . 19
6.3.3 Corrected mcRBM on Binary Data: . . . . . 19

**7 Conclusion** **21**

# 1 Introduction

Restricted Boltzmann Machine (RBM) is widely used in many applications that vary from modeling images [6], speech [4], to natural language [5] and many more. The success of RBM in efficiently modeling complex datasets as well as its application in pre-training deep networks has led to a huge interest in RBM recently, leading to a number of variants of simple RBM. RBM was initially designed to model binary distributions, but it has been extended to model distributions of continuous data as well.

In spite of the tremendous evolving and application of RBM, it is quite tricky to train it. Its dependence on the choice of parameters and hyper-parameters like the number of hidden units, learning rate, momentum, sampling methods, number of factors, initialization of factor weights makes it pretty difficult for a novice user to apply it to a new application. RBM in its most basic form has binary hidden units and is trained using Contrastive Divergence [9], but continuous hidden units and training techniques like Persistent Contrastive Divergence [8] and Fast Persistent Contrastive Divergence [7] present a variety of options for a user to train the network.

Another widely used variant of RBM for continuous data modeling, named mcRBM (mean-covariance RBM) has turned out to be very successful for modeling images [10] and speech [3]. However, no attempt has been made to apply a similar technique to improve the performance of RBM for binary data, for which the RBM was introduced in the first place.

In this work, we have tried to analyze the above mentioned aspects, and also tried to use higher-order interactions similar to mcRBM for binary data. The purpose of this work is to help someone new to RBM understand the basic RBM, subtle differences in their variations, and also appreciate the difference in performance resulting from different techniques. To validate the performance of different RBMs we have used two measures: reconstruction error from cropped (distorted) data and classification error. The first measure is classical for measuring the effectiveness of an unsupervised model, and the second one is for the most frequent use of RBMs to pre-train a multilayer neural network.

# 2 Energy-based Models and RBMs

## 2.1 Boltzmann Machines

We will first restrict ourselves to binary data and binary RBM, and later extend it to continuous RBM. The most salient feature of a RBM is the energy function defined by it. RBMs, as the name suggests, are derived from Boltzmann Machines. Boltzmann Machines belong to the category of Energy-based models, which tries to minimize the energy of the data they have to model. A Boltzmann Machine is very similar to Hopfield network, in which every configuration of the machine (values of the input units) has an energy associated with it.

$$E = -\sum_{ij} w_{ij} s_i s_j - \sum_i b_i s_i, \tag{1}$$

where $s_i$ and $s_j$ are binary states of unit $i$ and $j$. Each unit is updated, either synchronously or asynchronously, based on the weighted sum of the input it receives from all the other units. If this sum is greater than a threshold, the unit becomes active. Eventually the network will converge to a local minimum in the energy space. Boltzmann Machine is very similar to Hopfield Network, with the only exception that the weights are learned by maximum likelihood estimation and the units are turned on stochastically as opposed to deterministically in case of Hopfield network. The energy function of a Boltzmann Machine (with hidden units or without hidden units) is exactly identical to that of a Hopfield network, where some of the units can be visible units, and some can be latent hidden units.

## 2.2 Restricted Boltzmann Machine

Restricted Boltzmann machine is a variant of the Boltzmann Machine with a restriction that there are no connections between visible units. So the energy function for an RBM becomes

$$E(v, h) = -\sum_{ij} w_{ij} v_i h_j - \sum_i a_i v_i - \sum_j b_j h_j, \tag{2}$$

where $v_i$ and $h_j$ are, respectively, the binary states of visible unit $i$ and hidden unit $j$, and $a_i's$ and $b_j's$ are biases (we will use $v$ to denote visible units and $h$ to denote hidden units throughout the report). This energy function suggests that each configuration (states of the hidden units and visible units)

has an energy and consequently a probability associated with it. Obviously, the configurations with low energy are assigned high probability, and this can be expressed mathematically as

$$p(v,h) \propto -E(v,h) \tag{3}$$

$$p(v,h) = \frac{-E(v,h)}{Z} \tag{4}$$

$$Z = \sum_{v,h} e^{-E(v,h)} \tag{5}$$

where Z is called the partition function or the normalization function.

The parameter values, weights and biases in this case define a probability distribution over the visible and hidden units. It is noteworthy here that in RBM the hidden units are conditionally independent given the visible units, and the visible units are conditionally independent given the hidden units. This fact is the most salient feature of an RBM as far as the computational efficiency is concerned. Using this and equations 2-4, we can easily derive the two conditional probabilities:

$$P(h_j = 1|v) = \sigma(b_j + \sum_i v_i w_{ij}), \tag{6}$$

$$P(v_i = 1|h) = \sigma(a_i + \sum_j h_j w_{ij}), \tag{7}$$

where $\sigma(z) = \frac{1}{1+\exp(-z)}$. Thus, given the visible units we can sample the hidden units and vice versa. Sampling the visible units from the hidden units is required in the training procedure.

We can also calculate the free energy of the visible units since the hidden units used are binary, which gives us the unnormalized probability of the visible units, which in our case would be the data points.

$$F(v) = -\sum_i a_i v_i - \sum_j log(1 + e^{b_j + \sum_{i,j} w_{ij} vi}) \tag{8}$$

### 2.3 A Simple Example

We here provide a simple example to show how an RBM can define a probability distribution, by considering a simple RBM with 2 visible binary units and 1 binary hidden unit. Consider the energy function in equation 2 and assume that there are no biases. Figure 1 shows the probability of the all the 4 possible states for different values of $w_{ij}$. The probability distribution of any RBM is characterized by its weight and biases, and we can see how by changing the weights the RBM assign different probabilities to different visible states. The addition of hidden units increases the capacity of an RBM to express complex joint probability distributions, allowing the RBM to work as a product of experts model.

