Report
Restricted Boltzmann machines

Nicolas Rahmouni, Michel Deudon, Paul Bertin
(Dated: January 2018)

This work is done for the course *Probabilistic Graphical Models* taught by Guillaume Obozinski and Francis Bach in the master *Mathematiques Vision Apprentissage* in ENS Cachan.

I. INTRODUCTION

Restricted Boltzmann Machines (RBMs) are a specific type of generative models. They were invented by *Smolensky* in 1986 under the name **harmonium** [1] and constitute one of the most common buiding blocks for deep generative models. [2, 3]

RBMs are undirected probabilistic graphical models containing one layer of observable variables and a single layer of latent variables where all observable variables are connected to all latent variables but there are no connections among the visible units, nor any connections among the hidden units. Therefore RBMs are based on a bipartite graph.

This architecture was designed to make the learning process easier and allow efficient sampling such as *block Gibbs sampling*.

We first present Restricted Boltzman Machines and the underlying probabilistic graphical model. In section 3 and 4, we investigate inference and learning algorithms. We then illustrate these methods on the MNIST dataset and discuss results in section 5.

II. RESTRICTED BOLTZMANN MACHINE MODEL

The RBM is an energy based model. Its energy function is given by

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^\intercal \mathbf{v} - \mathbf{c}^\intercal \mathbf{h} - \mathbf{v}^\intercal \mathbf{W} \mathbf{h} \tag{1}$$

where $\mathbf{v}$ stands for visible units, $\mathbf{h}$ for hidden units and $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{W}$ are unconstrained, real-valued, learnable parameters. The probability of a given state is given by

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h})) \tag{2}$$

with Z the partition function that normalizes the probability.

The corresponding graphical model in which we see that the only connections are between the visible and hidden units, is shown in FIG. 1. Thanks to this bipartite architecture, the hidden units are independent given the visible units and vice versa. Therefore we can derive a simple expression for $p(\mathbf{h} \mid \mathbf{v})$. Furthermore, the conditional distributions $p(\mathbf{h} \mid \mathbf{v})$ and $p(\mathbf{v} \mid \mathbf{h})$ factorize nicely which allows us to apply efficient sampling methods during the learning process. In the following sections, we will explain how inference and learning are performed on this model.

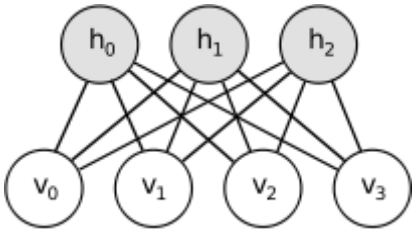


FIG. 1: A Restricted Boltzmann Machine Graphical Model (hidden nodes in gray and visible nodes in white)

III. INFERENCE

The inference problem on RBMs is to estimate one of the conditional probability $p(\mathbf{h} \mid \mathbf{v})$ or $p(\mathbf{v} \mid \mathbf{h})$. In the first, this comes to find the hidden variables for a set of visible variables or to find some missing variables. For example, if the visible variables represent the pixels of an image, we would like to find the object in the scene which are represented by the hidden units.

On the other case, we want to find some missing visible values. On images, we can think of this as an inpainting problem. This can also be used for recommending movies. Each visible unit represents the grade that one user gives to a movie. Obviously, there are a lot of missing data and the idea is to infer the latent variables $\mathbf{h}^u$ for each realization of $\mathbf{v}^u$ (each user) and to compute $p(\mathbf{v} \mid \mathbf{h^u})$ to estimate the grade that the user would give to each movie on the database. So for those examples, there are two inference steps.

At first, computing those probabilities seems difficult as the partition function Z is intractable (and therefore the joint probability distribution too). However, using the structure of the graphical model, we obtain the following expression for a binary RBM which we will use for our experimentation (see Appendix A for the derivations):

$$\forall\ j,\ p(h_j = 1 \mid \mathbf{v}) = \sigma(c_j + \mathbf{v}^T \mathbf{W}_{:,j}) \tag{3}$$

$$\forall\ i,\ p(v_i = 1 \mid \mathbf{h}) = \sigma(b_i + \mathbf{W}_{i,:} \mathbf{h}) \tag{4}$$

with $\sigma(x) = \frac{1}{1+e^{-x}}$ the sigmoid function.

