

Restricted Boltzmann Machines

Jason Liang, Julian Sia

Abstract—In this report, we describe an undirected graphical model called Restricted Boltzmann Machine (RBM) that is capable of learning complex distributions. We analyze learning procedures and applications of RBMs and show experimental results for training RBMs on a standard image dataset.

I. INTRODUCTION

Restricted Boltzmann Machines (RBMs) are commonly used in classification, dimensionality reduction, and representation learning tasks. They are a type of undirected graphical model consisting of two sets of nodes, a set of hidden units and a set of visible units. Given the graph $G = (V, E)$ that describes the RBM, $\forall v_{vis} \in V_{vis} \subset V, \exists e \in E$, such that $e = (v_{vis}, v_{hid})$, $\forall v_{hid} \in V_{hid} \subset V$. RBMs are a variant of the Boltzmann Machine model [1] posed by Hinton and Sejnowski, with the restriction that there are neither edges connecting any visible units nor edges connecting any hidden units [2]. This restriction guarantees that the visible units are conditionally independent of other visible units; and hidden units, conditionally independent of hidden units. Furthermore, it makes determining the gradient of the free energy equation, which will be described in more detail later, tractable. The structure can be visually elucidated with the representation below. In Fig. 1, we show an simple example RBM with a hidden layer of 3 units and a visible layer of 6 units.

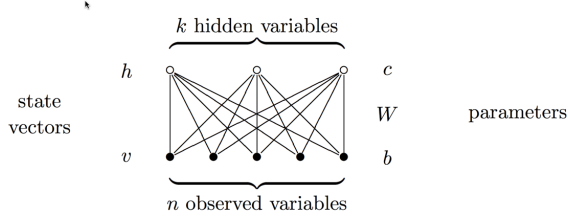


Fig. 1. A graphical representation of a RBM illustrated by Cueto et al. [3]. Note that there are no connections in between the visible units h and hidden units v .

II. PREVIOUS WORK

RBMs were created as a variant of Boltzmann Machines after practical difficulties were encountered in training Boltzmann Machines due to the fact that Boltzmann Machines are fully connected graphical models. Boltzmann Machine themselves were motivated primarily by limitations of Hopfield Networks [4]: their poor storage capacity and tendency to converge at local energy minima.

In the early days of Boltzmann machines, Hinton and Sejnowski calculated the gradient of the log likelihood by “[fixing] a training vector on the visible units, initializing the

hidden units to random binary states, and using sequential Gibbs sampling of the hidden units” [1], using simulated annealing to speed up the convergence process. Other attempts were made by Neal with persistent Markov chains [5] and Peterson and Anderson with mean-field methods (instead of Gibbs Sampling) to speed up the learning process [6]. After RBMs were posed as a simplifying restriction by Smolensky, Hinton discovered a way to speed up the learning process, coined contrastive divergence (CD) which is described in this report [7], [2], [8].

III. ALGORITHM DESCRIPTION

Given a set of training vectors, V , to train a RBM, one aims to maximize the average probability, $p(v)$, $v \in V$, where

$$p(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}, \quad Z = \sum_{v,h} e^{-E(v,h)} \quad (1)$$

This is the same as trying to minimize the negative log probability of $p(v)$. One can accomplish this by taking the derivative of the log probability of $p(v)$ with respect to the weights defined in the energy function, $E(v, h)$ defined as

$$\begin{aligned} E(v, h) &= - \sum_{i \in \text{visible}} a_i v_k - \sum_{j \in \text{hidden}} b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j \\ &= -b'v - c'h - h'Wv \end{aligned} \quad (2)$$

where W is a weight matrix and c and b are bias vectors corresponding to the hidden and visible nodes respectively. Differentiating with respect to $w_{i,j}$, we find that the partial derivative reduces to the following:

$$-\frac{\partial \log p(v)}{\partial w_{i,j}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \quad (3)$$

With this reduction the following learning rule with ϵ learning rate can be derived

$$\Delta w_{i,j} = \epsilon (\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}) \quad (4)$$