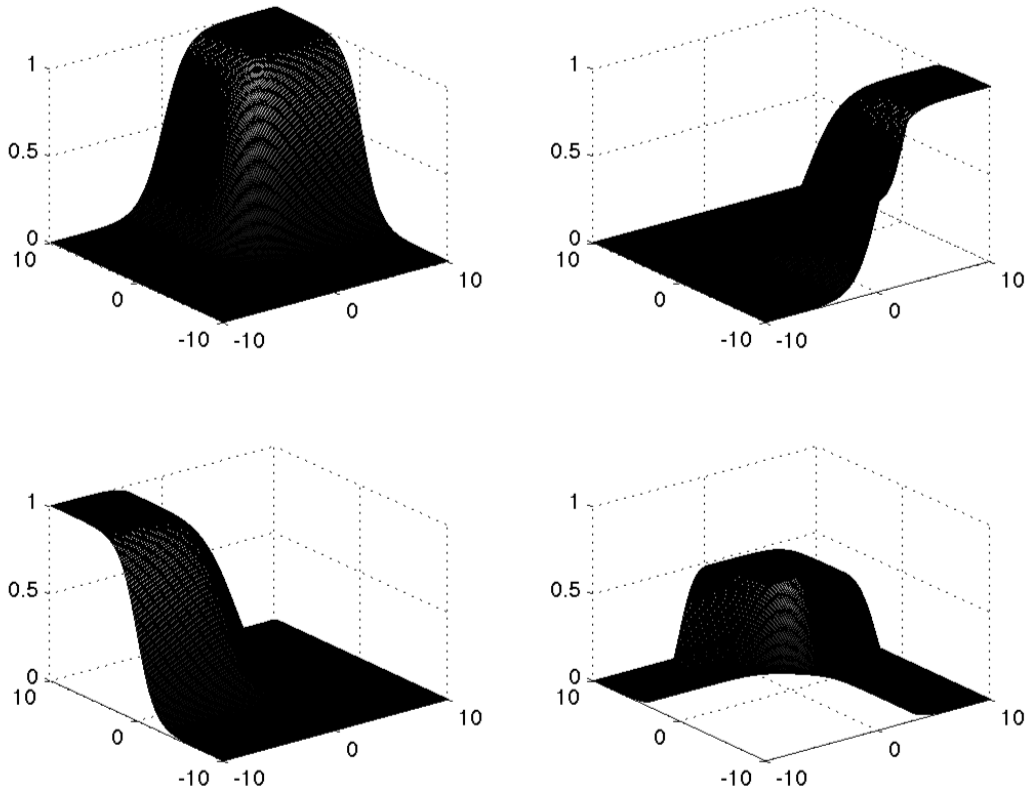



Figure 1: Clockwise from top left: The probabilities of 11, 10, 00, 01 defined by the RBM as the weights are changed. The z-axis is the probability and the x and y axes are the values of the $w_{11}$ and $w_{21}$

# 3 Training RBMs

Given that the RBM associates a probability with each configuration (data point), the straightforward way to train the network would be maximum likelihood. The parameters should be updated so that the entire set of training data points have high probability. We can use either free energy( probability of each data point exclusively) or the energy(probability of a data-point hidden unit combination) to maximize the log-likelihood of the observed dataset. We have always used the energy term in our work to update the parameters.

Given the energy term in 2 and the probabilities in 6-7, the derivative of the log probability, which will be used in gradient decent to update the weights can be written as :

$$\frac{\partial p(v)}{w_{ij}} =< v_i h_j >_{data\_point} - < v_i h_j >_{distribution} \tag{9}$$

$$\frac{\partial p(v)}{w_{ij}} = v_i p(h_j = 1/v) - < v_i h_j >_{distribution} \tag{10}$$

The corresponding derivatives for the biases would be

$$\frac{\partial p(v)}{a_i} =< v_i >_{data\_point} - < v_i >_{distribution} \tag{11}$$

$$\frac{\partial p(v)}{b_j} = p(h_j = 1/v) - < h_j >_{distribution} \tag{12}$$

With the help of equations 6 it is very easy to compute the first term of all the derivative, using the probabilities instead of the sampled binary value for the hidden units, and the observed data as the visible units. This term is also termed as positive statistics, as it tries to increase the probability of the observed data point. However the second term contains an expectation over the whole distribution defined by the RBM. This is called negative statistics, as it tried to decrease the probability of the samples generated form the current distribution. This means that we need samples (also called fantasy particles) from the exact distribution defined the RBM, and it is in the way this sample is generated that leads to different algorithms for training RBMs.

Equation 9 is very intuitive to understand, the learning tries to decrease the energy of the given data point, and increase the overall energy of all the data points defined by the RBM distribution. Eventually, the learning will stop when the RBM has updated it parameters in such a way, that the distribution defined by it is very close to the distribution of the data it is trying to model.

### 3.1 Sampling Methods

1. Gibbs Sampling:

   One way to get a sample from a joint distribution is Gibbs sampling. The idea is to start from a random state $v_0$ (visible units in our case), and using equations 6 and 7 an infinite number of time to reach the state $v_{\text{inf}}$. If this is done for a very long time, then $v_{\text{inf}}$ would be an accurate sample of the distribution. Even though theoretically robust, this procedure is not practical at all, because we need to run the chain for a very long time to get the exact sample each time we need to update the parameters.

