# Training a Restricted Boltzmann Machine

## 1. Introduction

Boltzmann Machines (BMs), which is a type of Markov random field graphical model (Fischer & Igel, 2012), was first presented in 1985(Hinton & Sejnowski, 1986), he describes BMs an undirected connected graph. A BM is used to learn aspects of an unknown probability distribution from samples of that distribution, the learning process of a BM is difficult and not viable in practice. However, certain constraints to the network topology can be imposed leading us to a Restricted Boltzmann Machine.

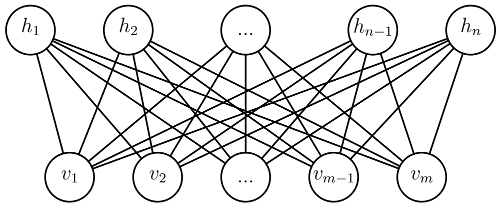
A Restricted Boltzmann Machine (RBM) is defined as a complete undirected bipartite graph, even though it is a special case of Boltzmann Machine, it was show by (Long & Servedio, 2010) that if $P \neq NP$, given and RBM M and a n-bit input x there is no polynomial-time algorithm that outputs an estimate of the probability of x by M with a factor of $e^{Kn}$ for any fixed positive constant $K$, more than that assuming that $RP \neq NP$, there is no polynomial-time randomized algorithm to generate a random sample from a probability distribution whose total variation distance from the distribution defined by M is at most 1/12. The goal of this project is to learn an effective set of parameters of a RBM for a specific application (image of digits), the learning process involves estimating the probability of an n-bit string and collecting a sample from an RBM thus the learning process is computationally intensive.

## 2. Definitions

An undirected bipartite graph can be represented as a triple $G(V, H, E)$, where $V$ and $H$ are a finite set of vertex with $V$ and $H$ being disjoint and $E$ is a set of undirected edges and every edge in $G$ connects a vertex of $U$ to a vertex of $H$. The set of vertices $V$ will be defined as visible units and the set of vertices $H$ as hidden unit. A representation of an RBM with m visible units and n hidden units is show on figure 1.

In this project, we will consider a RBM defined over a set of a binary states for the visible and hidden units i.e $v_i \in \{0,1\} \forall i = 1,2,...,m$ and $h_j \in \{0,1\} \forall j = 1,2,...n$, thus a visible vector $v \in 0,1^m$ and a hidden vector

*Figure 1.* The undirected graph of an RBM with m visible units and n hidden units



$h \in 0,1^n$. For example, consider a binary digit image of size $28X28$, it can be represented as a binary vector (or string) of size 784, the image pixels corresponds to the RBM visible units because they are observed and the hidden are commonly interpreted as features detectors.

Let $W = (w_{ij})$ be a weight matrix where $w_{ij}$ is the weight of the connection between the visible unit $i$ and hidden unit $j$, $v_i$ and $h_j$ be the binary state of the visible unit i and hidden unit j and $b_j, a_i$ be the biases of the jth and ith hidden and visible unit respectively, finally $\theta = \{W, a, b\}$ is the RBM set of parameters. The joint configuration of the visible and hidden unit is denoted as (v, h), the energy $E(v, h)$ of a joint configuration is given by:

$$E(v,h) = -\sum_{i \in V} a_i v_i - \sum_{j \in H} b_j h_j - \sum_{i \in V} \sum_{j \in H} v_i h_j w_{ij} \quad (1)$$

Which can be represented in a matrix notation as:

$$E(v,h) = -a^T v - b^T h - v^T W h \quad (2)$$

The probability of a joint configuration of visible and hidden unit (v, h) is defined as:

$$P(v,h) = \frac{e^{-E(v,h)}}{Z} \quad (3)$$

Where $Z$ is the normalizing factor called partition function, which is the sum of the probabilities over all possible configuration of the network. It can be seen by equation 4, that the partition function computation is computationally intractable because it is a sum over an exponential number of possible states of the visible and hidden layer.

$$Z = \sum_{v \in \{0,1\}^m} \sum_{h \in \{0,1\}^n} e^{-E(v,h)} \tag{4}$$

The probability of a single visible vector $v$ is given by

$$p(v) = \frac{\sum_{h' \in \{0,1\}^n} e^{-E(v,h')}}{Z} \tag{5}$$

Even though $p(x)$ and $p(v,h)$ are computationally intractable because of the partition function, the conditioned probability $p(h|v)$ and $p(v|h)$ are feasible to compute. It shows that to take a sample of the hidden units given the visible units or a sample of the v visible given the hidden units is a feasible task. Let $\sigma$ be the sigmoid function, the following propositions shows the conditioned probability result.

