Study of Restricted Boltzmann Machines and Their Application in Classification

Saurabh Sihag

December 17, 2016

1 Introduction

Restricted Boltzmann Machines (RBMs) are a type of undirected graphical models which follow Markov property, and hence belong to the family of Markov Random Fields. RBMs derive their structure from Boltzmann machines, which consist of two layers of nodes: visible nodes and hidden nodes, and allow visible-hidden, visible-visible and hidden-hidden connections among the nodes. Calculation of likelihood for a boltzmann machine is computationally expensive, hence, certain restrictions are put on its structure, which leads to the formation of RBMs.

Graphical models are generally used to learn the unknown probability distribution followed by a set of samples(also called training data), and then make inference such as prediction, classification and learning parameters. RBMs have been used to form generative models for different applications, as mentioned in [1]. [2] provides a detailed guide for training RBMs and it has been used extensively in the practical implementation in this project. In this project, it is intended to achieve the following objectives:
a) study the structure of RBMs
b) study the necessity of Contrastive Divergence algorithm and its implementation
c) application of RBMs in classification

This report is divided into various sections as follows: Section 2 briefly discusses the theory of markov random fields and their properties inherited by RBMs, Section 3 discusses the Contrastive Divergence algorithm, Section 4 discusses the application of RBMs for classification, Section 5 discusses the implementation and classification results over MNIST dataset, and Section 6 discusses the conclusions and observations made during this project.

2 Markov Random Fields and RBMs

A Markov Random Field (MRF) is an undirected graph that models the probability distribution of a set of random variables $X = \{X_i\}$ in agreement with the Markov property. The global markov property is described below [3]

Let the graphical model be represented by $G = \{V, E\}$, where V represents the vertices, and E represents the edges of the graph G. Each edge represents the dependence between its two vertices. The markov blanket of a node v is defined as the set of neighbours of the node v, and is represented as $MB(v)$. For an MRF, $P(v \mid V) = P(v \mid MB(v))$; which means that a node v is independent of all other nodes in the graph, given its markov blanket $MB(v)$. According to Hammersley Clifford theorem, the probability distribution over an MRF follows the Gibbs distribution.

Theorem 1 (*Hammersley Clifford Theorem*)**.** *A probability distribution p follows the markov property over an undirected graph G iff it can be factorized as a product of potential functions of the maximal cliques of the graph G.*

$$p(X) = \frac{1}{Z} \prod_{c \epsilon C} \phi(X_c)$$

where c is a maximal clique,
C is the set of all maximal cliques for graph G;
X_c is the set of nodes in clique c;
Z is the partition function to normalize $\sum_X p(X)$ to 1.

$$Z = \sum_X \prod_{c \epsilon C} \phi(X_c)$$

and ϕ is the potential function.

In general, ϕ can be any non negative function. $p(X)$ can also be written in the following form:

$$p(X) = \frac{1}{Z} \prod_{c \epsilon C} \phi(X_c) = \frac{1}{Z} e^{\sum_{c \epsilon C} log(\phi(X_c)} = \frac{1}{Z} e^{-E(X)} \ldots\ldots\ldots\ldots (1)$$

where $E(X)$ is called the energy function. The probability distribution that can be expressed as in the equation (1) is called Gibbs distribution.

An RBM is an MRF with two layers: hidden layer and visible layer. Each node in the hidden layer is connected to all the visible nodes and vice-versa, however, there are no hidden-hidden or visible-visible connections, unlike Boltzmann machines. The nodes in the RBM can be represented as $X = \{V, H\}$, where $V = \{v_i\}_{i=1}^{m}$ forms the set of visible nodes, and $H = \{h_i\}_{i=1}^{n}$ forms the set of hidden nodes. Figure 1 shows one possible structure of RBM.