In summary, those formulas permit to infer the hidden variables from the visible one and vice-versa which is the key to solve numerous problems having a well-trained RBM. Those formulas can also be used to sample those variables during training, which can be done very efficiently as they are independent (block Gibbs Sampling).

IV. LEARNING

To learn the parameters of the RBM, we aim to maximize the log-likelihood of the whole model given an input $v = \bar{v}$ which can be written as:

$$ln\ \mathcal{L}(\theta \mid \bar{v}) = ln \sum_h e^{-E(\bar{v},h)} - ln \sum_{v,h} e^{-E(v,h)} \tag{5}$$

with θ the parameters of the model which are $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{W}$. To find the parameters that maximize the log-likelihood, a *Gradient Ascent* algorithm is used. We update sequentially the parameters of the model with the following formula:

$$\theta^{(t+1)} = \theta^{(t)} + \eta \frac{\partial}{\partial \theta^{(t)}} (ln\ \mathcal{L}(\theta^{(t)} \mid \bar{v})) \tag{6}$$

So, to apply this algorithm, we just need to estimate the gradient of the log-likelihood (see Appendix B):

$$\begin{aligned} \frac{\partial ln\ \mathcal{L}(\theta \mid \bar{v})}{\partial \theta} &= -\mathbb{E}_{h \sim p(h|\bar{v})}\left(\frac{\partial E(\bar{v},h)}{\partial \theta}\right) \\ &\quad + \mathbb{E}_{v,h \sim p(v,h)}\left(\frac{\partial E(v,h)}{\partial \theta}\right) \end{aligned} \tag{7}$$

The first term can be computed easily but the second term is intractable as it involves the probability of the joint model which depends on Z. Hence, at each iteration we need to estimate the expectancy using a MCMC method. For each update, a sample $\mathbf{v}^*$ is drawn from the joint distribution and the expectancy is approximated by:

$$\begin{aligned} \mathbb{E}_{v,h \sim p(v,h)}\left(\frac{\partial E(v,h)}{\partial \theta}\right) &= \sum_{v,h} p(v,h) \frac{\partial E(v,h)}{\partial \theta} \\ &\approx \sum_h p(h \mid v^*) \frac{\partial E(v^*,h)}{\partial \theta} \end{aligned} \tag{8}$$

However, MCMC requires a lot of sampling steps to converge to an unbiased sample. Hence, the whole problem now is to find methods that permit to compute efficiently and precisely this expectancy, methods which we will now describe.

A. Contrastive divergence

The *Contrastive Divergence* learning algorithm [4], which has become the standard learning algorithm for RBM, samples a visible variable with a Gibbs chain such that $v^* = v^{(k)}$ is sampled from the k-th step. The chain is initialized at $v^{(0)}$ with a training example and at each step t $h^{(t)}$ is sampled from $p(h \mid v^{(t)})$ and $v^{(t+1)}$ from $p(v \mid h^{(t)})$. Obviously the estimation of the expectancy is biased but the bias tends to zero when k tends to infinity. However the biais can lead to a distortion of the learning process in the sense that the likelihood can diverge after a few iterations when k is small.

A batch version of the *k-step CD* algorithm for a binary RBM is shown in figure 2.

Algorithm 2.1: k-step contrastive divergence

Input: RBM $(V_1, \ldots, V_m, H_1, \ldots, H_n)$, training batch S

Output: gradient approximation Δw_{ij}, Δb_j and Δc_i for $i = 1, \ldots, n$, $j = 1, \ldots, m$

```
init Δw_ij = Δb_j = Δc_i = 0 for i = 1,...,n, j = 1,...,m
forall the v ∈ S do
    v^(0) ← v
    for t = 0,...,k−1 do
        for i = 1,...,n do sample h_i^(t) ~ p(h_i | v^(t))
        for j = 1,...,m do sample v_j^(t+1) ~ p(v_j | h^(t))
    for i = 1,...,n, j = 1,...,m do
        Δw_ij ← Δw_ij + p(H_i = 1 | v^(0))·v_j^(0) − p(H_i = 1 | v^(k))·v_j^(k)
    for j = 1,...,m do
        Δb_j ← Δb_j + v_j^(0) − v_j^(k)
    for i = 1,...,n do
        Δc_i ← Δc_i + p(H_i = 1 | v^(0)) − p(H_i = 1 | v^(k))
```