**Proposition (1):** Given an RBM with parameters $\theta$, the probability of a hidden vector given a visible vector $p(h|v)$ is given by $p(h|v) = \prod_{j \in H} p(h_j|v)$ and $p(h_j = 1) = \sigma(b_j + v^T W_{\cdot j})$.

**Proposition (2):** Given an RBM with parameters $\theta$, the probability of a visible vector given a hidden vector $p(v|h)$ is given by $p(v|h) = \prod_{k \in V} p(v_k|h)$ and $p(v_k = 1) = \sigma(a_k + h^T W_{k\cdot}^T)$.

We have that an RBM models the joint distribution $p(v,h)$, one important concept is the marginal distribution $p(v)$ which would show us what to change to increase the probability of x. It is show on the appendix that $p(v)$ can be written as $p(v) = e^{-F(v)}/Z$ where $F(v) = -a^T v - \sum_{j=1}^{m} softplus(b_j + v^T W_{\cdot j})$ where $softplus(x) = log(1 + e^x)$. The free energy concept will be used to derive the proposed method.

The objective function of an RBM is to maximize the log likelihood of the training data. Let $T = (x^{(1)}, x^{(2)}, ..., x^{(t)})$ be a training set with $t$ samples the loss function $\mathscr{L}(\theta, T)$ of an RBM with parameters $\theta$ over a data set $T$ is is define as:

$$\mathscr{L}(\theta, T) = \frac{1}{t} \sum_i -log\, p(x^{(i)}) \tag{6}$$

The derivative of the loss function with respect of the parameter $\theta$ of the model is:

$$\frac{\delta - log\, p(x^{(i)})}{\delta\theta} = \mathbb{E}_h \left[ \frac{\delta E(x^{(i)}, h)}{\delta\theta} | x^{(i)} \right] - \mathbb{E}_{v',h'} \left[ \frac{\delta E(x', h')}{\delta\theta} \right] \tag{7}$$

The first term $\mathbb{E}_h \left[ \frac{\delta E(x^{(i)},h)}{\delta\theta} | x^{(i)} \right]$ is called the positive phase, it is feasible to compute because it is conditioned on the training sample. The second term $\mathbb{E}_{v',h'} \left[ \frac{\delta E(x',h')}{\delta\theta} \right]$ is called negative phase, this one is intractable because it depends on the model visible and hidden units.

The goal of this project is to implement techniques from the literature to estimate the value of the negative phase and to propose a simple new technique to improve the learning process. The baseline which will be considered is the contrastive divergence (CD) which was first presented by.

## 3. Baseline

### 3.1. Contrastive Divergence

The most common method to train an RBM is the contrastive divergence (CD), this method for training an RBM was first proposed by (Hinton, 2002). Since the negative phase is intractable a point estimate of its value is used. One approach would be to run a Markov Chain Monte Carlo (MCMC) until reaching the stationary distribution, instead a Gibbs chain is run for k steps leading us to CD-k method. A Gibbs chain starts with a training sample $x^{(0)}$ and runs for k steps yielding a sample $x^{(k)}$, each steps consists of sampling the hidden unit given the visible unit $p(h^{(i)}|v^{(i)})$ and subsequently sampling the visible unit given the hidden unit $p(v^{(i+1)}|h^{(i)})$, this process is repeated k times.

The following python style pseudo-code describes the CD-k. It has as input an RBM, sample $x_0$ from the training set and the number of steps $k$. We initialize the chain $x_i$ as the training sample $x_0$, next for $k$ steps we sample the hidden units given the visible units $x_i$ then we sample the visible unit given the sampled hidden unit and set $x_i$ to be the next state of the visible units.

```
def cd_k(RBM, x_0, k):
    x_i = x_0
    for (i in range(k)):
        h_i = sample_h_given_v(x_i)
        x_i = sample_v_given_h(h_i)

    return x_i
```

One important detail is that $v^{(k)}$ is a biased estimate because it is not a sample from the stationary distribution. It was show by (Bengio and Delalleau) that the bias of a converging Gibbs chain starting at $v^{(0)}$ vanishes to zero as k goes to infinity.

### 3.2. Persistent Contrastive Divergence

The persistent contrastive divergence (PCD) was presented by (Tieleman, 2008). It takes advantage that the the model changes slightly between parameters updates, therefore instead of reinitializing the chain each time at $v^{(0)}$ and computing negative sample $x^{(k)}$ it keeps a persistent chain of that and starts the Gibbs chain at the state it ended for the previous model.

