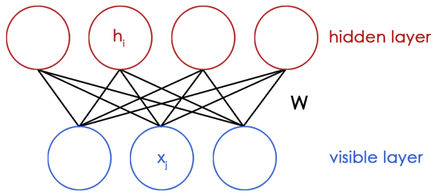
Restricted Boltzmann Machine

{ Chaïmaa Kadaoui, Othman Sbai and Xi Shen } MVA - ENS Paris Saclay

Definition

Restricted Boltzmann Machines are particular instance of **undirected graphical models** inspired by neural networks, where the units are organized in two layers. One layer of *visible* units and one layer of *hidden* units with no connections within each layer.



It is *restricted* because no connection between nodes of the same layer is supposed. The matrix W characterizes the connections (weights) between the two layers. For simplification, we suppose that the variables $\mathbf{x}$ and $\mathbf{h}$ are binary. This model is called Bernoulli RBM.
The hidden variables can be seen as features and the goal of the RBM is to represent meaningful features.

Energy and Joint probability

We introduce the two bias vectors c and b to define the **energy** of this graph:

$$E(x, h) = -h^\top W x - c^\top x - b^\top h$$

We can write the **joint probability** of $\mathbf{x}$ and $\mathbf{h}$ as:

$$p(x, h) = \exp(-E(x, h))/Z$$

Z is the partition parameter which can be computed by calculating all the possibles values of $\mathbf{x}$ and $\mathbf{h}$. In practice, this parameter is intractable.
Conditional inference We can prove that given $\mathbf{x}$, h_j follows a Bernoulli:

$$p(h_j = 1|x) = sigm(b_j + W_{j.}x)$$

$W_{j.}$ is the j-th row of W and $sigm$ defines the sigmoid function. We can prove a similar result for $p(x_i = 1|h)$.

Applications of RBM

- **RBM as a generative model**: generate samples from data distribution
- **RBM as feature extractor**: unsupervised feature extraction from data, instead of handcrafting features
- **RBM for computer vision**: such as object recognition, image denoising and inpainting.
- **RBM for collaborative filtering**: given a set of N users and M movies, we would like to recommend movies to the users.

Training RBM - Contrastive Divergence [2]

To train the RBM with our data $\mathbf{x}^{(t)}$, we would like to minimize the average loss: $\frac{1}{T}\sum_t l(f(\mathbf{x}^{(t)})) = \frac{1}{T}\sum_t -\log p(\mathbf{x}^{(t)})$
We will apply a stochastic gradient descent. We derive each term of the sum with respect to our model parameter θ as follow (where $\mathbb{E}_h$ and $\mathbb{E}_{x,h}$ are expectations of h and (x, h) respectively):

$$\frac{\partial -\log p(\mathbf{x}^{(t)})}{\partial \theta} = \mathbb{E}_h\left[\frac{\partial E(\mathbf{x}^{(t)}, h)}{\partial \theta}|\mathbf{x}^{(t)}\right] - \mathbb{E}_{x,h}\left[\frac{\partial E(\mathbf{x}, h)}{\partial \theta}\right]$$

The first term is called the *positive phase* and the second term the *negative phase*. Because of the difficulty to compute the second term, we will use an algorithm called *Contrastive Divergence*. The key points of the algorithm are:

- We estimate the expectation $\mathbb{E}_{x,h}$ by sampling a single point $\tilde{\mathbf{x}}$.
- To do so, we use Gibbs sampling as a MCMC technique (we apply it k times).
- We initialize our Gibbs sampling with $\mathbf{x^{(t)}}$.
- Update parameters W, b, c; with $h(x) = p(h|x) = sigm(b + Wx)$ and α the learning rate.

$$W \leftarrow W + \alpha\left(h(\mathbf{x^{(t)}})\mathbf{x^{(t)}}^\top - h(\tilde{\mathbf{x}})\tilde{\mathbf{x}}^\top\right) \quad , \quad b \leftarrow b + \alpha\left(h(\mathbf{x^{(t)}}) - h(\tilde{\mathbf{x}})\right) \quad , \quad c \leftarrow c + \alpha\left(\mathbf{x^{(t)}} - \tilde{\mathbf{x}}\right)$$

References

[1] Asja Fischer and Christian Igel. Training restricted boltzmann machines: An introduction. *Pattern Recognition*, 47(1):25–39, 2014.