FIG. 2: Batch version of CD-k

This algorithm computes the global gradient on a training set of visible examples. As we can see, the first for loop corresponds to a Gibbs sampler (we sample the coordinates independently) that computes a sample $v^{(k)}$ from the joint distribution. The three other for loops correspond to the computation of the gradient for this sample (see Appendix B for the formulas of the gradient). Note that we use $\bar{v} = v^{(0)}$ and the sample $v^* = v^{(k)}$ in the formulas derived above. One can note that the operations are highly parallelizable on the coordinates as they are independent. This can be useful when the number of variables in the network is high.

B. Persistent Contrastive divergence

One of the key issues is to lower the time needed to burn in the Markov Chain at each step. The CD algorithm initializes the Markov Chain from the data distribution. However, if we assume that the updates are small enough, the joint probability from the previous step will be close from the current joint probability.

It follows that the samples from the previous model's distribution will be very close to being fair samples from the current model's distribution. Therefore, we can initialize the Markov Chain with the samples from the previous step, and the mixing time of the Markov Chain will be small.

This approach is known as *Persistent Contrastive divergence* (PCD) [5]. Even if it tends to reduce the biais,

it does not avoid the divergence problem.

A variant of PCD algorithm is the *Fast Persistent Contrastive divergence* (FPCD) which introduces *fast* parameters to allow a faster mixing of the Markov Chain [6]. Those parameters are only used for the sampling process and not in the model itself.

C. Parallel Tempering

The idea behind Parallel Tempering [7] is to introduce supplementary Gibbs chains which will sample from the same model but at different temperatures. The higher the temperature, the smoother the probability distribution and thus the faster the mixing. The low temperature chains will benefit from the fast mixing of high temperature chains.

This can be formalized in the following way. Given M temperatures $1 = T_1 < T_2 ... < T_M$ a set of M Markov Chains is defined with probability distributions $p_r(v,h) = \frac{1}{Z_r} e^{-\frac{1}{T_r} E(v,h)}$. In each step of the algorithm, we run k (usually k=1) Gibbs sampling steps for each tempered Markov Chain yielding samples (v_1, h_1) ... (v_M, h_M). Afterwards, two neighbouring Gibbs chains may exchange particles (v_r, h_r) and (v_{r-1}, h_{r-1}) with probability based on the Metropolis ratio :

$$min(1, \frac{p_r(v_{r-1}, h_{r-1}) p_{r-1}(v_r, h_r)}{p_r(v_r, h_r) p_{r-1}(v_{r-1}, h_{r-1})}) \tag{9}$$

We then take the (eventually exchanged) sample v_1 and repeat this procedure L times to obtain L samples.

D. Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) [8] is a sampling method with a high acceptance rate enabling more efficient exploration. The key idea is to simulate a Hamiltonian dynamical system : a particle is given some velocity and moves in the potential field associated to the energy of the model.

Formally, HMC samples from a distribution $\pi(\theta, r) \propto p(\theta \mid \bar{v}) e^{-\frac{1}{2} r^T M^{-1} r}$ where the second term corresponds to the *kinetic energy*.

One of the limitations of the HMC technique is that it needs to compute the gradient of π (which gives the direction where the particle goes) at each step. One can use a noisy gradient estimate based on a subset of the data.

V. EXPERIMENTS

A. Details

As a toy problem we considered the MNIST handwritten digit recognition benchmark [9].The training set consists in 60000 samples of digits. Each image is a 28×28 grayscale pixels, which are binarized with a threshold value of 127. For some examples, see FIG. 3. Our goal is to learn the parameters of the RBM and see if we can reconstruct the original images from the latent variables of the network which would mean that the network has learned a good representation for this dataset.