The following pseudo code describes the PCD. It receives as input an RBM, a persistent chain for a sample $x$ and a number of Gibbs sampling iteration. In the first line, it sets $x_i$ as the persistent chain $px_0$ from the previous model, next it works exactly the same as CD for k steps and in the end it returns the new state of the persistent chain.

```
def cd_k(RBM, px_0, k):
    x_i = px_0
    for (i in range(k)):
        h_i = sample_h_given_v(x_i)
        x_i = sample_v_given_h(h_i)

    return x_i
```

For this project two baselines to estimate the negative phase were considered. The first one, is contrastive divergence (CD), which was first presented by (Hinton, 2002). The second baseline is persistent contrastive divergence (PCD), which was first presented by (Tieleman, 2008).

### 3.3. Proposed method

The goal of the proposed method is to be a simplification of Parallel Tempering (PT) it was first proposed to train RBMs by (Cho et al., 2010). The idea of PT is to keep a large number of 'parallel' chains running persistent contrastive divergence but each chain is running at a different temperature, after performing a step of parameters update it has a rule to exchange samples between neighbours chains based on Metropolis Hastings (MH) acceptance probability. The method proposed was to instead of keeping a large number of parallel chains we make a simple perturbation of the negative samples generated after performing the negative phase and before updating the parameters by and use MH to accept or reject the proposed modification.

Let $x$ be the current state of a negative sample generate by an RBM, the proposed method makes a perturbation of $x$ by flipping the value of two bits that are chosen uniformly creating a sample $\widetilde{x}$. One MH common choice for accepting a change $\widetilde{x}$ given $x$ is given by:

$$A(\widetilde{x}|x) = min\big(1, \frac{p(\widetilde{x})g(x|\widetilde{x})}{p(x)g(\widetilde{x}|x)}\big)$$

Where $g(\widetilde{x}|x)$ is the proposal distribution, which indicates the probability of moving to state $\widetilde{x}$ given that we currently are on state $x$, in out case is the probability of flipping two bits that are chosen uniformly. It can be show that the proposal distribution used in the proposed method is symmetric thus $g(\widetilde{x}|x) = g(x|\widetilde{x})$. As it was previously show, the marginal probability $p(x)$ can be written in terms of the free energy as $p(x) = e^{-F(x)}/Z$ thus we get the following acceptance probability:

$$A(\widetilde{x}|x) = min\big(1, e^{-(F(\widetilde{x})-F(x))}\big)$$

The following pseudo-code summarizes the proposed method. The proposed method (pm) receives as input an RBM a persisted sample px_0 and a number $k$ of steps to update the persisted chain. Initially it updates the persisted input sample $px_0$by performing PCD $k$ times, then it computes a proposal change to the negative sample generated by flipping the value of two bits that are chosen uniformly. Then it flips a coin with a probability of success defined by $A(\widetilde{x}|x)$, where $\widetilde{x}$ is the propose variable and $x$ is the negative variable.

```
def pm(RBM, px_0, k):
    negative = pcd(RBM, px_0, k)
    propose = flip_bits(negative, 2)
    if(flip_coin(propose, negative)):
        return popose
    return negative
```

d, the marginal probability $p(x)$ can be written as $p(x) = e^{-F(x)/Z}$.

### 3.4. Parameters Update

It is show on the appendix that using the equation (7) we can obtain the following update rule for each of the parameters RBM parameters $\theta = \{W, a, b\}$:

$$W = W + \lambda(xh(x)^T - \widetilde{x}h(\widetilde{x})^T)$$

$$a = a + \lambda(h(x) - h(\widetilde{x}))$$

$$b = b + \lambda(x - \widetilde{x})$$

where $h(x) = \sigma(b + x^T W)$ and $\lambda$ is the learning rate term which indicates how much we move in the gradient direction. The training is done by mini-batch,

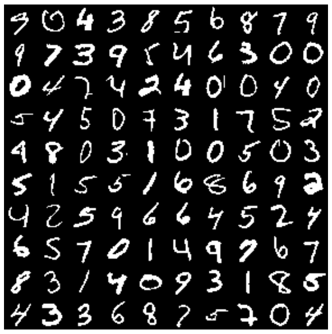


*Figure 2.* Sample digits from MNIST training data

where we partition the training input data in blocks of size $k'$. For each mini-batch it computes the mini-batch accumulated parameter update in the end the parameters are updated using the accumulated value.