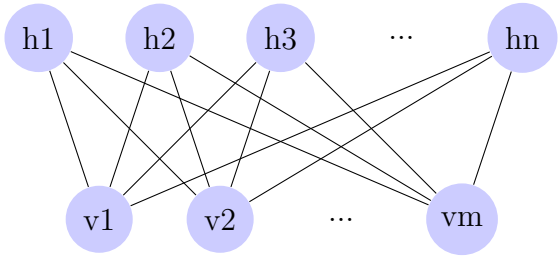

Figure 1: Restricted Boltzmann Machine

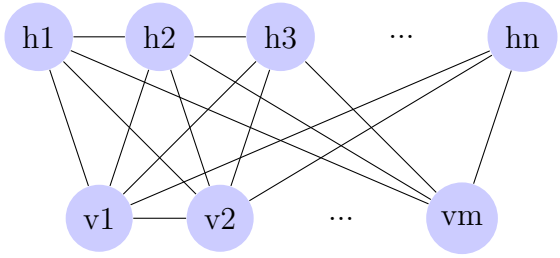

Figure 2: Boltzmann Machine

In this project, binary RBMs are used, which implies that $v_i \epsilon \{0, 1\}$ and $h_i \epsilon \{0, 1\}$. The joint probability distribution for RBM can be written as:

$$p(V, H) = \frac{1}{Z} e^{-E(V,H)} \ldots\ldots\ldots\ldots (2)$$

where,

$$E(V, H) = -\sum_{i=1}^{n} b_i h_i - \sum_{j=1}^{m} c_j v_j - \sum_{i=1}^{n} \sum_{j=1}^{m} h_i w_{ij} v_j \ldots\ldots\ldots\ldots (3)$$

b_i and c_j are the bias terms associated with i^{th} hidden node h_i and j^{th}visible node v_j respectively, and w_{ij} is the weight associated with the edge between h_i and v_j.

The parameters of the RBM are learnt using maximum likelihood estimation. The log-likelihood of the parameters

$$\Theta = \{w_{ij}, b_i, c_j\}, i = 1, 2, \ldots n, j = 1, 2, \ldots, m$$

can be written as:

$$\begin{aligned} LL(\Theta) &= log(\prod_{l=1}^{L} \frac{1}{Z} p(V^l \mid \Theta)), \text{where L is the number of samples} \\ &= \sum_{l=1}^{L} log(\sum_{H} \frac{1}{Z} p(V^l, H \mid \Theta)) \ldots\ldots\ldots\ldots (4) \end{aligned}$$

Therefore, the log-likelihood for each sample can be written as:

$$\begin{aligned} LL_l(\Theta) &= log(\sum_{H} \frac{1}{Z} p(V^l, H \mid \Theta)) \\ &= log(\sum_{H} e^{-E(V^l,H)}) - log(\sum_{V,H} e^{-E(V,H)}) \ldots\ldots\ldots\ldots (5) \end{aligned}$$

The likelihood in the above equations can be maximized using gradient ascent. Therefore,

$$\begin{aligned} \frac{\partial LL_l(\Theta)}{\partial \Theta} &= \frac{\partial}{\partial \Theta}(log(\sum_{H} e^{-E(V^l,H)})) - \frac{\partial}{\partial \Theta}(log(\sum_{V,H} e^{-E(V,H)})) \\ &= -\frac{1}{\sum\limits_{H} e^{-E(V^l,H)}} \sum_{H} e^{-E(V^l,H)} \frac{\partial E(V^l, H)}{\partial \Theta} + \frac{1}{\sum\limits_{V,H} e^{-E(V,H)}} \sum_{V,H} e^{-E(V,H)} \frac{\partial E(V, H)}{\partial \Theta} \\ &= -\sum_{H} p(H \mid V^l) \frac{\partial E(V^l, H)}{\partial \Theta} + \sum_{V,H} p(H, V) \frac{\partial E(V, H)}{\partial \Theta} \ldots\ldots\ldots\ldots (6) \end{aligned}$$