2. Contrastive Divergence:

   This is the most commonly used procedure for sampling the negative particles. The procedure is very simple, instead of generating the exact sample from the distribution by running the chain for a long time, this procedure uses the samples generated from the first step of the chain itself. The samples $v_1$ and $h_1$ are used to compute the negative statistics. It is important to note here that the chain is always started from the data point, unlike the Gibbs Sampling when the chain could be started from any random input point. This simple strategy makes this algorithm very computationally efficient. [1] gives an intuitive as well as mathematical explanation why such an sampling would work. The idea is that the one-step reconstruction would be closer to the data distribution than the data itself (since running the chain for an infinite time leads to the data distribution), and so treating this sample as negative particle would also serve our purpose of increasing the expected energy of the samples from the distribution. From the above explanation it is clear that running the chain for n steps instead of one step would provide even better samples, as the sample become more and more close to the data distribution. The algorithm, which obtains the negative sample by running the chain for one step is called CD-1, and the one that runs the algorithm for n chains is called CD-n.

3. Persistent Contrastive Divergence:

   In case of Contrastive Divergence (CD-1 or CD-n), each time the network sees a new data point, it starts a Markov chain from that data point itself. Here, instead of doing that, in PCD the network starts a Markov chain form a random point, and maintains (persists) this chain throughout the algorithm. This means that each time, the network wants to generate negative sample to update the parameters, it runs this chain one time (or n times for PCD-n) to sample a particle. The idea is that if the learning rate of the parameters are slow enough, then eventually the samples would be accurate. In the extreme case, lets consider that the learning rate is zero, then this is exactly the same as performing the infinite chain Gibbs sampling, since the parameters do not change at all. This is theoretically more sound than Contrastive Divergence, as it generates more accurate samples and there is not much computational overhead apart from maintaining the states of the chains. A variant of PCD available in the literature is Fast PCD, in which the negative samples are obtained using the original weights of the RBM, as well as an additional set of weights which are updated using a higher (fast) learning rate. We tried using this, but it did not improve the performance of the simple RBM.

# 4 RBMs for Continuous Data

Even though RBMs were introduced to model binary data, they have been successfully used to model continuous data as well with binary hidden units. Gaussian RBM is one such model which can capture the distribution of Gaussian units. The energy for the Gaussian RBM is :

$$E(v,h) = -\sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{ij} - \sum_i \frac{(v_i - a_i)^2}{2{\sigma_i}^2} - \sum_j b_j h_j \tag{13}$$

For simplicity we assume the variance of the visible units to be 1, leading to the energy function:

$$E(v,h) = -\sum_{i,j} v_i h_j w_{ij} - \sum_i \frac{(v_i - a_i)^2}{2} - \sum_j b_j h_j \tag{14}$$

Using this energy, we can derive the activation probability of hidden units as well as the conditional probability distribution of the visible units given the hidden units in a similar way to binary units. The only difference being that the visible units can now take an infinite number of real values instead of just binary values. It turns out that the visible units are conditionally independent and Gaussian distributed themselves:

$$p(v_i|h) = \mathcal{N}(\sum_j h_j w_{ij}, 1) \tag{15}$$

On top of this we used rectified linear units for visible points to sample the visible units. Equations 13 and 14 can be used with of the sampling techniques mentioned in 3.1 to generate negative particles for training the network. One key factor to be remembered when using Gaussian RBM is that the input has to be normalized before training. This was not necessary for binary RBM, but for Gaussian RBM the data should be normalized to mean 0 and variance 1.

Gaussian RBMs are very difficult to train using binary hidden units. This is because unlike binary data, continuous valued data lie in a much larger space (for images, each unit can take 255 different value, for binary each point can only take only 2). One obvious problem with the Gaussian RBM is that given the hidden units, the visible units are assumed to be conditionally independent, meaning it tries to reconstruct the visible units

independently without using the abundant covariance information present in all datasets. The knowledge of the covariance information reduces the complexity of the input space where the visible units could lie, thereby helping RBMs to model the distribution.[1] tried to gate the interaction between the visible units, leading to the energy function:

$$E(v,h) = \frac{1}{2}\sum_{i,j,k} v_i v_j h_k w_{ijk} - \sum_i a_i v_i - \sum_k b_k h_k \tag{16}$$

To understand the role of gated hidden units, let us consider the example of images. In case of images nearby pixels are always highly correlated, but presence of an edge or occlusion would make these pixels different. It is this flexibility that the above network is able to achieve, leading to multiple covariances of the dataset. Every state of the hidden units defines a covariance matrix. This type of RBMs are called Covariance RBM (cRBM).

To take advantage of both the Gaussian RBM (which provides the mean) and the cRBM, mcRBM uses an energy function that includes both the term:

$$\begin{aligned} E(v,h^g,h^m) &= \frac{1}{2}\sum_{i,j,k} v_i v_j {h_k}^g w_{ijk} - \sum_i a_i v_i - \sum_k b_k {h_k}^g \\ &\quad - \sum_{ij} v_i {h_j}^m w_{ij} - \sum_k c_k {h_k}^m \end{aligned} \tag{17}$$

In equations 16 and 17, each hidden unit modulate the interaction between each pair of pixels leading to a large number of parameters in $w_{ijk}$ to be tuned. However, most of the real world data for structured and do not need such explicit modulation between each pair of visible units. To reduce this complexity, [2] introduced factors approach to approximate the weight $w_{ijk}$.

$$w_{ijk} = \sum_f C_{if} C_{jf} P_{kf} \tag{18}$$

The energy function can now be written as

$$E(v,h^g,h^m) = \frac{1}{2}\sum_f \Big(\sum_i v_i C_{if}\Big)^2 \Big(\sum_k h_k w_{kf}\Big) - \sum_i a_i v_i - \sum_k b_k {h_k}^g$$

$$-\sum_{ij} v_i {h_j}^m w_{ij} - \sum_k c_k {h_k}^m \tag{19}$$

Using this energy function, we can again derive the activation probabilities of the hidden units, as well the respective gradients for training the network. Figure 2 explains the structure of this factored mcRBM, the hidden units on the left are called mean hidden units and those on the right are called covariance hidden units.