## 4. Experiments

### 4.1. Experimental set up

The algorithms described in this project were implemented using python version 3 and numpy, the code is available at [1]. The experiments were executed in a machine with 32 GB of RAM memory with an Intel i5-4460 processor with 3.20GHz.

The number of hidden units were fixed to 500. The learning was done by mini-batch of size 20 throughout 15 epochs, the learning rate was set to 0.01 with no decay. After each epoch a training pseudo likelihood estimate was computed, three metrics were considered: mean reconstruction error, mean cross entropy and an estimate of the mean pseudo likelihood but for simplicity only the pseudo likelihood will be reported.

### 4.2. Datasets

The dataset used for experimental evaluation was the MNIST dataset (LeCun & Cortes, 2010), which contains handwritten digit images of size 28 by 28 in gray scale. The MNIST dataset contains a training set with 60.000 sample digit images, which was split in two sets one with 50.000 samples and the other with 10.000 samples for validation, and a test set with 10.000 digit images. Since our RBM is only defined for binary input the input was binarized by rounding the pixels values to the closest integer. Figure 4.2 shows 100 digit sample from the training set.

[1]https://github.com/tcriscuolo/RBM

### 4.3. complexity analysis

**Time complexity**: The asymptotic time complexity depends on some parameters that are user defined such as the number of epochs $e$, number of hidden units $n$, number of Gibbs steps $k$. It is important to note that all the computation are written as matrix and vector operations. To compute the time complexity we have to consider the cost of performing the Gibbs sampling to generate the negative particle $\widetilde{x}$ and updating the model parameters.

To perform one Gibbs steps we have to sample the hidden units given the visible unit and then sample the visible unit given the current sampled hidden unit, this is done by Bernoulli trial where the probability of activating each hidden or visible unit is given by proposition 1 and 2. The time complexity of performing one Gibbs step is $O(mn + 2n + nm + 2m) = O(mn)$, the $mn$ and $nm$ term comes from the vector matrix product. The cost of computing each parameter update is dominated by the outer product between $x\widetilde{x}^T$ which asymptotically is $O(m^2)$. Thus the total cost of training an RBM with $e$ epochs with each epoch performing $k$ Gibbs sampling steps is given by $O(e \times k \times n \times m^3)$.

**Space complexity**: All the algorithms algorithms implemented in this project have the same asymptotic space complexity. For each one of them we have to store a weight matrix $W \in \mathbb{R}^{m\times n}$ and two bias vectors $a \in \mathbb{R}^m$ and $b \in \mathbb{R}^n$, in particular for PCD and PM we have to store a persistent chain $P \in \{0,1\}^{|B|\times m}$, where $|B|$ is the mini batch size. Lastly we also have to store the training data $T \in \{0,1\}^{t\times m}$, where $t$ is the number of training samples. Thus the space complexity is given by $O(mn + tm) = O(m(n + t))$.

### 4.4. Results

Restricted Boltzmann Machines are considered trick to evaluate. It is not know a right or best method to evaluate an RBM, for this project three common evaluation criteria to evaluate an RBM were considered: estimation of the log likelihood on training data, visual evaluation of the filters learned by each model and sampling negative particles from each model.

The first evaluation metric considered was the estimated likelihood during the training phase the estimation was done using the method described at [2]. Figure 4.4, shows the estimated likelihood of the best performing parameter for each of the models (CD, PCD, PM). It can be seen that CD showed a better estimated likelihood than PCD, this is because CD spend most of its probability mass in the training samples neighbour-

[2]http://deeplearning.net/tutorial/rbm.html

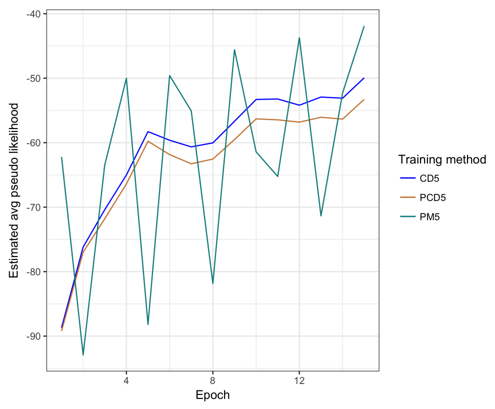


*Figure 3.* Comparison between the best performing configuration of each training method

hood, thus it is reasonable that it will have a higher likelihood on the training data. The proposed method showed a more aggressive learning curve, in one epoch it has a high likelihood and at the subsequent epoch it decreases dramatically, it might be because it started to explore a new region of the probability space thus decreasing the probability mass of the training samples. Even though, the proposed method has a high variance between each epoch it can be seen that it has an increasing learning curve.