FIG. 3: Examples of binarized images from the MNIST dataset

In our experiments, we implemented the k-step CD algorithm and investigated the influence of k and the number of hidden units on its convergence. For training, the RBMs were initialized with small random weights sampled from $\mathcal{N}(0, 4\sqrt{\frac{6}{n_v + n_h}})$. If not stated otherwise, 64 hidden units were used for modeling the MINIST data. The models were trained for 10000 epochs, using gradient ascent on CD-k with k in 1, 2, 3, 5, 10. For MNIST the training data was split into mini-batches of 100 samples. The learning rate was $\eta = 0.05$ for CD-learning. To keep the number of hyperparameters small, we did not use a momentum term, nor a weight decay term.

B. Results

We obtained pretty good results. As shown in FIG. 5, the model is able to reconstruct input images from the hidden units, meaning that it has learned a good representation of the digits in a 64-dimensional space. The convergence of the k-CD algorithm on the MNIST dataset is shown in FIG.4. The higher the value of k, the higher the reconstruction error. This is coherent because for higher k values, the reconstructed image passed through more sampling steps. The fact that the reconstruction error increases slowly with k shows that our model separates well the different labels and thus is robust.

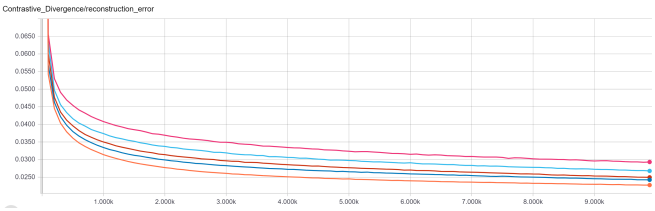


FIG. 4: Reconstruction error (MSE) between input image (test set) and reconstructed image after $k = 100/5/3/2/1$ (from top to bottom) Gibbs sampling.

Note that the value of k used to reconstruct the image is not tied to the value of k used for the training process. From a training point of view, the higher the value of

k, the better our estimation of the joint distribution and thus the faster the convergence. Nonetheless, a higher k also means a higher computational cost at each step and one needs to find a good balance between burning in and computation.

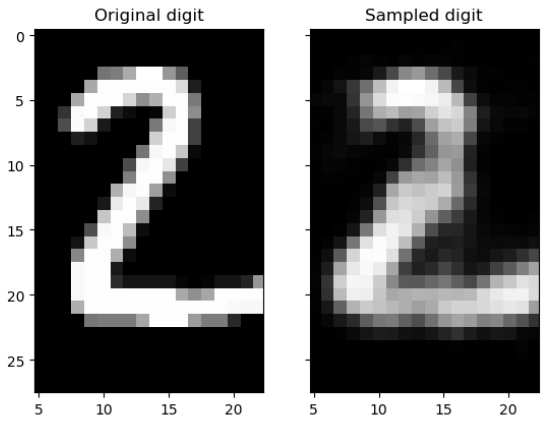


FIG. 5: Reconstruction error with k = 1 after training. Original data (left) and sampled data (right)

In FIG. 6 we show the effect of the number of hidden units on the reconstruction error. The higher the number of hidden units, the lower the reconstruction error. This is consistent with the fact that more hidden units allows the model to capture more complex patterns.

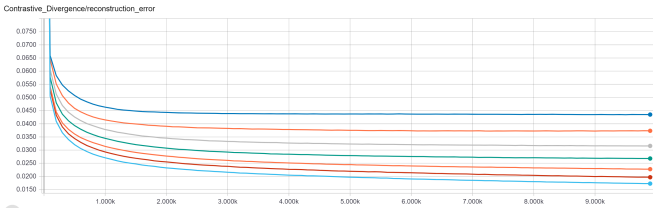
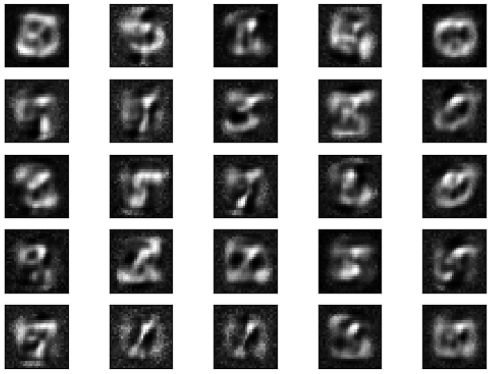
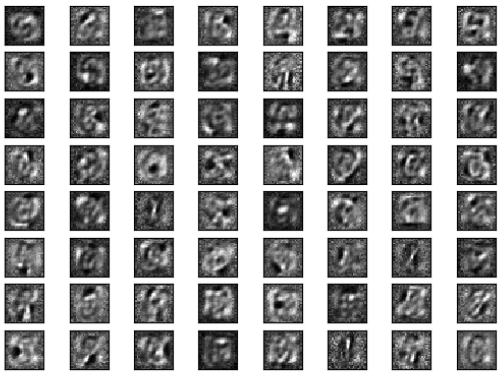