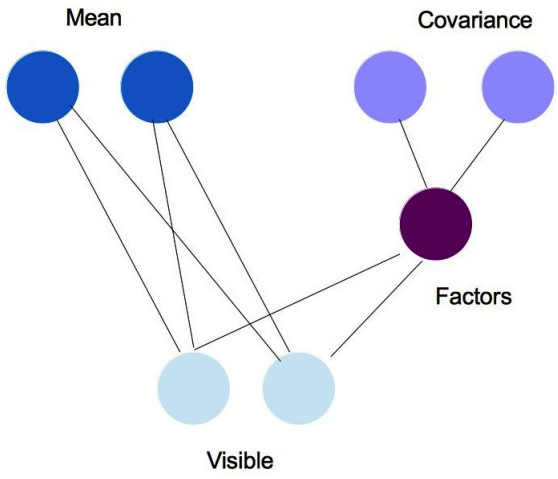



Figure 2: Structure of factored mcRBM with 2 mean hidden and 2 covariance hidden units

The energy function can also be used to sample the negative particles given the hidden units, but this requires computing the inverse of a matrix which is computationally very expensive for each training update. To get over this problem, [10] defines a sampling method called Hybrid Monte Carlo sampling to generate the negative particles. The is that given a starting point $P_0$ and an energy function, the sampler starts at $P_0$ and moves with randomly chosen velocity along the opposite direction of gradient of the energy function to reach a point $P_n$ with low energy. This is similar to the concept of CD (or PCD), where an attempt is make to reach as close as possible to the actual model distribution. The term n is specified by the leap-frog steps, which we chose to be 20. [10]provides the details of the exact algorithm.

Since we want to sample a visible point, we need the free energy of the samples instead of the joint energy of the samples and hidden units. The free energy can be easily computed for binary hidden units can be obtained in a similar way to equation 8

### 4.1 Issues with mcRBM

Training mcRBMs is very tricky, because they depend a lot on initialization of the factor weights(P and C), their learning rates and normalization of P and C.

- P is initialized and constrained to be positive. According to equation 19, if any value of P is allowed to be negative, the HMC can obtain extreme negative or positive value for the negative particles, since they would have very low energy(close to inf). To understand this, we can think of a concave quadratic function and try to sample a point with low energy from it. This is indeed the reality with mcRBM and so it is very important to satisfy this constraint.

- The biases of the covariance hidden units are all assigned positive values, which makes the units to be ON most of the time. The only way to make a hidden unit off is when the factors connected to the hidden units provide large input. This is multiplied with the negative value of P, and can turn OFF the hidden unit. This can be thought of a constraint gating, where the violation of a constraint leads to turning off the hidden units.

- Both the P and C matrix are normalized to have unit norm along their columns. Along with this, the data is also normalized along its length. The normalization of data and C leads to the model being invariant to the magnitude of the input point, rather it only depends on the cosine of the angle between the input and P filters. Normalization of P does not influence the performance, and we have not used it. This normalization of the input data, changes the energy function which has to be taken care of while computing its gradient during HMC.

- Learning rate for C is assumed to be very low. This is required because by empirical evaluation we found that a comparable learning rate to P, leads to instability in the performance.

- The input data is preprocessed using PCA whitening to remove the noise present in the data. Whitening helps to get rid of the strong pairwise correlations in the data, which are not much informative like correlation between adjacent pixels in an image. This step also reduces

the dimensionality of the data points, thereby helping the algorithm computationally. It is a crucial step, because working on the raw data leads to the network modeling noise more than the important features.

- The P matrix is often initialized using a topographical mapping, which leads to pooling of the factors. This means that nearby factors (in a topographical sense) capture similar features. To understand topographical mapping, we can think of $n^2$ hidden units arranged on $nxn$ grid at layer 1, and similarly the $m^2$ factors arranged on a $mxm$ grid at layer 0. Each hidden unit is now only connected to its closest few factors in the lower layer.

These are some of the precautions and initialization tricks that have to be taken care of while using an mcRBM. With these the mcRBM can detect interesting second-order correlations present in the data, leading to better modeling of data.

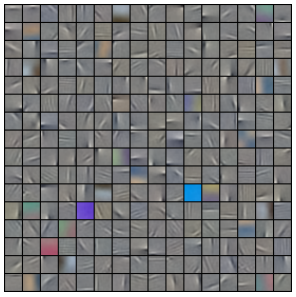
### 4.2 Even Higher-Order Correlations

The mcRBMs successfully captures second-order correlations in the data. If the data would be purely Gaussian, the highest correlation present would be second-order. But real world data are not purely Gaussian, so we can also look for even higher order correlations in a similar procedure. To capture third order correlations, we modify the energy function in equation 19 as follows:

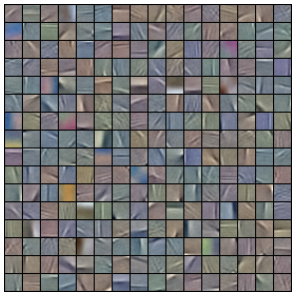
$$\begin{aligned} E(v, h^{g3}, h^m) &= \frac{1}{3}\sum_f (abs(\sum_i v_i C_{if}))^3 (\sum_k {h_k}^{g3} w_{kf}) - \sum_i a_i v_i - \sum_k b_k {h_k}^{g3} \\ &\quad - \sum_{ij} v_i {h_j}^m w_{ij} - \sum_k c_k {h_k}^m \end{aligned} \tag{20}$$

We take the absolute value of the C filter outputs. If this was not used, there was no way we could constraint P to ensure that the HMC does not give extreme values as negative particles. The absolute value has to be considered when computing the gradient during HMC. Figure 3 shows the filters learned by third order-interactions.