$$\text{because } \frac{e^{-E(V^l,H)}}{\sum\limits_{H} e^{-E(V,H)}} = \frac{\frac{e^{-E(V^l,H)}}{Z}}{\frac{\sum\limits_{H} e^{-E(V,H)}}{Z}} = \frac{p(V^l, H)}{p(V^l)} = p(H \mid V^l)$$

$$\text{and } \frac{e^{-E(V,H)}}{\sum\limits_{V,H} e^{-E(V,H)}} = \frac{\frac{e^{-E(V,H)}}{Z}}{\frac{\sum\limits_{V,H} e^{-E(V,H)}}{Z}} = \frac{p(V, H)}{Z} = p(V, H)$$

Computing the term $\sum\limits_{V,H} p(H, V) \frac{\partial E(V,H)}{\partial \Theta}$ is difficult because it involves all possible combinations of H and V, which has exponential computational complexity.

The gradient of $LL_l(\Theta)$ can also be written in the following form:

$$\frac{\partial LL_l(\Theta)}{\partial \Theta} = -E_{p(H|V^l)}(\frac{\partial E(V^l, H)}{\partial \Theta}) + E_{p(V,H)}(\frac{\partial E(V, H)}{\partial \Theta}) \dots\dots\dots\dots (7)$$

The terms in equation (7) are hard to compute, and hence are approximated using MCMC sampling methods like Gibbs Sampling. This leads to the contrastive divergence algorithm for training the parameters of RBM, which will be discussed in the next section.
From (3)

$$\frac{\partial E(V, H)}{\partial w_{ij}} = -h_i v_j, \frac{\partial E(V, H)}{\partial b_i} = -h_i \text{ and } \frac{\partial E(V, H)}{\partial c_j} = -v_j \dots\dots\dots (8)$$

Using (7), equation (6) can be written as:

$$\begin{aligned}
\frac{\partial LL_l}{\partial w_{ij}} &= \sum_H p(H \mid V^l) h_i v_j^l - \sum_V p(V) \sum_H p(H \mid V) h_i v_j \\
&= p(h_i = 1 \mid V^l) v_j^l - \sum_V p(V) p(h_i = 1 \mid V) v_j \dots\dots\dots (9)(i) \\
\frac{\partial LL_l}{\partial b_i} &= \sum_H p(H \mid V^l) h_i - \sum_V p(V) \sum_H p(H \mid V) h_i \\
&= p(h_i = 1 \mid V^l) - \sum_V p(V) p(h_i = 1 \mid V) \dots\dots\dots\dots (9)(ii) \\
\frac{\partial LL_l}{\partial c_j} &= \sum_H p(H \mid V^l) v_j^l - \sum_V p(V) \sum_H p(H \mid V) v_j \\
&= v_j^l - \sum_V p(V) v_j \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (9)(iii)
\end{aligned}$$

The conditional probability of a variable being one can be written in the following form:

$$p(H_i = 1 \mid V) = \sigma(\sum_{j=1}^{m} w_{ij} v_j + b_i)$$

$$p(V_j = 1 \mid H) = \sigma(\sum_{i=1}^{n} w_{ij} h_i + c_j)$$

$$\text{where } \sigma(x) = \frac{1}{1 + e^{-x}}$$

From (4) and (5), it is implied that

$$LL(\Theta) = \sum_{l=1}^{L} LL_l(\Theta)$$

The gradient of $LL(\theta)$ can be written in the following way:

$$\frac{\partial LL}{\partial \Theta} = -\left\langle \frac{\partial E(V,H)}{\partial \Theta} \right\rangle_{p(h^{samples}|v^{samples})} + \left\langle \frac{\partial E(V,H)}{\partial \Theta} \right\rangle_{p(h^{model},v^{model})} \dots\dots\dots (10)$$