The second evaluation metric, was a visual inspection of the filters learned by each model. Since, each hidden unit has a connection to each visible units, we can take the weight of this connection and transforms it into an image, the images 4, 5 and 6 shows as each box the connection weights of a hidden unit. The weights are normalized to a scale between 0 and 1, darker pixels shoes that if the visible unit is 1 then this hidden unit has a lower probability of activating and whither pixels shows that if the visible unit is 1 then this hidden unit has a high energy of being activated. It can be seen that the filters learned by CD-5 are difficult to interpret most of them do not form a reasonable pattern or the filter just learned to recognize a digit which is not useful. Next, PCD-5 and PM-5 learned some filters that identify pen-stroke, those filters are more interesting because the model learned that digits can be decomposed as pen-strokes.

The last evaluation criteria, was to sample negative particles from each model by starting from a random initialized image, the goal of this criteria is to check if the model learned using each one of the methods described can generate random digits. Figures 7, 8 and 9 shows negative samples (particles) taken from each model, each column contains a distinct Gibbs Chain, all three methods starts from the same random im-

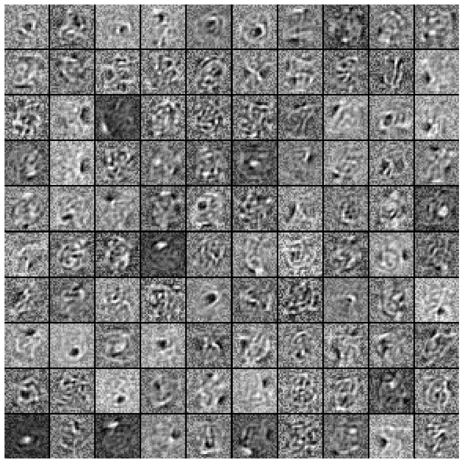


*Figure 4.* Filters learned by CD-5

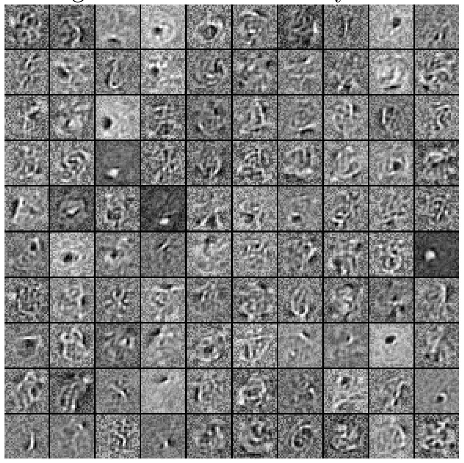


*Figure 5.* Filters learned by PCD-5

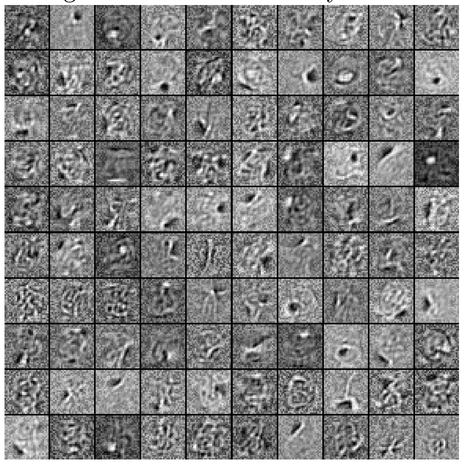


*Figure 6.* Filters learned by PM-5

age, each row is the previous row updated performing 100 steps of Gibbs sampling. It can be seen that CD-5 never generates a reasonable digits, indicating that this learning method just expends the probability mass around the training samples and this is not good enough for generate new random digits. In contrast to CD-5, PCD-5 that had a smaller estimated log likelihood was able to generate random digits, but it generated only the digits 9 and 6, showing that it was not able to explore the probability space well enough for the given amount of training time and some Gibbs chains takes a few iteration to converge to an actual digit. The proposed method showed a visible best performance, it was able to converge to an actual digit faster than PCD-5 and the generated digits were diverse it was not able to generate only the digits 0 and 1.