FIG. 6: Reconstruction error (MSE) between input image (test set) and reconstructed image after 1 block Gibbs sampling steps, with $n_h = 16/25/36/49/64/81/100$ (from top to bottom) hidden units.

In FIG. 7 we show the receptive fields associated to each hidden unit : we activate a single hidden unit and then sample visible units according to the conditional distribution. Those images represent the patterns which are most likely to activate a given hidden unit. For instance, with $n_h = 25$, we can see that some hidden units are clearly sensitive to the number 3 and others to the number 0. When the number of hidden units gets higher, the detected patterns are more abstract and the actual digits will be linear combinations of those patterns. This allows the model to detect variations within each class of digit and thus to be more accurate.



(a) $n_h = 25$



(b) $n_h = 64$

FIG. 7: Receptive fields. Images obtained by activating a single hidden neuron and then sampling from the visible nodes. ($n_h = 16$ and $n_h = 64$)

VI. CONCLUSION

Restricted Boltzmann Machines are a very interesting application of Graphical models. They yield very good results as shown here on the MNIST dataset but also on more complex tasks : RBMs were used to win a Netflix competition to predict user/movie ratings on a dataset of over 100 million samples [10].

On top of that, RBMs can be stack together to form Deep Belief Networks which are able to detect more complex patterns.

The training computation cost is still an important drawback, and many sampling methods such as Parallel Tempering and Hamiltonian Monte Carlo try to tackle this issue.

[1] P. Smolensky, "Parallel distributed processing: Explorations in the microstructure of cognition," 1986.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] A. Fischer, *Training restricted Boltzmann machines*. PhD thesis, University of Copenhagen, 2014.

[4] M. . Carreira-Perpin and G. Hinton, "On contrastive divergence learning. artificial intelligence and statistics," 2005.

[5] Tieleman, "Training restricted boltzmann machines using approximations to the likelihood gradient," 2008.

[6] T. Tieleman and G. Hinton, "Using fast weights to improve persistent contrastive divergence,"

[7] Cho, Raiko, and Ilin, "Parallel tempering is efficient for learning restricted boltzmann machines," 2010.

[8] T. Chen, E. B. Fox, and C. Guestrin, "Stochastic gradient hamiltonian monte carlo," 2014.

[9] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[10] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering,"

Appendix A: Inference derivations

We would like to express the conditional probabilities for the RBM model. With the expression of the joint probability, we have:

$$\begin{aligned} p(\mathbf{h} \mid \mathbf{v}) &= \frac{p(\mathbf{h}, \mathbf{v})}{p(\mathbf{v})} \\ &= \frac{\exp(\mathbf{b}^T\mathbf{v} + \mathbf{c}^T\mathbf{h} + \mathbf{v}^T\mathbf{W}\mathbf{h})}{Zp(\mathbf{v})} \\ &= \frac{\exp(\mathbf{c}^T\mathbf{h} + \mathbf{v}^T\mathbf{W}\mathbf{h})}{Z'(v)} \end{aligned}$$

Now, we remark that the hidden variables are independent given the visible ones, so we can write:

$$p(h_i \mid \mathbf{v}) = \frac{\exp(c_i + \mathbf{v}^T\mathbf{W}_{:,i}h_i)}{Z'(v)}$$

If we assume that we have binary variables, h_i can be either 0 or 1. So we can express the probability as the partition function is only the sum of two terms:

$$\begin{aligned} p(h_i = 1 \mid \mathbf{v}) &= \frac{\exp(c_i + \mathbf{v}^T\mathbf{W}_{:,i})}{1 + \exp(c_i + \mathbf{v}^T\mathbf{W}_{:,i})} \\ &= \sigma(c_i + \mathbf{v}^T\mathbf{W}_{:,i}) \end{aligned}$$

The same formula holds for $p(\mathbf{v} \mid \mathbf{h})$ because the problem is symmetric.