where $\langle\rangle_p$ is the average of the gradient calculated according to the samples generated by p. Therefore, the first term is driven by the probabilistic model based on the training data samples, while the second term is driven by the true probabilistic model. Since calculation of the second term is not feasible, it is approximated using Gibbs Sampling. Using the set of equations (9),the gradient of complete log-likelihood $LL(\Theta)$ can be written as:

$$\frac{\partial LL}{\partial w_{ij}} = \langle h_i v_j \rangle_{samples} - \langle h_i v_j \rangle_{model} \dots\dots\dots\dots\dots (11)(i)$$
$$\frac{\partial LL}{\partial b_i} = \langle h_i \rangle_{samples} - \langle h_i \rangle_{model} \dots\dots\dots\dots\dots (11)(ii)$$
$$\frac{\partial LL}{\partial c_j} = \langle v_j \rangle_{samples} - \langle v_j \rangle_{model} \dots\dots\dots\dots\dots (11)(iii)$$

Here, h_i and v_j are the samples from model distribution, which are estimated by gibbs sampling in the CD algorithm. The parameters $\{w_{ij}, b_i, c_j\}$ are updated using gradient ascent:

$$w_{ij} = w_{ij} + \eta \frac{\partial LL}{\partial w_{ij}}$$
$$b_i = b_i + \eta \frac{\partial LL}{\partial b_i}$$
$$c_j = c_j + \eta \frac{\partial LL}{\partial c_j}$$

3 RBM Learning using Contrastive Divergence

As discussed in the previous section, the exact value of the gradient of the log likelihood of an RBM is difficult to compute because of the dependence of one of its terms on the true probabilistic model. This term can be estimated using MCMC methods like Gibbs Sampling. Contrastive Divergence is based on this premise and is a standard parameter learning algorithm for RBMs [4] [5].

Under Contrastive Divergence, the second term of the log-likelihood gradient in equation (10) is estimated using a sample from a Markov chain that is initialized by the training data. Therefore, the initial stage of the Markov chain is a sample from the training data, represented by V^0, and after k steps, yields the sample V^k. If k is large enough, this sample can be considered as a sample from the true or stationary distribution of the RBM. However, in practice, k is usually set to 1. Each step k consists of sampling the hidden variables H^k from $p(H \mid V^k)$, followed by sampling V^{k+1} from $p(V \mid H^k)$.

Algorithm 1 provides a pseudo code for a k-step contrastive divergence algorithm.

Algorithm 1 k-step Contrastive Divergence

$V \leftarrow$ visible nodes
$H \leftarrow$ hidden nodes
$RBM \leftarrow \{V_1, V_2, .., V_m, H_1, H_2, .., H_n\}$
$W = \{w_{ij}\} \leftarrow$ weight corresponding to edge between H_i and V_j
$b_i \leftarrow$ bias term for H_i
$c_j \leftarrow$ bias term for V_j
$S \leftarrow$ training samples
$lr \leftarrow$ learning rate
$w_{ij} = 0, b_i = 0, c_j = 0 \ \forall i, j$
$V^0 = S$
for `t = 0 to k-1` **do**
    $H^t \sim$ sample from $p(H \mid V^t)$
    $V^{t+1} \sim$ sample from $p(V \mid H^t)$
for `i = 1 to n and j = 1 to m` **do**
    $w_{ij} = w_{ij} + lr(p(H_i = 1 \mid V^0)V_j^0 - p(H_i = 1 \mid V^k)V_j^k)$
    $b_i = b_i + lr(p(H_i = 1 \mid V^0) - p(H_i = 1 \mid V^k))$
    $c_j = c_j + lr(V_j^0 - V_j^k)$

It is important to note that if k is not large enough, V^k is not a sample from the model distribution, and hence the Contrastive Divergence algorithm is biased. It does not aim at maximizing the likelihood, but minimizing the difference between two KL divergences [4] [3]:

$$KL(q \mid\mid p) - KL(p^k \mid\mid p)$$

where q is the distribution obtained from training samples, p^k is the distribution obtained after k samples of Markov chain and p is the model distribution. As k becomes very large, p^k approaches p and hence, the error introduced by the approximation using Gibbs sampling vanishes.