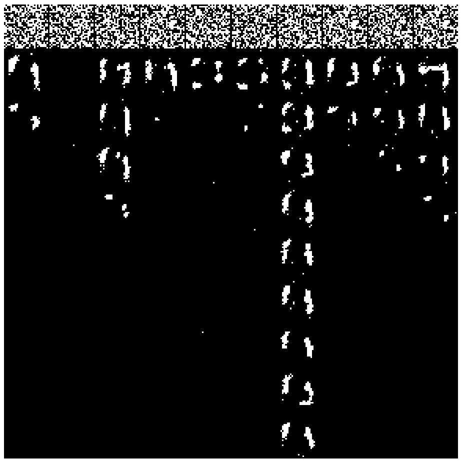


*Figure 7.* CD-5 negative samples

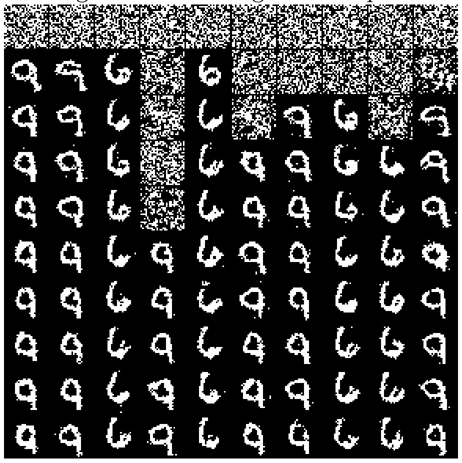


*Figure 8.* PCD-5 negative samples

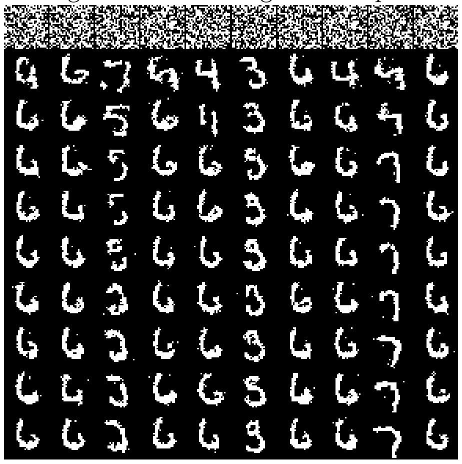


*Figure 9.* PM negative samples

## 5. Conclusion

Restricted Boltzmann Machines, is a probabilistic graphical model based on energy that is able to learn an unknown probability distribution. The negative phase of the likelihood gradient with respect to a model parameter is not practical to compute since it has to compute a sum over a exponential number of terms (each possible configuration of visible and hidden unit), thus one solution is to substitute it by a point estimate. The Contrastive and Persistent Contrastive Divergence are the most popular methods to find a point estimate for the negative phase, each one with its advantage and disadvantage and a new simple method was proposed that showed to be able to speed up the learning process.

## References

Cho, KyungHyun, Raiko, Tapani, and Ilin, Alexander. Parallel tempering is efficient for learning restricted boltzmann machines. In *IJCNN*, pp. 1–8. IEEE, 2010. ISBN 978-1-4244-6916-1. URL http://dblp.uni-trier.de/db/conf/ijcnn/ijcnn2010.html#ChoRI10.

Fischer, Asja and Igel, Christian. *An Introduction to Restricted Boltzmann Machines*, pp. 14–36. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-33275-3. doi: 10.1007/978-3-642-33275-3_2. URL http://dx.doi.org/10.1007/978-3-642-33275-3_2.

Hinton, G. E. and Sejnowski, T. J. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning and Relearning in Boltzmann Machines, pp. 282–317. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL http://dl.acm.org/citation.cfm?id=104279.104291.

Hinton, Geoffrey E. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002. ISSN 0899-7667. doi: 10.1162/089976602760128018. URL http://dx.doi.org/10.1162/089976602760128018.

LeCun, Yann and Cortes, Corinna. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Long, Philip M. and Servedio, Rocco A. Restricted boltzmann machines are hard to approximately evaluate or simulate. In Frnkranz, Johannes and Joachims, Thorsten (eds.), *ICML*, pp. 703–710. Omnipress, 2010. URL http://dblp.uni-trier.de/db/conf/icml/icml2010.html#LongS10.

Tieleman, Tijmen. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 1064–1071, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390290. URL http://doi.acm.org/10.1145/1390156.1390290.