Alternatively, using free energy notation, the partial derivative can be defined as

$$-\frac{\partial \log p(v)}{\partial \Theta} = \frac{\partial \mathcal{F}(v)}{\partial \Theta} - \sum_{\tilde{v}} p(\tilde{v}) \frac{\partial \mathcal{F}(\tilde{v})}{\partial \Theta} \quad (5)$$

where we define free energy as

$$\mathcal{F}(v) = -\log \sum_h e^{-E(v,h)} \quad (6)$$

Here, Θ represents the model parameters for the RBM which include the weights W , hidden node biases b and visible node

biases c . Because the computation of the second term in (5) is intractable, we can approximate the second term using a fixed number of samples \tilde{v} from the model:

$$-\frac{\partial \log p(v)}{\partial \Theta} = \frac{\partial \mathcal{F}(v)}{\partial \Theta} - \frac{1}{|\mathcal{N}|} \sum_{\tilde{v} \in \mathcal{N}} p(\tilde{v}) \frac{\partial \mathcal{F}(\tilde{v})}{\partial \Theta} \quad (7)$$

The free energy formula for RBMs in general is the following:

$$\mathcal{F}(v) = -b'v - \sum_i \log \sum_{h_i} e^{h_i(c_i + W_i v)} \quad (8)$$

For modeling binary images and to simplify calculations, one can use a binary RBM. In the case of a binary RBM, where each node can only take on a value of zero or one, the free energy function reduces to the following:

$$\mathcal{F}(v) = -b'v - \sum_i \log(1 + e^{c_i + W_i v}) \quad (9)$$

Using the free energy function defined in (9) and the derivative defined in (7), we can define the following update equations for the RBM's negative log likelihood gradient:

$$\begin{aligned} -\frac{\partial p(v)}{\partial W_{i,j}} &= E_v[p(h_i|v) \cdot v_j] - v_j^{(i)} \cdot \text{sigm}(W_i \cdot v^{(i)} + c_i) \\ -\frac{\partial p(v)}{\partial c_i} &= E_v[p(h_i|v)] - v_j^{(i)} \cdot \text{sigm}(W_i \cdot v^{(i)}) \\ -\frac{\partial p(v)}{\partial b_j} &= E_v[p(v_j|h)] - v_j^{(i)} \end{aligned} \quad (10)$$

To obtain samples from $p(v)$ and compute the gradient, one can treat the RBM as a Markov chain and perform Gibbs sampling by alternating between fixing the hidden nodes, sampling the visible nodes and vice-versa. We note the conditional independence assumptions for the hidden and visible layers:

$$\begin{aligned} p(h|v) &= \prod_i p(h_i|v) \\ p(v|h) &= \prod_j p(v_j|h) \end{aligned} \quad (11)$$

and also the conditional distributions for each visible and hidden node:

$$\begin{aligned} p(h_i = 1|v) &= \text{sigm}(c_i + W_i v) \\ p(v_j = 1|h) &= \text{sigm}(b_j + W'_j h) \end{aligned} \quad (12)$$

where $\text{sigm}(x)$ is the logistic sigmoid function.

As a consequence of (10) and (11), we can perform block Gibbs Sampling with the following update rules:

$$\begin{aligned} h^{(n+1)} &\sim \text{sigm}(W'v^{(n)} + c) \\ v^{(n+1)} &\sim \text{sigm}(Wh^{(n+1)} + b) \end{aligned} \quad (13)$$

As $t \rightarrow \infty$, $(v^{(t)}, h^{(t)})$, v^t and h^t will be accurate samples drawn from the RBM's distribution. Nevertheless, one can further speed up the process by using the contrastive divergence (CD) algorithm, which computes an approximation

of the gradient by performing Gibbs sampling for a finite number of steps. This involves initializing the Markov chain with a training example $v \in V$ and running the Markov chain for k (often with $k = 1$) steps. CD makes the following simplifications for computing the gradient:

- 1) Replace the first term (expectation over all input samples) with a single sample.
- 2) For the second term, run the Markov chain for fixed k steps.

We will describe CD-1 below, which runs the Markov Chain for only 1 step.