4 Using RBMs for Classification

RBMs form the building block for deep hierarchical models, which have been used for classification in [1] and [6]. The Deep networks are formed by stacking up layers of RBMs and training them layer by layer in the first step, followed by fine tuning of weights using supervised learning.

Learning the parameters for classification in deep models consists of two steps:
1. Unsupervised pre-training: Learning the parameters by gradient ascent based maximum likelihood estimation (Contrastive Divergence)
2. Fine tuning: The target variables are added as an additional layer on top of the hidden layer, and the parameters are fine tuned with weights from pre-training step as initialization (supervised training)

In this project, both these steps are implemented for classification using RBM. After the pre-training stage, the label variables are added to the network as shown in figure 4. Let $L = \{l_1, l_2, \ldots l_p\}$ be a set of label variables for a model with p classes, with $l_i = 1$ for class i, and rest of variables 0. Therefore, $L = \{0, 1, 0, 0, 0, 0, 0, 0, 0, 0\}$ is a label variable for a sample belonging to class 2 in a model with 10 classes. Adding an additional layer of label variables on the top of an RBM is equivalent to setting the visible nodes as $\{V, L\}$, where V takes the input from the training data, and L takes input from the corresponding training labels. Thus, the networks in figure 3 and figure 4 are equivalent.

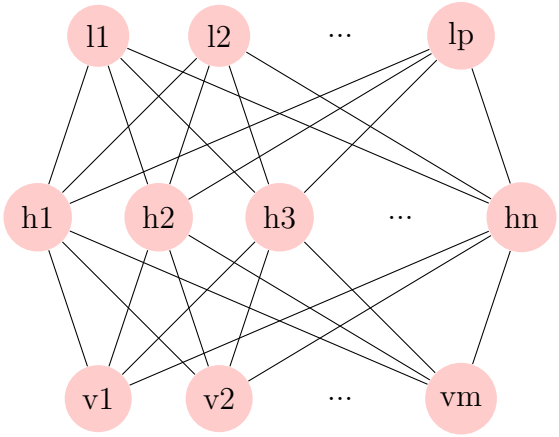

Figure 3: Network Structure with label variables

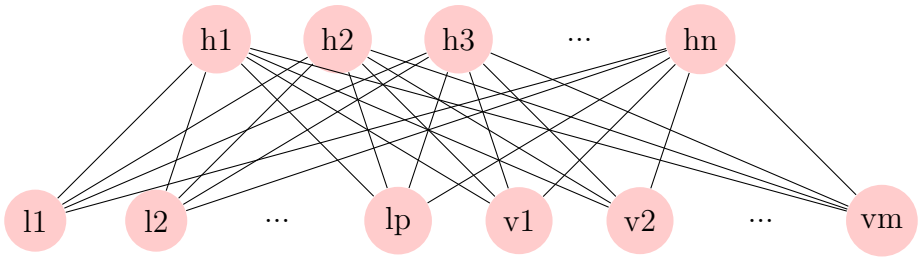

Figure 4: RBM with Equivalent Network Structure to the one in Figure 4

The random variables in the RBM in figure 4 are: Visible Layer: $\{l_1, l_2, ...l_p, v_1, v_2, ..., v_m\}$ and Hidden Layer: $\{h_1, h_2, ..., h_p\}$. Therefore, the parameters of this RBM can be initialized as:

$$w_{ij} = \begin{cases} 0, & \forall j \epsilon \{1, 2, ..p\} \\ \text{initialized from pre-training step }, & \forall j \epsilon \{p+1, p+2, ..p+m\} \end{cases}$$

$$c_j = \begin{cases} 0, & \forall\ j \epsilon\ \{1, 2, ..p\} \\ \text{initialized from pre-training step }, & \forall\ j\ \epsilon \{p+1, p+2, ..p+m\} \end{cases}$$

b_i initialized from pre-training step $\forall\ i\ \epsilon \{1, 2, ..n\}$

The log-likelihood of the parameters of this RBM is maximized using maximum likelihood estimation with the help of contrastive divergence. The log-likelihood of this RBM is:

$$LL(\Theta) = log(\prod_{s=1}^{S} \frac{1}{Z} P(V^s, L^s \mid \Theta)), \text{where S is the number of samples}$$