[2] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

[3] Hugo Larochelle. Neural networks: Restricted boltzmann machine.

[4] 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.

Results On MNIST and CIFAR-10 data

We implemented the RBM training algorithm in python and then we compared our implementation performance with an existing deep learning library using tensorflow (YADLT), we applied it to the MNIST dataset (images of handwritten digits) and CIFAR-10 (color images of different classes). In the case of image data, each pixel is considered as a node; the intensity $p \in [0, 1]$ of a pixel is considered as a probability.

The figure 1 is the evolution of the average stochastic reconstruction $\|\mathbf{x^{(t)}} - \tilde{\mathbf{x}}\|^2$ for both the training and the test dataset at each iteration.

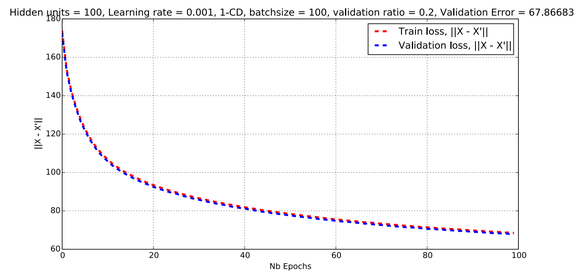


Figure 1: Evolution of training loss over epochs

Generating samples from the learned distribution gives us the following reconstruction images shown in figure 2.

Figure 2: Generated samples after 20 epochs

The figure 3 is the visualization of the extracted features.

Figure 3: Extracted features after 20 Epochs, left: MNIST, right: CIFAR

In order to understand the influence of parameters involved in RBM training, we compare the evolution of the reconstruction loss over epochs of different batch sizes (figure 4) and different number of hidden units (figure 5) and different number of contrastive divergence steps.

Batch size influence

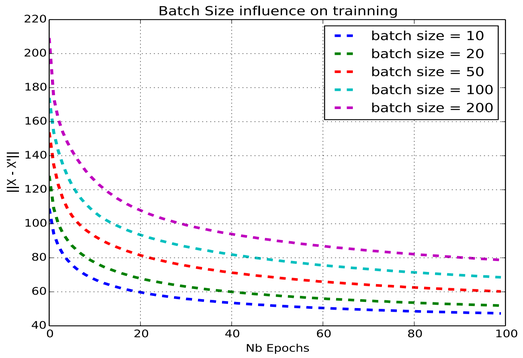


Figure 4: Reconstruction loss with 100 hidden units and varying batch size

Small batch size achieves smaller reconstruction loss.

Number of hidden units influence

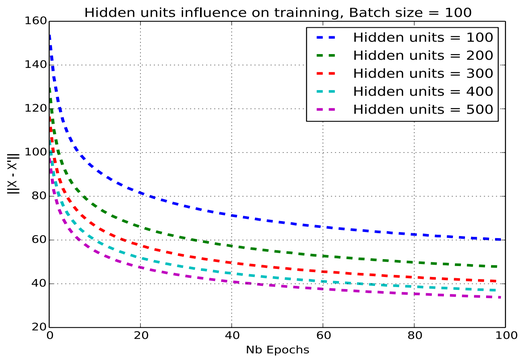


Figure 5: Reconstruction loss with varying number of hidden units

Increasing the number of hidden units improves the training performance but rises computation time, 250 hidden units is a good choice for MNIST and CIFAR data.

Conclusion and improvements

- The more steps we use for Gibbs sampling, the lower is the bias of our estimate. However, it takes more time to compute. An alternative of the Contrastive Divergence algorithm is called Persistent Contrastive Divergence: at each iteration of the algorithm, we initialize the Gibbs sampling with the sample from the previous iteration.
- It is possible to adapt the RBM model for unbounded reals by adding a quadratic term ($\frac{1}{2}\mathbf{x}^\top\mathbf{x}$) to the energy. Then, $p(\mathbf{x}|h)$ becomes a Gaussian distribution. This is called Gaussian-Bernoulli RBM. However, this model is less stable.