- 1) Take a training sample v , compute the probabilities of the hidden units and sample a hidden activation vector h from this probability distribution.
- 2) Compute the outer product of v and h and call this the positive gradient.
- 3) From h , sample a reconstruction v' of the visible units, then resample the hidden activations h' from this. (Gibbs sampling step)
- 4) Compute the outer product of v' and h' and call this the negative gradient.
- 5) Let the weight update to $w_{i,j}$ be the positive gradient minus the negative gradient, times some learning rate:

$$\Delta w_{i,j} = \epsilon(vh^T - v'h'^T) \quad (14)$$

The bias updates for the visible and hidden layers respectively can be defined by these expressions:

$$\begin{aligned} \Delta v_b &= \epsilon(v - v') \\ \Delta h_b &= \epsilon(h - h') \end{aligned} \quad (15)$$

CD- k is a generalization of CD-1 by repeating the sampling process in step three k times, though in practice it has been found that 1 iteration (CD-1) works fairly well in "[ensuring] that hidden features retain most of the information in the data vector" [8]. One extension of contrastive divergence is persistent contrastive divergence (PCD). When given a new input sample for training, PCD computes the second term (the negative gradient) by continuing Gibbs sampling from the stopping position of the Markov chain of the previous input instead of rerunning from scratch [9]. Experimental results show that it allows more accurate samples to be drawn from the RBM's current distribution.

IV. APPLICATIONS OF RBMS

RBMs has been show to be capable of representation learning, also known as manifold learning. Most real world data are high dimensional, but they tend to lie close to a low dimensional manifold that is embedded in the high dimensional space. What RBMs are essentially doing is learning a mapping that transforms the original input data (high dimensional vectors) into coordinates along that manifold (low dimensional vectors). After an RBM has been trained, it can be reinterpreted as a deterministic encoder, a single layer neural network that performs dimensionality reduction. Other dimensionality reduction techniques such as PCA can only learn a

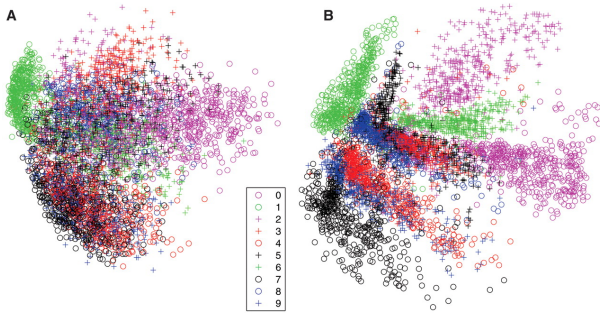


Fig. 2. Comparison of A) PCA and B) RBM in their ability to perform dimensionality reduction on the MNIST dataset [10].

linear manifold or hyperplane. However, RBMs with sufficient number of hidden units are capable of learning any arbitrary manifold shapes. As shown in Fig. 2, RBMs are much more effective than PCA in performing dimensionality reduction on an example dataset such as MNIST [11]. The different classes of digits are more visually distinct and separable when using RBMs instead of PCA to transform the digits.

A popular application of RBMs that exploits their representation learning capabilities is stacking them to construct deep neural networks. After a RBM has been trained, the samples h from the hidden layer can be used as input to train a second RBM. This can be extended to an arbitrary number of RBMs. After training multiple RBMs, they can be converted into a deep neural network by treating each RBM layer deterministically. A deep neural network of such variety can then be fine tuned with common supervised training methods such as backpropagation [10] and used for tasks like image classification. In 2006, Hinton used stacked RBMs to train a deep neural network that achieved an error rate of 1.39% and outperformed a SVM with RBF kernel, the state of the art at the time [2].

V. EXPERIMENTAL RESULTS

We implement our own RBM in Python with the NumPy library [12]. We train a RBM with 784 visible units and 500 hidden units on the MNIST handwritten digit dataset [11], a standard benchmark for image classification. The learning algorithm used is CD-1, which is the simplest of the CD methods and tends to work very well in practice. We initialize the weights of the RBM randomly by sampling them from a normal distribution with zero mean and 0.01 standard deviation. Next, we apply CD-1 to all 60,000 training images for 15 epochs with a learning rate 0.005.

In Fig. 3, we visualize the trained weights of the RBM. Each hidden node is connected to all 784 visible nodes and thus we can display the weights corresponding to a hidden node as an image with the same dimension as the input digits. We see that the filters learned are interesting, with a diverse variety of different edge and blob detectors.