The RBM in figure 4 represents a joint model of labels and training samples, and its parameters are learnt such that the joint distribution $P(V, L \mid \Theta)$ is maximized. For testing this network on the test dataset, the energy function (given in (3)) is computed for all possible classes for every test sample, and the class which gives the lowest value for the energy function is chosen as the classification result for that test sample.

5 Implementation and Results

Previous sections discuss the theory behind training the RBMs and fine tuning the parameters to maximize the joint likelihood of training data and labels. In this section, some important points for implementation are discussed, which have been mentioned in [2].

The RBMs used in this project are binary. When the hidden units are sampled from the visible units, they are assigned binary values, however, in the last update, the hidden units are assigned the probability value itself to avoid sampling noise. Therefore, in CD_k, the last update of hidden units uses the probability. The visible units could be updated as stochastic binary units or as real valued probabilities, as mentioned in [2].
The parameters of RBM can be updated by using the one training sample at a time. However, this process is very tedious, especially for large number of training samples, as is the case with MNIST dataset, which has 60000 training samples. Hence, the training samples are divided into batches of size 100, and the gradient for each batch is divided by the size of the batch. If the size of the batch is too large, it may have a negative effect on the rate of learning as the update in weights would be too small.

Description of the Dataset

MNIST dataset consists of grayscale images of handwritten digits, that are of equal dimensions, and the digits are size normalized and centered inside the image.[] The images are grouped into 10 classes: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ where the class name also represents the digit associated with it. The training set consists of 60000 images and the test dataset consists of 10000 images. The dimesion of each image is 28×28.

Results

- Figure 5 and Figure 6 show the visualization of the weights associated with the edges between hidden units and visible units after CD-1. This visualization is one form of representation of the features learnt by the hidden units. Each square represents the weights associated with one hidden unit.