## 6. Appendix

### 6.1. Free energy

The marginal distribution $p(x)$ can be written as $p(x) = e^{-F(x)}/Z$ where

$$F(X) = -a^T x - \sum_{j=1}^{m} softplus(b_j + x^T W_{.j})$$

and

$$softplus(x) = log(1 + e^x)$$

*Proof.* Let $Z$ be the partition function, we have from the definition that

$Zp(x) = \sum_{h\in\{0,1\}^m} e^{-E(x,h)}$

$Zp(x) = \sum_{h\in\{0,1\}^m} e^{a^T x + b^T h + x^T W h}$

we can move $e^{a^T x}$ to outside the summation since it does not depend on $h$, thus

$Zp(x) = e^{a^T x} \sum_{h\in\{0,1\}^m} e^{b^T h + x^T W h}$

$Zp(x) = e^{a^T x} \sum_{h_1\in\{0,1\}} \cdots \sum_{h_m\in\{0,1\}} e^{\sum_j b_j h_j + x^T W_{.j} h_j}$

since we have an exponential of summation we can transform it in products of exponential

$Zp(x) = e^{a^T v} \prod_{j=1}^{m} e^{\sum_{h_j\in\{0,1\}} b_j h_j + v_j W_{.j} h_j}$

for each term in the product above we have only two cases, expanding both cases we have

$Zp(x) = e^{a^T x} \prod_{j=1}^{m} 1 + e^{b_j - x^T W_{.j}}$

using the fact that $e^{log(x)} = x$, we have that

$Zp(x) = e^{a^T x} \prod_{j=1}^{m} e^{log(1+e^{b_j + x^T W_{.j}})}$

$Zp(x) = e^{a^T x} e^{\sum_{j=1}^{m} log(1+e^{b_j + x^T W_{.j}})}$

$Zp(x) = e^{a^T x + \sum_{j=1}^{m} log(1+e^{b_j + x^T W_{.j}})}$

making $F(X) = -a^T x - \sum_{j=1}^{m} softplus(b_j + x^T W_{.j})$

$p(x) = e^{-F(x)}/Z$

□

### 6.2. Conditioned Probability

Given an RBM with parameters $\theta$, the probability of a hidden vector given a visible vector $p(h|v)$ is given by $p(h|v) = \prod_{j\in H} p(h_j|v)$ and $p(h_j = 1) = \sigma(b_j + W_j.v)$.

*Proof.* We have that

$p(h|v) = \frac{p(v,h)}{p(x)}$

$p(h|v) = \frac{e^{-E(v,h)}/Z}{\sum_{h'\in\{0,1\}^m} e^{-E(v,h')}/Z}$

we have that the partition function cancels out and expanding the numerator and denominator we get

$p(h|v) = \frac{e^{a^T v + b^T h + v^T W h}}{e^{\sum_{h'\in\{0,1\}} e^{a^T x + b^T h' + v^T W h'}}}$

we can cancel $e^{a^T v}$ from the numerator and denominator and using a similar manipulation to find the free energy we can write

$p(h|v) = \frac{e^{\sum_j b_j h_j + v^T W_{.j} h_j}}{\prod_j (1+e^{b_j + v^T W_{.j}})}$

since the numerator is an exponential of summation we can write it as a product of exponential and both numerator and denominator are products over same interval thus

$p(h|v) = \prod_j \frac{e^{b_j h_j + v^T W_{.j} h_j}}{(1+e^{b_j + v^T W_{.j}})}$

multiplying the numerator and denominator by $e^{-b_j - v^T W_{.j}}$ we get

$p(h|v) = \prod_j \frac{1}{1+e^{-b_j - v^T W_{.j}}}$

$p(h|v) = \prod_j sigmoid(b_j + v^T W_{.j})$

$\square$

## 6.3. Gradient function

The derivative of the loss function with respect to a parameter $\theta$ is

*Proof.* We have that $p(v) = \frac{\sum h' e^{-E(v,h')}}{Z}$ where $Z = \sum_{v'} \sum_{h'} e^{-E(v',h')}$ thus by the chain rule we get

$\frac{\delta - log(p(v))}{\delta\theta} = \frac{1}{p(v)} \frac{\delta p(v)}{\delta\theta}$

applying the quotient rule we get

$\frac{-Z}{\sum_{h'} e^{-E(v,h')}} \left[ \frac{1}{Z} \frac{\delta \sum_{h'} e^{-E(v,h')}}{\delta\theta} - \frac{\sum_{h'} e^{-E(v,h)}}{Z} \frac{\delta Z}{\delta\theta} \right]$

putting the summation over $h'$ in evidence

$\sum_{h'} \left( \frac{-1}{\sum_{h^*} e^{-E(v,h^*)}} \frac{\delta e^{-E(v,h')}}{\delta\theta} \right) - \frac{1}{Z} \frac{\delta Z}{\delta\theta}$