The natural thing would be now to combine the second order and third order terms. Figure 3 shows the filters obtained for the second order mcRBM, third order mcRBM and their combination when they are trained on patches of colored images. It can be observed that the presence of explicit second-order capturing filters leaves the third order filters to capture third order interactions only, which is very less in natural images. Again we can see the

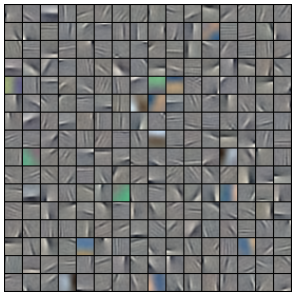


(a) Only second order

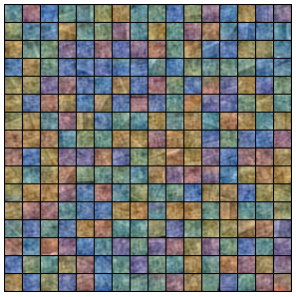


(b) Only third order

Figure 3: C filters obtained when second order and third order mcRBM are used independently, Equation 19 and 20



(a) Second order filters



(b) Third order filters

Figure 4: C filters obtained when second order and third order mcRBM are used together.

presence of third order filters enables the second order ones to model only second order interactions, making the filters more sharp. The third-order filters are still able to detect some colored edges (not clear in the figure). With different datasets, hopefully even the third order ones would be able to capture significant third-order correlations.

# 5 Higher-order Correlations in Binary Data

## 5.1 Lateral Connections

One extension of a simple RBM is to introduce lateral connections between the visible units to capture second order interactions in the data. The idea is that if two units are highly correlated, then this correlation can be captured by the lateral weights, and the RBM weights can capture more interesting features than strong pairwise correlations. [14] The energy function of such a Boltzmann machine, also termed as Semi-Restricted Boltzmann Machines with lateral connection can be written as:

$$E(v,h) = -\sum_{i,k} w_{ik} v_i h_k - \sum_i a_i v_i - \sum_k b_k h_k - \sum_{ij} v_i v_j L_{ij} \tag{21}$$

where L defines the lateral weights between the visible units constrained with $L_{ii} = 0$

In such a network, the hidden units are still conditionally independent given the visible units, but the visible units are no longer conditionally independent. So, the visible units cannot be sampled according to equation 7, and we need to apply mean-field reconstruction of the visible units. This is a computational disadvantage on the regular RBM, where the visible units can be sampled in parallel. Using the energy function above ,we can derive the following:

$$\alpha_i(n) = \sigma(a_i + \sum_k h_k w_{ik} + \sum_j v_i(n-1) L_{ij}) \tag{22}$$

$$v_i(n) = \lambda v_i(n-1) + (1-\lambda)\alpha_i(n) \tag{23}$$

where $\lambda$ is the parameter for mean-filed reconstruction. We chose this parameter to be 0.2, however, the choice of lambda did not seems to have a great impact on the performance. and used the above equation from $n = 1$ to $n = 10$, assigned $v(0)$ to be either the data (CD) or the persistent fantasy particle (PCD)

The $L_{ij}$ weight vector indeed captures the true covariance matrix (approximately) of the dataset. There is also a difference in the quality of filters obtained as a result of lateral connections, whereby the filters appear to extract more interesting features than filters without lateral connections which are active for a small blob of input space (like when modeling images). However, if the hidden units are high enough then all the good features are captured without lateral connections as well.

### 5.2 cRBM for binary data

Lateral connections is one way to capture higher order correlations in binary data. However, it provides only one correlation matrix for the dataset. The idea of cRBM, described above was first introduced for binary data [2], claiming that it can capture better features than a simple RBM. Unfortunately, there has been no possible attempt to use this modification in improving the performance of the binary RBM. The energy function of equation 16 can be used for binary data.

This is indeed a better model for binary data, as it allows hidden units to gate the interactions between visible units. It is to be noted here that the visible units are no longer independent given the hidden units, so we cannot use Gibbs sampling to compute all the visible unit activations simultaneously. Instead we need to perform Gibbs sampling sequentially for all the dimensions of the visible units. This can be very computationally expensive for inputs with relatively high dimension (even 500). To avoid this, we use an approximations in the form of mean field updates to get the samples. This is not as accurate as exact Gibbs sampling but fulfills our purpose most of the time. We used mean field in a similar way to the one used in lateral connections with $\lambda$ 0.2, and performed 10 mean field updates for each sampling. Similar to GRBM, we tried to extend this cRBM to mcRBM for binary data, leading to the energy functions 24 (which is same as 17)

$$E(v, h^g, h^m) = \frac{1}{2} \sum_{i,j,k} v_i v_j {h_k}^g w_{ijk} - \sum_{ij} v_i {h_j}^m w_{ij} + bias\_terms \tag{24}$$