Appendix B: Gradient of the log-likelihood

The log-likelihood of the RBM model is written:

$$ln\ \mathcal{L}(\theta \mid \bar{v}) = ln \sum_h e^{-E(\bar{v},h)} - ln\ \sum_{v,h} e^{-E(v,h)}$$

Let's now compute the derivative of this quantity:

$$\begin{aligned} \frac{\partial ln\ \mathcal{L}(\theta \mid \bar{v})}{\partial\theta} &= \frac{\sum_h \frac{\partial e^{-E(\bar{v},h)}}{\partial\theta}}{\sum_h e^{-E(\bar{v},h)}} - \frac{\sum_{v,h} \frac{\partial e^{-E(v,h)}}{\partial\theta}}{\sum_{v,h} e^{-E(v,h)}} \\ &= -\frac{\sum_h e^{-E(\bar{v},h)} \frac{\partial E(\bar{v},h)}{\partial\theta}}{\sum_h e^{-E(\bar{v},h)}} + \frac{\sum_{v,h} e^{-E(v,h)} \frac{\partial E(v,h)}{\partial\theta}}{\sum_{v,h} e^{-E(v,h)}} \end{aligned}$$

Here we recognize that $p(h \mid \bar{v}) = \frac{e^{-E(\bar{v},h)}}{\sum_h e^{-E(\bar{v},h)}}$ and $p(v,h) = \frac{e^{-E(\bar{v},h)}}{\sum_{v,h} e^{-E(v,h)}}$ so we have:

$$\frac{\partial ln\ \mathcal{L}(\theta \mid \bar{v})}{\partial\theta} = -\sum_h p(h \mid \bar{v}) \frac{\partial E(\bar{v},h)}{\partial\theta} + \sum_{v,h} p(v,h) \frac{\partial E(v,h)}{\partial\theta}$$

Which shows that:

$$\frac{\partial ln\ \mathcal{L}(\theta \mid \bar{v})}{\partial\theta} = \mathbb{E}_{v,h\sim p(v,h)}(\frac{\partial E(v,h)}{\partial\theta}) - \mathbb{E}_{h\sim p(h|\bar{v})}(\frac{\partial E(\bar{v},h)}{\partial\theta})$$

For a binary RBM, let's derive the equations for each parameter (using the approximation in Equation (8)):

$$\begin{aligned} \frac{\partial ln\ \mathcal{L}(\theta \mid \bar{v})}{\partial w_{i,j}} &= \mathbb{E}_{v,h\sim p(v,h)}(-v_i h_j) - \mathbb{E}_{h\sim p(h|\bar{v})}(-\bar{v}_i h_j) \\ &\approx p(h_j = 1 \mid \bar{v})\bar{v}_i - p(h_j = 1 \mid v^*)v_i^* \end{aligned}$$

$$\begin{aligned} \frac{\partial ln\ \mathcal{L}(\theta \mid \bar{v})}{\partial b_i} &= \mathbb{E}_{v,h\sim p(v,h)}(-v_i) - \mathbb{E}_{h\sim p(h|\bar{v})}(-\bar{v}_i) \\ &\approx \bar{v}_i - v_i^* \end{aligned}$$

$$\begin{aligned} \frac{\partial ln\ \mathcal{L}(\theta \mid \bar{v})}{\partial c_i} &= \mathbb{E}_{v,h\sim p(v,h)}(-h_i) - \mathbb{E}_{h\sim p(h|\bar{v})}(-h_i) \\ &\approx p(h_j = 1 \mid \bar{v}) - p(h_j = 1 \mid v^*) \end{aligned}$$