Next in Fig. 4, we show the samples generated by the RBM when given test digits as input. The first column shows the original digits while the subsequent columns shows samples

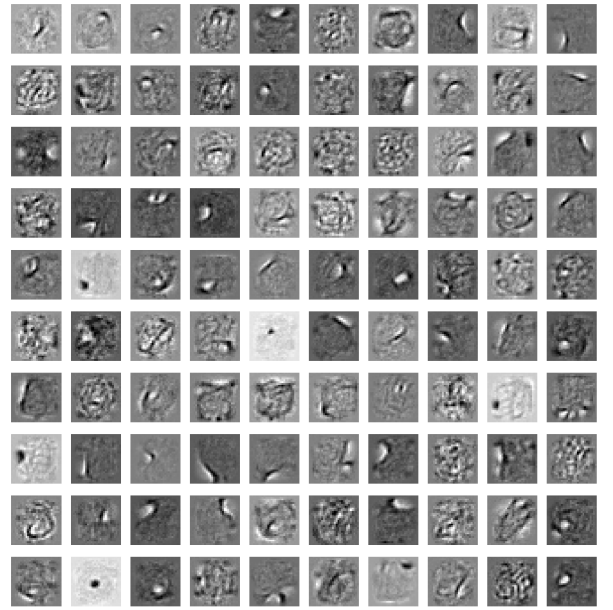


Fig. 3. Filters corresponding the weights of the first 100 hidden nodes.

outputted by the model after an interval of 1000 steps of Gibbs sampling (to ensure proper mixing of the Markov chain). The images in Fig. 4 show that RBM has properly learned the distribution for input digits and that samples are drawn from that distribution properly.

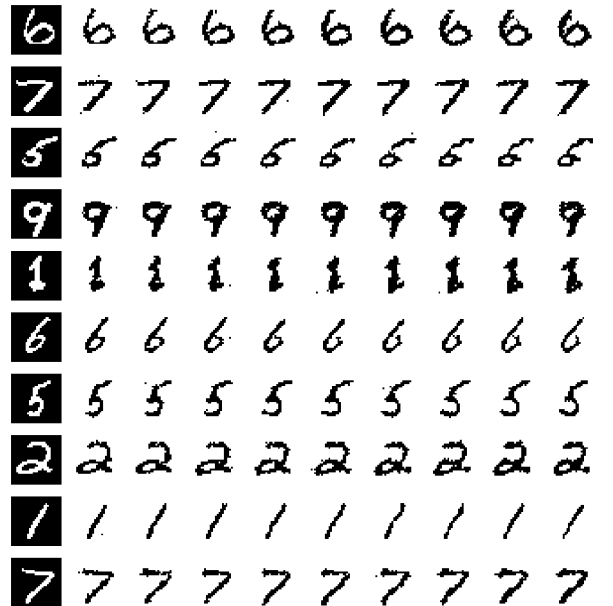


Fig. 4. Samples generated by RBM. The first column shows original test digit sample. Subsequent columns show digits sampled from RBM with an interval of 1000 steps of Gibbs sampling.

Lastly, we explore the usefulness of RBMs in learning good

representations of the input data and performing dimensionality reduction. As a preprocessing step prior to classification, we treat our trained RBM as a deterministic single layer neural network and use it to transform the input digits into a 500 dimensional feature vector. This feature vector is then used for supervised training of a logistic regression classifier. Without an RBM layer, the misclassification rate on a test set of 10,000 digits is 8.20%. With the addition of the RBM layer, the error rate of the logistic regression classifier decreases to 3.07%, a relative improvement of over 150%.

VI. CONCLUSION

As demonstrated in previous research and in our experiments, Restricted Boltzmann Machines are capable of learning complex distributions and good representations of the input data. They are also useful as a preprocessing step that improves the classification accuracy for the input data. With their relatively simple training procedures and given recent innovations in the use of RBMs, they are widely applied in many problem domains and are a crucial step for building deep neural networks.

REFERENCES

- [1] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines*. *Cognitive science*, 9(1):147–169, 1985.
- [2] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- [3] Maria Angelica Cueto, Jason Morton, and Bernd Sturmfels. Geometry of the restricted boltzmann machine. *arXiv preprint arXiv:0908.4425*, 2009.
- [4] John J Hopfield and David W Tank. neural computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.
- [5] Radford M Neal. Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113, 1992.
- [6] Carsten Peterson. A mean field theory learning algorithm for neural networks. *Complex systems*, 1:995–1019, 1987.
- [7] Miguel A Carreira-Perpinán and Geoffrey E Hinton. On contrastive divergence learning. *AISTATS 2005*, page 33, 2005.
- [8] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.
- [9] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM, 2008.
- [10] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [11] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 141:142, 2012.
- [12] Travis E Oliphant. *A Guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.