This contains both the terms for modeling the mean as well as the covariance of the dataset. However, this does not perform better than a simple RBM in practice. This is because the expansion of the left terms in equation 24 contains terms like ${v_i}^2 {h_k}^g w_{iik}$, which are also present in the mean side $v_i {h_k}^m w_i k$, because ${v_i}^2 = v_i$ for binary data. This leads to some kind of competition between the mean and covariance hidden units depending on their learning rates, leading to instability in learning. Similarly, it can be observed that equation 16 contains terms for both mean and covariance information, but in this case a hidden unit is made to model both this information simultaneously. To get rid of these issues, we introduced a slight modification in the energy function, to make sure that mean and covariance hidden units do not overlap with each other. We call this corrected mcRBM, with the following energy function expressed in terms of the factors:

$$\begin{aligned} E(v, h^g, h^m) &= -\frac{1}{2} \sum_f ((\sum_i v_i C_{if})^2 - \sum_i {v_i}^2 Cif^2)(\sum_k h_k w_{kf}) \\ &\quad - \sum_{ij} v_i {h_j}^m w_{ij} + bias\_terms \end{aligned} \tag{25}$$

This is exactly same to ensuring that in , $w_{iik} = 0 \ \forall i$ . This ensures that the two hidden units model the mean and covariance information re-

spectively. Remaining part of the algorithm , HMC sampling and derivatives, are adjusted accordingly. This gives better performance than normal mcRBM as well as ordinary RBM.

It is to be noted here that this modification was relatively simple while capturing second order correlations, for higher order correlations, it would not be as straightforward.

One reason why this idea of capturing high order correlation has not been investigated for binary data, is that mean RBM perform exceptionally well for RBM. In fact, it has been showed mean RBM can capture any kind of probability distribution, given that there are sufficient hidden units.[**?**]. But we still see that sometimes, capturing the second order and even higher order information explicitly helps the performance of RBM, and perhaps using this information in a Deep Belief Network might be beneficial even for binary data.

# 6 Experiments and Results

## 6.1 Datasets

We used four datasets to validate the previous works on RBM, various variations on RBM, as well as our proposed method to capture higher order interactions in binary and continuous data

**MNIST dataset**: This is a dataset of 70000 hand written digits belonging to 10 class, and is the most popular dataset used for binary classification. Each input image is of size $28 \times 28$ leading to an input dimension of 784. The training set consists of 60000 points, equally divided in the 10 classes. The data is gray-scale pixel values and is divided uniformly by 255 to get values between 0 and 1. Although the data is not exactly binary, the distribution of each dimension is peaked at 0 and 1 making it suitable to use for binary purposes.

**USPS dataset**: This is another hand written digit image dataset of size $16X16$. There are 11000 data points belonging to 10 classes uniformly, which we divided into training and testing in the ratio $8:3$. The dataset is normalized in the same manner as MNIST. One motivation to use this dataset was the low dimensionality of 256.

**The ENCODE Transcription Factor binding dataset**: This is a dataset of 11400 protein-coding genes, each containing the binding activities of 116 Transcription Factors (TFs). This is a purely binary dataset, and we chose this dataset as it has some high order correlations. This is an unsupervised dataset with no class labels.

**Newsgroup 20 by date**: This is a dataset for document classification, containing 11200 training documents, and 7500 testing points. Each documents consists of many words, and we chose the 5000 most frequent words in all the documents combined to form a binary bag of word representation for the documents, consisting of 5000 features.

## 6.2 Comparing Algorithms

The main motivation of this work has been to compare various aspects (parameters, learning rates, sampling procedures) of RBMs , and come up

with one that works the best. Given that this is an unsupervised algorithm, we could not directly use classification percentages to compare them. So we used two techniques:

- **Reconstruction**
  A good quality which is looked after in any unsupervised algorithm is its ability to regenerate original data from corrupt data. To accomplish this for binary data, we randomly assigned some bits of the input data to 0 or 1, and tried to recover the original data. For images, we cropped the test data by some rows of pixels from the bottom of the images and for the ENCODE data we assigned some of the TFs (randomly chosen) to be absent/present. Then we used the RBM equations to regenerate the original data by running multiple steps of Gibbs sampling for hidden and visible units, each time clamping the uncorrupted part of the data (rows which are not removed, and TFs which are not altered) to the corresponding visible units. We chose the number of Gibbs step to be 500. For reconstruction, we did not search for the number of hidden units that give the best reconstruction, rather we fixed the total number of hidden units to be same for all the algorithms.

- **Classification** (RBM for pre-training)
  RBMs are widely used as pre-training a Deep Belief network. We did not try to go deep, rather just built a logistic layer on top of a pre-trained RBM to classify the data. A better model of the data, would give better features and consequently lead to lower classification. The learning rate used for 0.1 and no momentum was used. Out goal was not to be better than the state of the art algorithms, rather the goal was to find out whether different parameters of RBM influence the learning or not. Unlike reconstruction, in classification we did model selection to find the best number of hidden units.

  Apart from these techniques we also visualize the filters obtained after training (mean or covariance) to measure the effectiveness of the algorithm, particularly for image datasets (MNIST and USPS).

### 6.3 Analysis of algorithms

For all the datasets and algorithms, we used mini-batch of 20 data points for training.

### 6.3.1 CD/PCD/CD-k/PCD-k:

For a basic RBM, this is a crucial decision as it defines the sampling method needed for generating the negative statistics. k defines the number of Markov steps for both CD and PCD, and does not play a part for continuous data, as sampling is done by HMC. Tables 1,2,3 shows the reconstruction errors for the two binary image datasets and Table 4 shows the classification performance. This shows that PCD performs slightly better than CD, and increasing the value of k indeed improves the performance. It is to be noted that PCD demands a low learning rate for reasons explained before, and the computational expense of PCD is same as CD. However, introducing 'k' has a significant influence on the computation as can be seen from the Table 1. So, the general conclusion is to work with k=1 and use PCD instead of CD,however with an abundance of computational resource PCD-k outperforms PCD. Learning rates used were 0.01, no momentum and training was done for 50 epoch with annealing after 20 epochs.