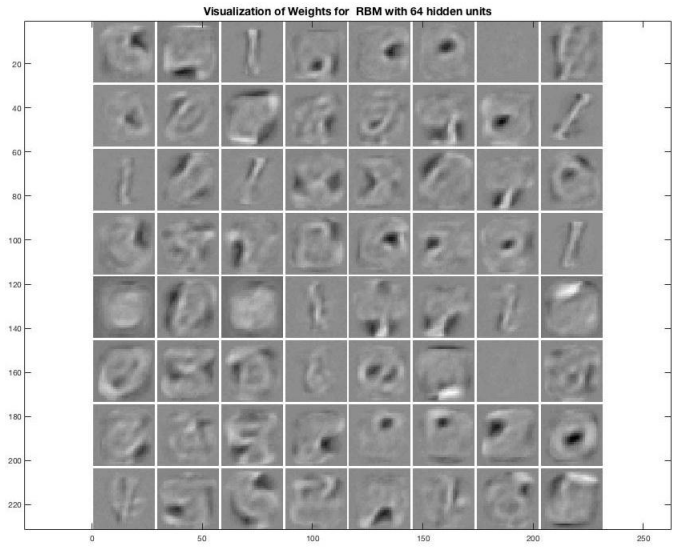

Figure 5: Visualization of weights for RBM with 64 hidden units

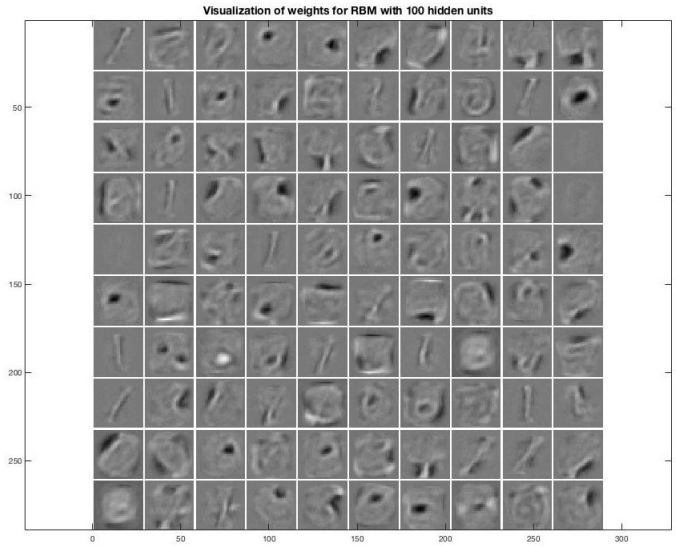

Figure 6: Visualization of weights for RBM with 100 hidden units

It can be observed that there are some hidden units in Figure 4 which do not appear to be utilized. If the number of hidden units is made larger, it could lead to overfitting over training data, or a significant number of hidden units that are not used.

- Figure 7 shows the plot of reconstruction error versus the number of iterations for RBMs with different number of hidden units.

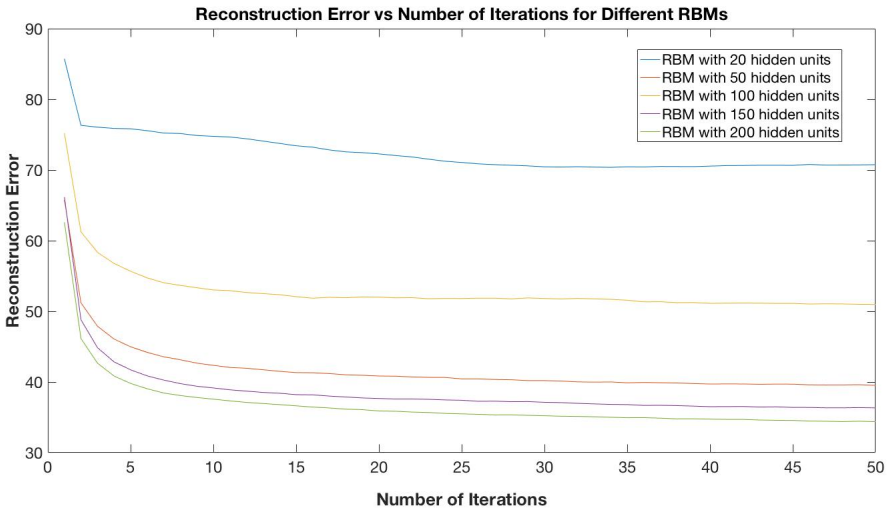

Figure 7: Reconstruction Error vs Number of Iterations for different RBMs

It can be concluded from the above figure that an RBM with more number of hidden units is able to model the training dataset better than the ones with lower number of hidden units. However, reconstruction error can't be used as a reliable metric for the classification performance because small reconstruction error may imply overfitting.

- Figure 8 shows the plot of classification performance vs the number of hidden units in the RBM. It reiterates that more number of hidden units in one layer may not mean improvement in classification performance.