computing the derivative of the exponential in the left side we get

$\sum_{h'} \frac{e^{-E(v,h)}}{\sum_{h^*} e^{-E(v,h^*)}} \frac{\delta E(v,h')}{\delta\theta} - \frac{1}{Z} \frac{\delta Z}{\delta\theta}$

which is equivalent to

$\sum_{h'} p(h|v) \frac{\delta E(v,h')}{\delta\theta} - \frac{\sum_{v,h} e^{-E(v,h)}}{Z} \frac{\delta E(v,h)}{\delta\theta}$

the right side is also equivalent to

$\sum_{h'} p(h|v) \frac{\delta E(v,h')}{\delta\theta} - \sum_{v,h} p(v,h) \frac{\delta E(v,h)}{\delta\theta}$

thus we get

$\mathbb{E}_h \left[ \frac{\delta E(v,h)}{\delta\theta} |v \right] - \mathbb{E}_{v,h} \left[ \frac{\delta E(v,h)}{\delta\theta} \right]$

as desired $\square$

## 6.4. Estimated Likelihood

Plots 6.4, 6.4 and 6.4 shows the estimated log likelihood during the training, CD-k stands for training using a Gibbs chain of length k to estimate the negative particle. It can be seen that increasing the value of k increases the model performance according to the estimated likelihood.

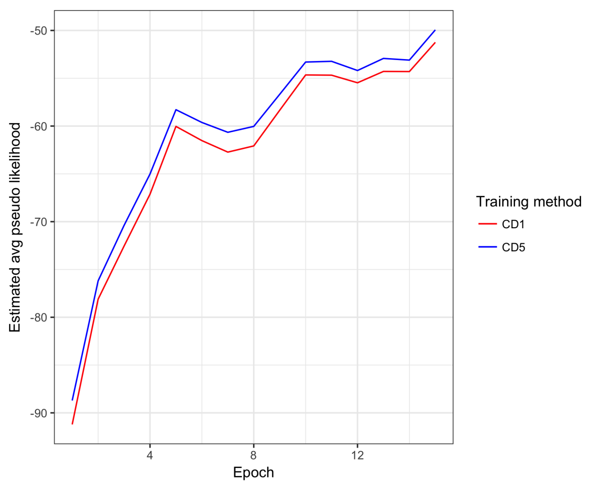


*Figure 10.* CD-1 and CD-5 mean estimated likelihood

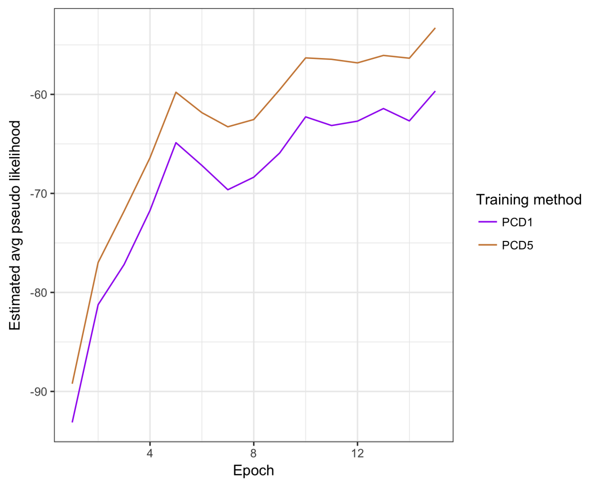


*Figure 11.* PCD-1 and PCD-5 mean estimated likelihood

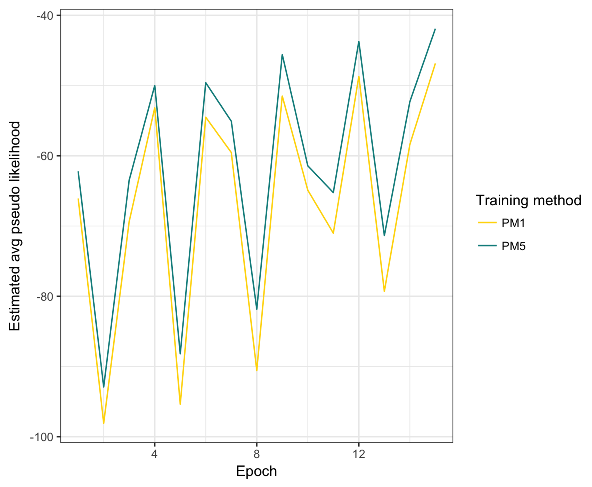


*Figure 12.* PM-1 and PM-5 mean estimated likelihood