### 6.3.2 Lateral Connections:

We cannot use classification to measure the effectiveness or utility of lateral connections, because even though [14] claims that introducing lateral connections leads to some redundant features being ignored, but we found that an RBM with sufficient hidden units can still capture all the discriminating features. To see whether the covariance information helps or not, we use reconstruction errors. Table 1,2 shows the reconstruction error for USPS and the ENCODE datasets. The lateral connections did not seem to help much as there are sufficient hidden units. It can be concluded that if the number of hidden units are less, the lateral connections help a lot but once the number of hidden units is sufficient, they do not help that much. Table 5 shows the reconstruction error on USPS data for PCD and Lateral connections as the number of hidden units are increased.

### 6.3.3 Corrected mcRBM on Binary Data:

We tried to compare the performance of cRBM, mcRBM and corrected mcRBM on MNIST, USPS and the ENCODE dataset using classification (for the first two) and reconstruction (Table 4). We used learning rate of 0.01 for mean weights, 0.01 for C matrix (for documents we used learning of 0.001 for mean and C as well) and 0.001 for P matrix. The P and C filters were initialized randomly , and we applied column-normalization for P matrix. The number of epcoh used for training were 50, and we annealed the learning rate after 20 epochs. Table 10 gives a comparative view of the classification percentage and Table 6,7,8 shows the reconstruction errors. For reconstruction, we keep the total number of hidden units along different

algorithms to be same to get unbiased comparisons. The reconstruction error is better for the corrected mcRBM most of the times particularly when there is more cropping (or distortion), showing that indeed some good features are captured which help the mean features for modeling data. For classification, we used the best case number of hidden units, and observed not much improvement. However, mcRBM is very computationally expensive as compared to binary RBM, as can be seen from Table 6.

| MNIST(the number of rows cropped) | CD | CD-10 | PCD | PCD-10 | Lateral |
|---|---|---|---|---|---|
| 16 | 17.87 | 17.35 | 17.34 | 17.00 | 17.51 |
| 14 | 14.13 | 13.83 | 13.89 | 13.85 | 14.23 |
| 12 | 11.24 | 11.00 | 10.86 | 10.73 | 11.20 |
| 10 | 8.99 | 8.65 | 8.77 | 8.68 | 8.88 |
| 8 | 6.34 | 6.14 | 6.14 | 6.00 | 6.35 |
| Time taken (sec/epoch) | 17.1 | 88.2 | 23.1 | 101.1 | 95.0 |

Table 1: Reconstruction performance for MNIST under various algorithms, each with 300 hidden units and different cropping size. Timing analysis is for 500 hidden units and mini-batch of 100.

| USPS(the number of rows cropped) | CD | CD-10 | PCD | PCD-10 | Lateral |
|---|---|---|---|---|---|
| 8 | 25.39 | 24.88 | 25.64 | 24.78 | 24.83 |
| 7 | 24.45 | 24.02 | 24.67 | 23.62 | 24.60 |
| 6 | 23.18 | 22.89 | 22.89 | 22.20 | 23.21 |
| 5 | 21.92 | 21.01 | 21.68 | 20.81 | 22.03 |
| 4 | 20.00 | 19.35 | 20.35 | 18.78 | 20.13 |

Table 2: Reconstruction performance for USPS under various algorithms, each with 300 hidden units and different cropping size

# 7 Conclusion

In this work, we tried to analyze the different varieties of RBMs existing in the literature. The objective is to provide someone relatively new to RBM a basic guide about some of the advanced applications and modifications of RBM, both in binary and continuous domain. Along with this, an attempt was made to apply mcRBM type energy function to binary data to capture the covariance information in binary data. Although, the results do show some improvement in the performance of basic RBMs, but it comes at a very high computational cost. The simple binary RBM is very strong in itself in modeling data, so the use of mcRBM to model data may not be an attractive option, but if the task is collaborative filtering or reconstruction distorted data, it does extract better features than a basic RBM. As far as continuous data is concerned, the 2-mcRBM performs very well and the experiments provide hope for even 3 and more higher order mcRBM.

| ENCODE(the number of TFs corrupted %) | CD | CD-10 | PCD | PCD-10 |
|---|---|---|---|---|
| 50 | 20.35 | 19.98 | 20.72 | 20.01 |
| 33 | 20.11 | 19.45 | 20.07 | 18.85 |
| 25 | 21.32 | 21.00 | 21.45 | 20.87 |
| 20 | 17.05 | 16.85 | 17.88 | 17.45 |
| 10 | 21.89 | 21.30 | 21.78 | 21.42 |

Table 3: Reconstruction performance for ENCODE dataset under various algorithms, each with 300 hidden units and different cropping size

| Classification | CD | CD-10 | PCD | PCD-10 | Lateral |
|---|---|---|---|---|---|
| MNIST | 1.52 | 1.50 | 1.43 | 1.43 | 1.50 |
| USPS | 2.1 | 2.1 | 2.2 | 2.0 | 2.1 |

Table 4: Classification under various algorithms(MNIST-500 hidden units, USPS-300 hidden units) and different cropping size

| Number of hidden units | PCD-1 | Lateral |
|---|---|---|
| 50 | 25.00 | 22.32 |
| 100 | 22.10 | 19.35 |
| 200 | 18.23 | 18.45 |
| 300 | 17.34 | 18.20 |
| 400 | 17.24 | 17.89 |

Table 5: Reconstruction of MNIST data, (16 rows cropped) under PCD and with lateral connections

| MNIST(the number of rows cropped) | RBM(PCD-1) | cRBM | mcRBM | corrected mcRBM |
|---|---|---|---|---|
| 16 | 17.00 | 20.1 | 18.23 | **15.45** |
| 14 | 13.95 | 16.35 | 14.22 | **12.31** |
| 12 | 10.73 | 12.05 | 11.84 | **10.45** |
| 10 | 8.68 | 10.67 | 9.67 | **8.13** |
| 8 | **6.34** | 8.23 | 8.10 | 6.95 |
| Time taken (sec/epoch) | 23.1 | 122.3 | 134.4 | 142.5 |

Table 6: Reconstruction error performance for MNIST under various high order RBM: RBM (300 mean), cRBM(300 covariance), mcRBM (200 mean + 100 cov), corrected mcRBM(200 mean +100 cov).
Timing analysis of for 500 hidden units( total), and minibatch of 100

| USPS(the number of rows cropped) | RBM(PCD-1) | cRBM | mcRBM | corrected mcRBM |
|---|---|---|---|---|
| 8 | 24.78 | 26.76 | 25.25 | **22.38** |
| 7 | 23.62 | 25.34 | 23.89 | **22.10** |
| 6 | 22.20 | 24.33 | 22.89 | **21.75** |
| 5 | **20.81** | 22.10 | 21.78 | 21.12 |
| 4 | **18.78** | 21.07 | 19.83 | 19.00 |

Table 7: Reconstruction performance for USPS under various high order RBM: RBM (300 mean), cRBM(300 covariance), mcRBM (200 mean + 100 cov), corrected mcRBM(200 mean +100 cov)

| ENCODE(the number of TFs corrupted %) | RBM(PCD-1) | cRBM | mcRBM | corrected mcRBM |
|---|---|---|---|---|
| 50 | 20.72 | 22.32 | 20.35 | **17.35** |
| 33 | 20.07 | 22.21 | 20.14 | **17.77** |
| 25 | 21.45 | 22.76 | 21.78 | **19.38** |
| 20 | 17.88 | 20.10 | 17.23 | **15.68** |
| 10 | 21.78 | 23.12 | 21.02 | **19.23** |

Table 8: Reconstruction performance for the ENCODE dataset under various high order RBM: RBM (300 mean), cRBM(300 covariance), mcRBM (200 mean + 100 cov), corrected mcRBM(200 mean +100 cov)

| Document(the number of words corrupted %) | RBM(PCD-1) | corrected mcRBM |
|---|---|---|
| 50 | 14.38 | **2.33** |
| 33 | 11.32 | **2.13** |
| 25 | 7.96 | **2.04** |
| 20 | 7.08 | **2.13** |
| 10 | 4.12 | **1.80** |

Table 9: Reconstruction performance for News Group dataset under high order RBMs: RBM (700 mean) and corrected mcRBM (500 mean + 200 covariance)

| Classification | RBM(PCD-1) | cRBM | mcRBM | corrected mcRBM |
|---|---|---|---|---|
| MNIST | 1.43 | 1.50 | 1.50 | 1.46 |
| USPS | 2.1 | 2.0 | 2.1 | 1.7 |

Table 10: Classification under various algorithms and different cropping size:RBM (500 mean), cRBM(500 covariance), mcRBM (300 mean + 300 cov), corrected mcRBM(300 mean +300 cov)

## References


[1] Geoffrey E. Hinton, "Learning to represent visual input" *Philosophical Transactions of the Royal Society B: Biological Sciences* 365(1537):177–184 Jan 12, 2010

[2] Geoffrey E. Hinton: A Practical Guide to Training Restricted Boltzmann Machines. *Neural Networks: Tricks of the Trade (2nd ed.)* 2012: 599-619

[3] Marc'Aurelio Ranzato, Geoffrey E. Hinton: Modeling pixel means and covariances using factorized third-order boltzmann machines. *CVPR* 2010: 2551-2558

[4] Abdel-rahman Mohamed, Geoffrey E. Hinton: Phone recognition using Restricted Boltzmann Machines. *ICASSP* 2010: 4354-4357

[5] Salakhutdinov, R. R. and Hinton, G. E. (2009). Replicated softmax: An undirected topic model. *In Advances in Neural Information Processing Systems*,2009 volume 22.

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

[7] Tijmen Tieleman, Geoffrey E. Hinton: Using fast weights to improve persistent contrastive divergence. *ICML* 2009: 130

[8] Tijmen Tieleman: Training restricted Boltzmann machines using approximations to the likelihood gradient. *ICML* 2008: 1064-1071

[9] Geoffrey E. Hinton: Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation* 14(8): 1771-1800 (2002)

[10] George E. Dahl, Marc'Aurelio Ranzato, Abdel-rahman Mohamed, Geoffrey E. Hinton: Phone Recognition with the Mean-Covariance Restricted Boltzmann Machine. *NIPS* 2010: 469-477

[11] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences of the USA*, vol. 79 no. 8 pp. 25542558, April 1982.

[12] Hinton, G. E.; Sejnowski, T. J. (1986). "Learning and Relearning in Boltzmann Machines". In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Volume 1: Foundations (Cambridge: MIT Press): 282317.

[13] A. Krizhevsky. Learning multiple layers of features from tiny images, 2009. MSc Thesis, Dept. of Comp. Science, Univ. of Toronto.

[14] Simon Osindero, Geoffrey E. Hinton: Modeling image patches with a directed hierarchy of Markov random fields. *NIPS* 2007

[15] Hugo Larochelle, Yoshua Bengio: Classification using discriminative restricted Boltzmann machines. *ICML* 2008: 536-543