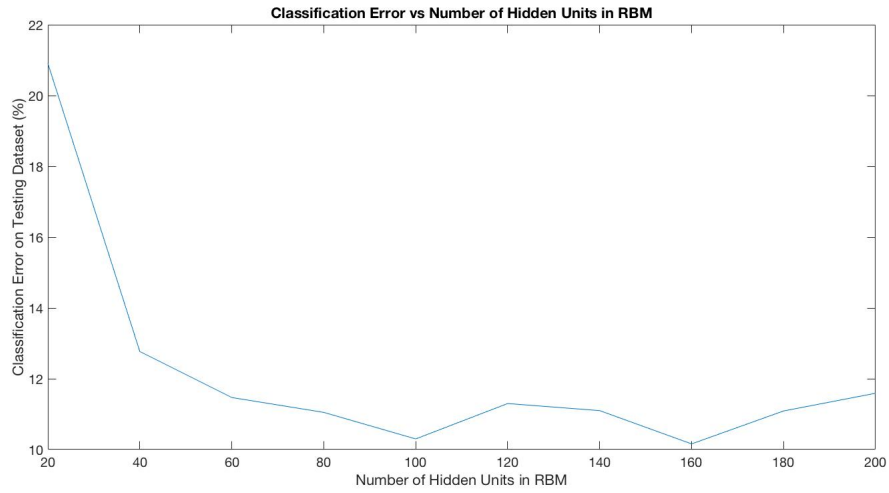

Figure 8: Classification Error vs Number of Hidden units in the RBM

Table 1: Classification Error for different classes for different RBMs

Number of Hidden Units in RBM	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	Average Error
20	14.5	5.1	18.5	27.3	30.3	28.0	12.4	16.3	31.3	25.4	20.91
40	5.8	3.0	14.9	16.0	11.8	16.7	6.8	12.4	17.8	22.5	12.77
60	4.1	2.5	12.7	17.7	11.8	18.9	7.2	11.1	11.9	16.8	11.47
80	3.7	2.3	13.1	17.0	13.1	13.7	6.8	11.8	13.7	15.3	11.05
100	3.9	5.5	11.6	15.0	12.0	13.0	5.2	9.3	12.8	14.7	10.30
120	4.3	7.3	12.6	14.9	11.0	19.9	5.0	11.9	15.1	11.0	11.30
140	3.1	7.9	12.4	18.0	10.5	12.7	7.3	16.5	11.5	11.1	11.10
160	2.5	5.8	12.4	11.1	10.0	13.9	7.3	12.1	11.7	14.8	10.16
180	4.2	8.5	14.2	12.8	8.5	13.2	6.2	19.5	11.7	12.1	11.09
200	2.9	4.7	16.8	14.5	14.1	16.2	7.9	15.2	10.1	13.5	11.59

Table 1 lists the classification error for each class (columns C0 to C9) for different number of hidden units in the RBM.

6 Conclusions

- In this project, contrastive divergence algorithm is implemented to learn the parameters of an RBM. RBM is a generative model, that models its parameters according to the data provided. This can be seen in Figures 4 and 5, where the weights seem to learn the features from the training dataset.
- Learning the parameters of RBM using maximum likelihood estimation is tricky because gradient of the log-likelihood depends on the samples from the stationary or model distribution. CD algorithm uses Gibbs sampling to generate samples that are used to approximate the gradient of log-likelihood. 1-step CD algorithm is used in this project as additional number of sampling steps in the algorithm did not seem to provide any major improvement in the reconstruction error or the classification performance after fine tuning.
- Supervised Learning is performed by adding the label variables in the RBM as shown in figures 3 and 4. The joint density function of label variables and training samples is maximized using maximum likelihood estimation. The classification is performed by selecting the class that gives the minimum energy function for a test sample.
- It is also observed that increasing the number of hidden units helps RBM to model the training data better, but it does not improve the classification performance over the test dataset beyond a certain point. This could be because more number of hidden units leads to overfitting on the training data, but no increase in the learning ability of the network.

- RBMs are the building blocks for deep models such as deep boltzmann machines and deep belief networks. The contrastive divergence training algorithm for RBMs, which has been reviewed and implemented in this project, is also utilized for training deep models.

References

[1] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[2] G. Hinton, "A practical guide to training restricted boltzmann machines," *Momentum*, vol. 9, no. 1, p. 926, 2010.

[3] A. Fischer and C. Igel, "An introduction to restricted boltzmann machines," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2012, pp. 14–36.

[4] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[5] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[6] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines."