université
PARIS-SACLAY

Machines restreintes de Boltzmann

Travail présenté à Aurélien Decelle

réalisé par
Arnaud Gardille;

August 20, 2021

preface

Have you ever seen a person who doesn't exist? Well, it is now possible, thanks to generative models! check on `www.thispersondoesnotexist.com`.
Of course, these methods do not apply only to images, but also to music, text ... and indeed, any kind of data! Let's put aside all the potential societal issues raised by these techniques, and turn to one on the most famous generative model, the *restricted Boltzmann machine*.

About the author

At the end of my bachelor in fundamental and applied mathematics, Dr Aurélien Decelle suggested that I work on this subject during the summer. I was very excited to get the big picture of machine learning research and use my math skills. I had to work from home, because of the pandemic, which did not make my job any easier. However, this experience was very valuable to me. In particular, it highlighted the importance of the concepts I learned during the first year of my master's degree in "mathematics of artificial intelligence". For example, this introduced me to the maximum likelihood estimator in statistics, PCA (principal component analysis) and deep learning methodology. Also, this gave me a very positive insight of applied research, and motivated me to pursue my studies with a PhD.

Code

I coded my own RBM in python, which is available at `https://github.com/dechiktorren/RBM`

- The file *classe_RBM.py* contains my implementation of the diverse versions of the RBM. It is code in object-oriented style, in order to reduce the redundancy, and to permits easy comparisons. The optimisation of the computation through matrix product with numpy was crucial to the performance of the algorithm. In fact, gaussian RBM are very long to train, and this is where the ability to optimise your code becomes important.
- The file *ArtificialGaussianCluster.py* permits to generate artificial data to evaluate efficiently the gaussian RBM.
- The file *RBM_final.py* contains the most advanced version of the RBM (adaptive gaussian with annealing)

Contents

0.1 Notations

- $\mathbf{h} = \{h_1, ..., h_H\}$ the hidden layer
- $\mathbf{x} = \{v_1, ..., v_V\}$ the visible layer
- $\mathbf{b} = \{b_1, ..., b_H\}$ the hidden bias
- $\mathbf{c} = \{c_1, ..., c_V\}$ the visible bias
- $\mathbf{W} = \{W_{jk}\}$ the connections between the two layers, with W_{jk} the connection between the j^{th} hidden neuron, and the k^{th} visible neuron
- $x^{(1)}$, ..., $x^{(N)}$ the training examples
- $\widetilde{x}$ the approximation of $\mathbb{E}[x]$ according to the parameters. *(obtained by Gibbs sampling)*
- E the energy
- P for a probability
- θ for one of the parameters : **b, c** or **W**

0.2 Usual functions

- the probability density function of the Gaussian distribution :

$$\mathcal{N}\left(x|\mu,\sigma^2\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

- The Sigmoid function

$$\text{Sigm}(x) = \frac{1}{1+\exp(-x)}$$

- the Softplus function :

$$\text{softplus}(x) = log(1+\exp(x))$$

Introduction

Even if the first aim of deep learning was classification, it has provided some powerful generative models. Theses models are able to learn a probability distribution over multiples variables, and to draw samples of them. We will study and generalise one of the most famous, the *restricted Boltzmann machine*, developed in 1986 by Paul Smolensky. It consists in two layers of neurons :

- The **visible layer** gets an example, or gives a generated one.
- The **hidden layer** is used during the training to get a new representation of the data.

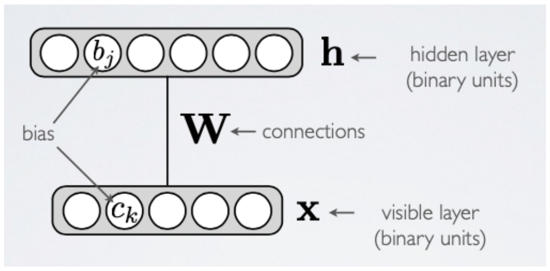
In order to simplify the model, a neuron isn't connected to the other neurons of it's layer, but is connected to every neuron of the other layer. This gives us the conditional Independence of each neuron, knowing the other layer.

The RBM is an energy based model, meaning we define the joint probability distribution of the neurons using an *energy function*:

$$p(x, h) = \frac{exp(-E(x, h))}{Z}$$

Where the energy function E may depends on both on the parameters and the states of the neurons. The *partition function* Z ensures that p is a probability:

$$Z = \sum_{x,h} \exp(-E(x, h))$$



Each neuron has a bias, noted b for visible neurons and c for hidden ones. A visible neuron x_i is connected to the hidden neuron h_j through the "weight" w_{ij}. As always with neural networks, the main idea is to adjust bias and weights so that the model fit to the data. In our case, we want to increase the probability of generating data statistically similar to a given data set.

We will begin by introducing the classical RBM, which works with binary data. Then, we'll generalise to gaussian distributions over the visible layer by modifying the energy function. We will conclude by some illustrations, and presentations over diverse optimisation methods useful for the training of RBMs.

1 The Restricted Boltzmann Machine

1.1 Definitions

For now, a neuron can only be activated (1) or deactivated (0). The stronger the signal it receives, the more likely it is to activate.

We define the **energy function** as :

$$\begin{aligned} E(x,h) &= -{}^t\boldsymbol{h}\boldsymbol{W}\boldsymbol{x} - {}^t\boldsymbol{c}\boldsymbol{x} - {}^t\boldsymbol{b}\boldsymbol{h} \\ &= -\sum_j \sum_k W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j \end{aligned}$$

1.2 Inference

In order to go back and forth between the two layers, we need to know the probability of a neuron knowing the other layer.

As the nodes of a layer aren't connected to each other, their distribution can be considered as independent. It makes the calculations of the conditional distributions much easier.

$$p(h|x) = \prod_j p(h_j|x)$$

$$p(x|h) = \prod_k p(x_k|h)$$

Proof. In order to prove that the Independence follows from the definition of the energy function, we just have to make some simple calculations :

$$\begin{aligned}
p(\mathbf{h}|\mathbf{x}) &= p(\mathbf{x},\mathbf{h}) / \sum_{\mathbf{h}'} p(\mathbf{x},\mathbf{h}') \\
&= \frac{\exp(\mathbf{h}^\top \mathbf{W}\mathbf{x} + \cancel{\mathbf{c}^\top \mathbf{x}} + \mathbf{b}^\top \mathbf{h}) / \cancel{Z}}{\sum_{\mathbf{h}' \in \{0,1\}^H} \exp(\mathbf{h}'^\top \mathbf{W}\mathbf{x} + \cancel{\mathbf{c}^\top \mathbf{x}} + \mathbf{b}^\top \mathbf{h}') / \cancel{Z}} \\
&= \frac{\exp(\sum_j h_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j)}{\sum_{h'_1 \in \{0,1\}} \cdots \sum_{h'_H \in \{0,1\}} \exp(\sum_j h'_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h'_j)} \\
&= \frac{\prod_j \exp(h_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j)}{\sum_{h'_1 \in \{0,1\}} \cdots \sum_{h'_H \in \{0,1\}} \prod_j \exp(h'_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h'_j)} \\
&= \frac{\prod_j \exp(h_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j)}{\left(\sum_{h'_1 \in \{0,1\}} \exp(h'_1 \mathbf{W}_{1\cdot}\mathbf{x} + b_1 h'_1)\right) \ldots \left(\sum_{h'_H \in \{0,1\}} \exp(h'_H \mathbf{W}_{H\cdot}\mathbf{x} + b_H h'_H)\right)} \\
&= \frac{\prod_j \exp(h_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j)}{\prod_j \left(\sum_{h'_j \in \{0,1\}} \exp(h'_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h'_j)\right)} \\
&= \frac{\prod_j \exp(h_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j)}{\prod_j (1 + \exp(b_j + \mathbf{W}_{j\cdot}\mathbf{x}))} \\
&= \prod_j \frac{\exp(h_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j)}{1 + \exp(b_j + \mathbf{W}_{j\cdot}\mathbf{x})} \\
&= \prod_j p(h_j|\mathbf{x})
\end{aligned}$$

□

We can deduce the activation probability of each neuron, knowing the other layer.

$$\begin{aligned}
p(h_j = 1|x) &= \frac{1}{1 + exp(-(b_j + W_j \cdot x))} \\
&= Sigm(b_j + W_j \cdot x)
\end{aligned}$$

with W_j the j^{th} row of W

$$\begin{aligned}
p(x_k = 1|h) &= \frac{1}{1 + exp(-(c_j + h \cdot W_k))} \\
&= Sigm(b_j + h \cdot W_k)
\end{aligned}$$

with W_k the k^{th} column of W

Proof. By definition of the sigmoid function, □

$$\begin{aligned} p(h_j = 1|\mathbf{x}) &= \frac{\exp(b_j + \mathbf{W}_{j\cdot}\mathbf{x})}{1 + \exp(b_j + \mathbf{W}_{j\cdot}\mathbf{x})} \\ &= \frac{1}{1 + \exp(-b_j - \mathbf{W}_{j\cdot}\mathbf{x})} \\ &= \mathrm{sigm}(b_j + \mathbf{W}_{j\cdot}\mathbf{x}) \end{aligned}$$

1.3 Free energy

What about p(x) ?

$$\begin{aligned} p(x) &= \sum_{h\in\{0,1\}^H} p(x,h) \\ &= \frac{1}{Z}\sum_{h\in\{0,1\}^H} \exp(-E(x,h)) \\ &= \frac{1}{Z} exp({}^t c \cdot x + \sum_{j=1}^{H} log(1 + exp(b_j + W_j \cdot x))) \\ &= \frac{1}{Z} exp(-F(x)) \end{aligned}$$

with $F = {}^t c \cdot x + \sum_{j=1}^{H} log(1 + exp(b_j + W_j \cdot x))$, the free energy

Proof. The proof is straightforward:

□

$$
\begin{aligned}
p(\mathbf{x}) &= \sum_{\mathbf{h}\in\{0,1\}^H} \exp(\mathbf{h}^\top \mathbf{W}\mathbf{x} + \mathbf{c}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h})/Z \\
&= \exp(\mathbf{c}^\top \mathbf{x}) \sum_{h_1\in\{0,1\}} \cdots \sum_{h_H\in\{0,1\}} \exp\left(\sum_j h_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j\right)/Z \\
&= \exp(\mathbf{c}^\top \mathbf{x}) \left(\sum_{h_1\in\{0,1\}} \exp(h_1 \mathbf{W}_{1\cdot}\mathbf{x} + b_1 h_1)\right) \cdots \left(\sum_{h_H\in\{0,1\}} \exp(h_H \mathbf{W}_{H\cdot}\mathbf{x} + b_H h_H)\right)/Z \\
&= \exp(\mathbf{c}^\top \mathbf{x})\,(1+\exp(b_1 + \mathbf{W}_{1\cdot}\mathbf{x}))\ldots(1+\exp(b_H + \mathbf{W}_{H\cdot}\mathbf{x}))/Z \\
&= \exp(\mathbf{c}^\top \mathbf{x})\exp(\log(1+\exp(b_1 + \mathbf{W}_{1\cdot}\mathbf{x})))\ldots\exp(\log(1+\exp(b_H + \mathbf{W}_{H\cdot}\mathbf{x})))/Z \\
&= \exp\left(\mathbf{c}^\top \mathbf{x} + \sum_{j=1}^{H} \log(1+\exp(b_j + \mathbf{W}_{j\cdot}\mathbf{x}))\right)/Z
\end{aligned}
$$

Finally,

$$p(x) = {}^t c \cdot x + \sum_{j=1}^{H} softplus(b_j + W_j \cdot x))$$

1.4 The training

How to adjust the parameters of the RBM in order to generate new data, which looks similar to the training sample?

As we know our density function, we will try to maximise the the likelihood of our examples. Thanks to the generalisation capability of our neural network, This will enable us to generates data similar to the training sample.

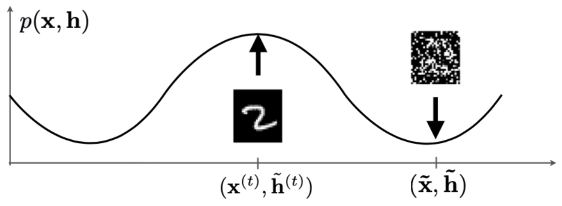
For computational convenience, we prefer to minimize the average negative log-likelihood (NLL), which is fully equivalent :

$$NLL(x^{(1)}, ..., x^{(T)}) = \frac{1}{T}\sum_{t=1}^{T} -\log(p(x^{(t)})$$

with $x^{(t)}$ an example.
We now have to describe the minimisation method we'll use, a famous variant of the gradient descent.

1.5 Stochastic gradient descent

If the training set is very large, a single pass on the data set can be very expensive. This is why we sometimes prefer to minimize by stochastic gradient descent.


$p(\mathbf{x},\mathbf{h})$
$(\mathbf{x}^{(t)},\tilde{\mathbf{h}}^{(t)})$
$(\tilde{\mathbf{x}},\tilde{\mathbf{h}})$

Definition 1.1. *(stochastic gradient descent)*
At each step, we choose a random example, Z_{i_t}, then we take a step in the direction of its gradient.

$$h_{t+1} = h_t - \alpha_t \nabla_h \ell(h_t, Z_{i_t})$$

with α_t strictly positive and decreasing.

By taking the expectation on i_t, which follows a uniform distribution on {1, ..., n }, we obtain $\mathbb{E}_{i_t}[\nabla_h \ell(h_t, Z_{i_t})] = \nabla_h R_S(h)$, which justifies the use of this optimization technique.

Now, let's remark that the negative log-likelihood can be split that way:

$$\begin{aligned} -log(p(x^{(t)})) &= -log\left(\frac{1}{Z}exp(-F(x^{(t)}))\right) \\ &= F(x) + log(Z) \end{aligned}$$

Let's compute the gradient with respect to θ, one of the parameter.

$$\frac{\partial - log(p(x^{(t)}))}{\partial\theta} = \frac{\partial F(x)}{\partial\theta} + \frac{\partial \log(Z)}{\partial\theta}$$

The equation can be rewritten with expectations.

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

with θ one of the parameters $W_{j,k}$, b_j or c_k.

Proof.

$$
\begin{aligned}
-\frac{\partial \log p(\boldsymbol{x})}{\partial \theta} &= -\frac{\partial \log \frac{e^{-F(\boldsymbol{x})}}{Z}}{\partial \boldsymbol{\theta}} && (1)\\
&= \frac{\partial (F(\boldsymbol{x}) + \log Z)}{\partial \boldsymbol{\theta}} && (2)\\
&= \frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{\theta}} + \frac{\partial \log Z}{\partial \boldsymbol{\theta}} && (3)\\
&= \frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{\theta}} + \frac{1}{Z}\frac{\partial \sum_{\boldsymbol{x}} e^{-F(\boldsymbol{x})}}{\partial \boldsymbol{\theta}} && (4)\\
&= \frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{\theta}} - \frac{1}{Z}\sum_{\boldsymbol{x}} e^{-F(\boldsymbol{x})}\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{\theta}} && (5)\\
&= \frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{\theta}} - \sum_{\boldsymbol{x}} p(\boldsymbol{x})\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{\theta}} && (1)\\
&= \frac{\partial - \log \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}}{\partial \boldsymbol{\theta}} - \sum_{\boldsymbol{x}} p(\boldsymbol{x})\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{\theta}} && (6)\\
&= \frac{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}\frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}}}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}} - \sum_{\boldsymbol{x}} p(\boldsymbol{x})\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{\theta}} && (7)\\
&= \sum_{\boldsymbol{h}} \frac{e^{-E(\boldsymbol{x},\boldsymbol{h})}}{\sum_{\boldsymbol{h}'} e^{-E(\boldsymbol{x},\boldsymbol{h}')}}\frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}} - \sum_{\boldsymbol{x}} p(\boldsymbol{x})\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{\theta}} && (8)\\
&= \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{x})\frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}} - \sum_{\boldsymbol{x},\boldsymbol{h}} p(\boldsymbol{x},\boldsymbol{h})\frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}} && (9)\\
&= \mathbb{E}_{\boldsymbol{h}|\boldsymbol{x}}[\frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}}] - \mathbb{E}_{\boldsymbol{x},\boldsymbol{h}}[\frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}}] && (2)\\
&= \text{positive phase contribution} - \text{negative phase contribution} && (10)
\end{aligned}
$$

□

The positive phase increases the probability of training data (by reducing the corresponding free energy), while the negative phase decreases the probability of samples generated by the model (by increasing the energy of all $x \sim P$).

The negative phase is intractable, and will have to be estimated. We will discuss that in the next section.
Hopefully, the positive phase can be compute explicitly :

$$\frac{\partial F(x^{(t)})}{\partial c_k} = x_k$$

$$\begin{aligned}
\frac{\partial F(x^{(t)})}{\partial b_j} &= \frac{exp(x^{(t)})}{1 + exp(b_j + W_j \cdot x^{(t)})} \\
&= Sigm(b_j + W_j \cdot x^{(t)}) \\
&= P(h_j = 1|x^{(t)})
\end{aligned}$$

$$\begin{aligned}
\frac{\partial F(x^{(t)})}{\partial W_{i,j}} &= \frac{x_i^{(t)} \cdot exp(x^{(t)})}{1 + exp(b_j + W_j \cdot x^{(t)})} \\
&= x_i^{(t)} \cdot P(h_j = 1|x^{(t)})
\end{aligned}$$

1.6 Contrastive divergence

In order to compute the negative phase, we will make some approximations using a Monte-Carlo approach. This means that we replace the expectation $\mathbb{E}_{x,h}\left[\frac{\partial E(x,h)}{\partial \theta}\right]$ by the mean of N point $\widetilde{x}$ samples according to the distribution of x. Obviously, this approximation makes sens because of the law of large numbers.

The algorithm we will obtain is called "Contrastive divergence".

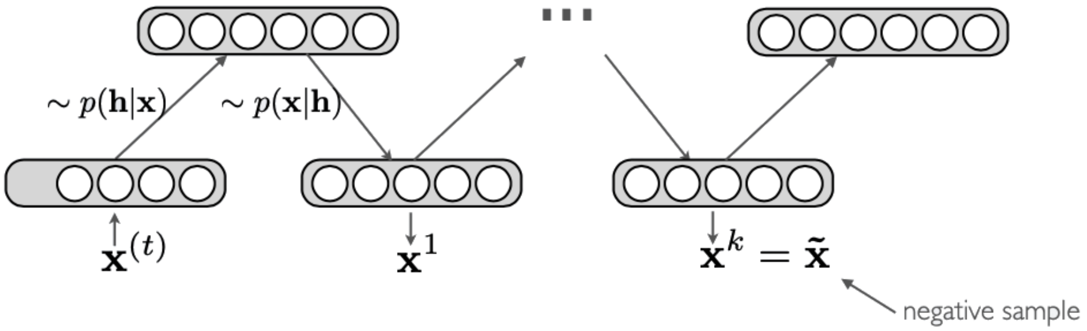
$$-\frac{\partial \log p(\boldsymbol{x})}{\partial \theta} \approx \frac{\partial F(\boldsymbol{x})}{\partial \theta} - \frac{1}{||N||} \sum_{\tilde{\boldsymbol{x}} \in N} \frac{\partial F(\tilde{\boldsymbol{x}})}{\partial \boldsymbol{\theta}}$$

In order to sample efficiently, it is necessary to do multiple pass between the layers. (without updating any weight, obviously)

The main practical question is where to start the sampling chain. What if we keep using the previous sample? There is big change the RBM will stay very near from it, and badly estimate the true distribution. And it's even worst if we start from an example. On the other hand, it takes much time to go from a random state to a likely state of the distribution.

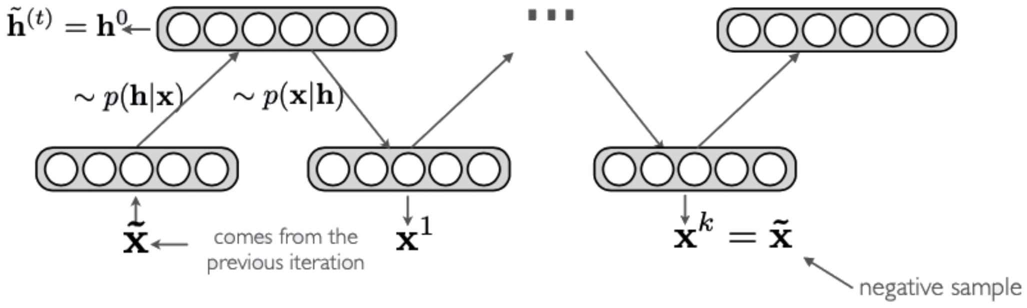
The good compromise is to run N sampling chains independently, and to keep using the previous sample for the next estimation. Then, we can hope the different clusters of our distributions will be taken into account. This is often refereed to as *persistent contrastive divergence*. As we will see in our later experimentation, there may be random favouritism between the different clusters of our distribution.

As the reader notice, we are into the domain of tricks, and we lack of strong theoretical results to clarify this situation.



$\sim p(\mathbf{h}|\mathbf{x})$
$\sim p(\mathbf{x}|\mathbf{h})$
$\mathbf{x}^{(t)}$
$\mathbf{x}^1$
$\mathbf{x}^k = \tilde{\mathbf{x}}$
negative sample

Figure 1: The original contrastive divergence, sampling one chain for many iterations starting from a random point.



$\tilde{\mathbf{h}}^{(t)} = \mathbf{h}^0$
$\sim p(\mathbf{h}|\mathbf{x})$
$\sim p(\mathbf{x}|\mathbf{h})$
$\tilde{\mathbf{x}}$
comes from the previous iteration
$\mathbf{x}^1$
$\mathbf{x}^k = \tilde{\mathbf{x}}$
negative sample

Figure 2: The persistent contrastive divergence, sampling multiple chains starting from the previous points.

We also make the following approximations :

$$\mathrm{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x}^{(t)},\mathbf{h})}{\partial\theta}\,\middle|\mathbf{x}^{(t)}\right] \approx \frac{\partial E(\mathbf{x}^{(t)},\tilde{\mathbf{h}}^{(t)})}{\partial\theta} \qquad \mathrm{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right] \approx \frac{\partial E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}{\partial\theta}$$

1.7 Learning rules

A step of stochastic gradient descent consist in updating our weights the following way:

$$W \Leftarrow W - \alpha \cdot (\nabla_w - log(p(x^{(t)})))$$

with $x^{(t)}$ one randomly selected example. But as we've seen before, this gradient is intractable, and must be partially estimated thanks to Gibbs sampling:

$$\begin{aligned}
\nabla_w - log(p(x^{(t)})) &= \mathbb{E}_h \left[\nabla_w E(x^{(t)}, h) | x^{(t)}\right] - \mathbb{E}_{x,h} \left[\nabla_w E(x, h)\right] \\
&= \mathbb{E}_h \left[\nabla_w E(x^{(t)}, h) | x^{(t)}\right] - \mathbb{E}_h \left[\nabla_w E(\widetilde{x}, h) | \widetilde{x}\right]
\end{aligned}$$

finally, the weight update simply becomes :

$$W \Leftarrow W - \alpha \cdot (h(x^{(t)}) \cdot x^{(t)} - h(\widetilde{x}) \cdot \widetilde{x})$$

Also, we will make k iterations of Gibbs sampling.
In general, the bigger k is, the less biased the estimate of the gradient will be.
This justify the updates in the final algorithm :

1. For each training example $x^{(t)}$
   - generate $\widetilde{x}$ using k steps of Gibbs sampling,starting at $x^{(t)}$
   - update parameters :

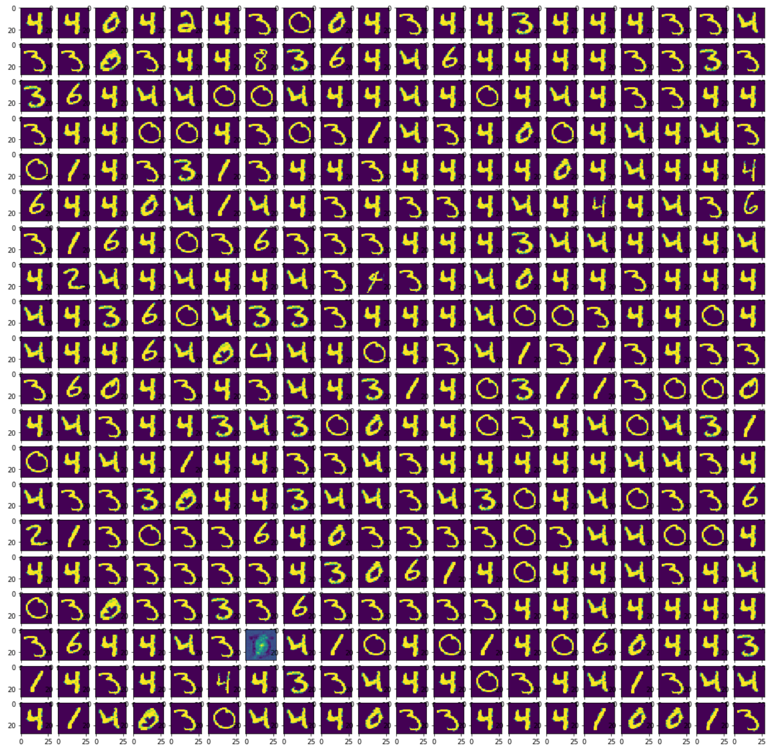
$$\begin{aligned}
W_{j,k} &\Leftarrow W_{j,k} + \alpha \cdot \left(h(x^{(t)})_j \cdot x_k^{(t)} - h(\widetilde{x})_j \cdot \widetilde{x}_k\right) \\
b_j &\Leftarrow b_j + \alpha \cdot \left(h(x^{(t)})_j - h(\widetilde{x})_j\right) \\
c_k &\Leftarrow c_k + \alpha \cdot (x_k^{(t)} - \widetilde{x}_k)
\end{aligned}$$

2. Go back to 1. until stopping criteria

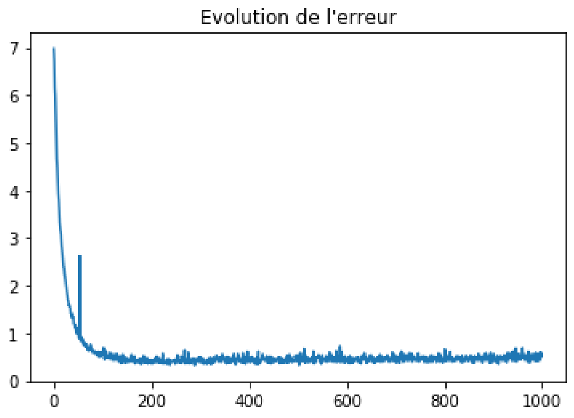
1.7.1 Tests on real data

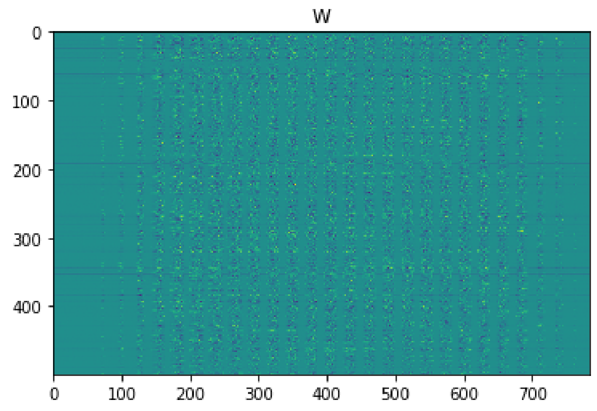
We are now willing to test this basic RBM on the famous MINST dataset. We use 100 examples, 1000 steps of training, a learning rate of 0.01, 784 visible neurons and 500 hidden neurons.

We generate 400 images after the training :



The image has grey levels, because we printed the probabilities of activation of the visible neurons, instead of a sample.
We notice that even if the model is able to generated new images, there is sometimes a lack of diversity.
We can observe on the next diagram that the evolution of the quadratic distance between the data and the generated images decreases up to a certain level. This levels represents the mean quadratic distance between the samples.





Let's now observe the weights matrix, on the right. The vertical lines on the weights matrix suggest that only a part of the pixel of the images are involved in the learning process. that's obviously the case, since the pixels on the border aren't so much used in the data set.

2 Real value visible unite

2.1 Usual energy function

In order to learn real valued data, we could be tempted to scale visible values to [0, 1], and to use the probabilities. However, it doesn't work that well. A way to improve that is to use Gaussian visible neurons.

In order to get gaussian visible neurons, we replace the previous Energy function with :

$$E(x,h|\theta) = \sum_k \frac{(x_k - c_k)^2}{2\sigma_k^2} - \sum_j \sum_k W_{j,k} h_j \frac{x_k}{\sigma_k} - \sum_j b_j h_j$$

Now, $x_k \in \mathbb{R}$ and $h_j \in \{0,1\}$

The conditional probabilities become :

$$p(x_k = y|h) = \mathcal{N}\left(v \mid c_k + \sigma_k \sum_j h_j W_{j,k} \ ; \sigma_k^2\right)$$

Proof :

$$\begin{aligned}
P(x|h) &= \frac{P(x,h)}{P(h)} \\
&= \frac{P(x,h)}{\sum_{x'} P(x',h)} \\
&= \frac{exp\left(-\sum_k \frac{(x_k - c_k)^2}{2\sigma_k^2} + \sum_j \sum_k W_{j,k} h_j \frac{x_k}{\sigma_k} + \sum_j b_j h_j\right)}{\sum_{x'} exp\left(-\sum_k \frac{(x_k^{'} - c_k)^2}{2\sigma_k^2} + \sum_j \sum_k W_{j,k} h_j \frac{x_k^{'}}{\sigma_k} + \sum_j b_j h_j\right)} \\
&= \frac{exp\left(-\sum_k \frac{x_k^2}{2\sigma_k^2} + 2\sum_k \frac{x_k \cdot c_k}{2\sigma_k^2} + \sum_j \sum_k W_{j,k} h_j \frac{x_k}{\sigma_k}\right)}{\sum_{x'} exp\left(-\sum_k \frac{(x_k^{'})^2}{2\sigma_k^2} + 2\sum_k \frac{x_k^{'} \cdot c_k}{2\sigma_k^2} + \sum_j \sum_k W_{j,k} h_j \frac{x_k^{'}}{\sigma_k}\right)}
\end{aligned}$$

Also,

$$p(h_j = 1|x) = Sigm\left(b_j + \sum_k W_{j,k} \frac{x_k}{\sigma_k}\right)$$

The partial derivative of the log likelihood with respect to each parameters gives us the gradient update rule.

$$
\begin{aligned}
W_{j,k} &\Leftarrow W_{j,k} + \frac{1}{\sigma_k} \cdot \left(h(x^{(t)})_j \cdot x_k^{(t)} - h(\widetilde{x})_j \cdot \widetilde{x}_k \right) \\
b_j &\Leftarrow b_j + \cdot \left(h(x^{(t)})_j - h(\widetilde{x})_j \right) \\
c_k &\Leftarrow c_k + \frac{1}{\sigma_k^2} \cdot (x_k^{(t)} - \widetilde{x}_k) \\
\sigma_k &\Leftarrow \sigma_k + \frac{1}{\sigma_k^3} \left(\left((x_k^{(t)} - c_k)^2 - (\widetilde{x}_k - c_k)^2 \right) - 2 \sum_j (x_k^{(t)} - \widetilde{x}_k) h_j W_{j,k} \right)
\end{aligned}
$$

However, the weights and visible bias are scaled by σ_i with a different exponents. This may cause some instabilities. We will correct that in the next section.

2.2 Modified energy function

We square the standard deviation in the second member of the previous Energy :

$$
E(x, h|\theta) = \sum_k \frac{(x_k - c_k)^2}{2\sigma_k^2} - \sum_j \sum_k W_{j,k} h_j \frac{x_k}{\sigma_k^2} - \sum_j b_j h_j
$$

The conditional probabilities change a bit :

$$
p(x_k = y|h) = \mathcal{N} \left(v \mid c_k + \sum_j h_j W_{j,k} \ ; \sigma_k^2 \right)
$$

$$
p(h_j = 1|x) = Sigm \left(b_j + \sum_k W_{j,k} \frac{x_k}{\sigma_k^2} \right)
$$

And the gradient update rules become :

$$
\begin{aligned}
W_{j,k} &\Leftarrow W_{j,k} + \frac{1}{\sigma_k^2} \cdot \left(h(x^{(t)})_j \cdot x_k^{(t)} - h(\widetilde{x})_j \cdot \widetilde{x}_k \right) \\
b_j &\Leftarrow b_j + \cdot \left(h(x^{(t)})_j - h(\widetilde{x})_j \right) \\
c_k &\Leftarrow c_k + \frac{1}{\sigma_k^2} \cdot (x_k^{(t)} - \widetilde{x}_k) \\
\sigma_k &\Leftarrow \sigma_k + \frac{1}{\sigma_k^3} \left(\left((x_k^{(t)} - c_k)^2 - (\widetilde{x}_k - c_k)^2 \right) - 2 \sum_j (x_k^{(t)} - \widetilde{x}_k) h_j W_{j,k} \right)
\end{aligned}
$$

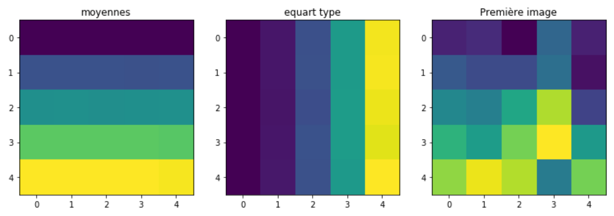
The gradient for both weights and visible bias are now scaled identically.

In the current model, σ is adapted globally. We would like it to be adapted to each cluster, which implies adding a second weight matrix inside a new energy function.

2.3 Comparison with the precedent model

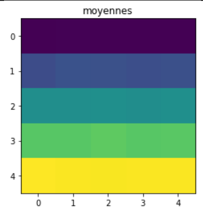
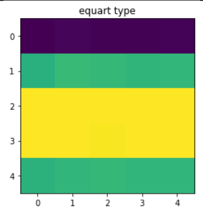
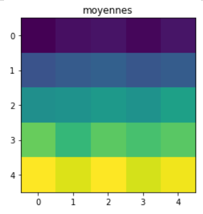
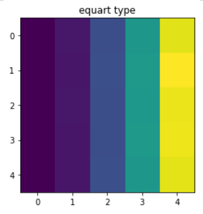
We want to test the ability of our models to learn both the mean and variance of our data. We begin by generating artificial data

The first picture represent the mean, the second the variance, and the third is the first generated image.



(I created 5000 images of 5*5 pixels. The mean goes from 0 to 4/5 from the first line to the last one. The squared scale goes from 0 to 12/50 from the first column to the last one.)

To test our models, we run 100 steps of training over all the images.

	Mean	Variance
Classical RBM		
Gaussian RBM		

The binary RBM learns the mean very well, but isn't able to learn the variance, when the gaussian RBM learns both efficiently.

3 Tunable variances

The following model can be used in order to tune the variance for each cluster found by the RBM

$$E(x,h|\theta) = \sum_k \frac{(x_k - c_k)^2}{2\sigma_k^2} - \sum_j \sum_k W^{(i)}_{j,k} h_j \frac{x_k}{\sigma_k^2} - \sum_j b_j h_j - \sum_j \sum_k W^{(ii)}_{j,k} h_j x_k^2 \tag{11}$$

Now we have two weight matrices : $W^{(i)}$ and $W^{(ii)}$.

3.1 Conditional distributions

The conditional probabilities for the hidden layer become:

$$P(h_j = 1|x) = Sigm\left(\sum_k W^{(i)}_{j,k} \frac{x_k}{\sigma_k^2} + b_j + \sum_k W^{(ii)}_{j,k} x_k^2\right) \tag{12}$$

Proof.

$$\begin{aligned}
P(h|x) &= \frac{P(x,h)}{P(x)} \\
&= \frac{P(x,h)}{\sum_{h'} P(x,h')} \\
&= \frac{exp\left(-\sum_k \frac{(x_k-c_k)^2}{2\sigma_k^2} + \sum_j \sum_k W^{(i)}_{j,k} h_j \frac{x_k}{\sigma_k^2} + \sum_j b_j h_j + \sum_j \sum_k W^{(ii)}_{j,k} h_j x_k^2\right)/Z}{\sum_{h'} exp\left(-\sum_k \frac{(x_k-c_k)^2}{2\sigma_k^2} + \sum_j \sum_k W^{(i)}_{j,k} h'_j \frac{x_k}{\sigma_k^2} + \sum_j b_j h'_j + \sum_j \sum_k W^{(ii)}_{j,k} h'_j x_k^2\right)/Z} \\
&= \frac{exp\left(\sum_j \sum_k W^{(i)}_{j,k} h_j \frac{x_k}{\sigma_k^2} + \sum_j b_j h_j + \sum_j \sum_k W^{(ii)}_{j,k} h_j x_k^2\right)}{\sum_{h'} exp\left(\sum_j \sum_k W^{(i)}_{j,k} h'_j \frac{x_k}{\sigma_k^2} + \sum_j b_j h'_j + \sum_j \sum_k W^{(ii)}_{j,k} h'_j x_k^2\right)} \\
&= \frac{\prod_j exp\left(\sum_k W^{(i)}_{j,k} h_j \frac{x_k}{\sigma_k^2} + b_j h_j + \sum_k W^{(ii)}_{j,k} h_j x_k^2\right)}{\sum_{h'_1\in\{0,1\}} \cdots \sum_{h'_H\in\{0,1\}} \prod_j exp\left(\sum_k W^{(i)}_{j,k} h'_j \frac{x_k}{\sigma_k^2} + b_j h'_j + \sum_k W^{(ii)}_{j,k} h'_j x_k^2\right)} \\
&= \frac{\prod_j exp\left(\sum_k W^{(i)}_{j,k} h_j \frac{x_k}{\sigma_k^2} + b_j h_j + \sum_k W^{(ii)}_{j,k} h_j x_k^2\right)}{\sum_{h'_1\in\{0,1\}} exp\left(\sum_k W^{(i)}_{j,k} h'_1 \frac{x_k}{\sigma_k^2} + b_j h'_1 + \sum_k W^{(ii)}_{j,k} h'_1 x_k^2\right) \cdots \sum_{h'_H\in\{0,1\}} exp\Bigg(\sum_k W^{(i)}_{j,k} h'_H \frac{x_k}{\sigma_k^2} + b_j h'_H + \sum} \\
&= \frac{\prod_j exp\left(\sum_k W^{(i)}_{j,k} h_j \frac{x_k}{\sigma_k^2} + b_j h_j + \sum_k W^{(ii)}_{j,k} h_j x_k^2\right)}{\prod_j \sum_{h'_j\in\{0,1\}} exp\left(\sum_k W^{(i)}_{j,k} h'_j \frac{x_k}{\sigma_k^2} + b_j h'_j + \sum_k W^{(ii)}_{j,k} h'_j x_k^2\right)} \\
&= \prod_j \frac{exp\left(\sum_k W^{(i)}_{j,k} h_j \frac{x_k}{\sigma_k^2} + b_j h_j + \sum_k W^{(ii)}_{j,k} h_j x_k^2\right)}{1 + exp\left(\sum_k W^{(i)}_{j,k} \frac{x_k}{\sigma_k^2} + b_j + \sum_k W^{(ii)}_{j,k} x_k^2\right)}
\end{aligned}$$

and $P(h|x) = \prod_j P(h_j|x)$
We get the desired equation by identification.

$\square$

The conditional probabilities for the visible layer become:

$$P(x_k|h) = \mathcal{N}\left(x_k \mid \mu_k, \tilde{\sigma}_k^2\right) \tag{13}$$

with :

$$\mu_k = \frac{\sum_j W_{j,k}^{(i)} h_j + c_k}{1 - 2\sigma_k^2 \sum_j W_{j,k}^{(ii)} h_j} \tag{14}$$

$$\tilde{\sigma}_k = \frac{\sigma_k}{\sqrt{1 - 2\sigma_k^2 \sum_j W_{j,k}^{(ii)} h_j}} \tag{15}$$

In order to prove this result, will make use of the following lemma:

Lemma 1. *value of the Gaussian integer*

$$\int exp\left(-ax^2 + bx + c\right) \mathrm{d}x = \sqrt{\frac{\pi}{a}} exp\left(\frac{b^2}{4a} + c\right)$$

Proof.

$$\begin{aligned}
P(x|h) &= \frac{P(x,h)}{P(h)} \\
&= \frac{P(x,h)}{\int P(y,h)\, dy} \\
&= \frac{exp\left(-\sum_k \frac{(x_k-c_k)^2}{2\sigma_k^2} + \sum_j \sum_k W_{j,k}^{(i)} h_j \frac{x_k}{\sigma_k^2} + \sum_j b_j h_j + \sum_j \sum_k W_{j,k}^{(ii)} h_j x_k^2\right)}{\int exp\left(-\sum_k \frac{(y_k-c_k)^2}{2\sigma_k^2} + \sum_j \sum_k W_{j,k}^{(i)} h_j \frac{y_k}{\sigma_k^2} + \sum_j b_j h_j + \sum_j \sum_k W_{j,k}^{(ii)} h_j (y_k)^2\right) dy} \\
&= \frac{exp\left(-\sum_k \frac{(x_k-c_k)^2}{2\sigma_k^2} + \sum_j \sum_k W_{j,k}^{(i)} h_j \frac{x_k}{\sigma_k^2} + \sum_j \sum_k W_{j,k}^{(ii)} h_j x_k^2\right)}{\int exp\left(-\sum_k \frac{(y_k-c_k)^2}{2\sigma_k^2} + \sum_j \sum_k W_{j,k}^{(i)} h_j \frac{y_k}{\sigma_k^2} + \sum_j \sum_k W_{j,k}^{(ii)} h_j (y_k)^2\right) dy} \\
&= \frac{\prod_k exp\left(-\frac{(x_k-c_k)^2}{2\sigma_k^2} + \sum_j W_{j,k}^{(i)} h_j \frac{x_k}{\sigma_k^2} + \sum_j W_{j,k}^{(ii)} h_j x_k^2\right)}{\int \prod_k exp\left(-\frac{(y_k-c_k)^2}{2\sigma_k^2} + \sum_j W_{j,k}^{(i)} h_j \frac{y_k}{\sigma_k^2} + \sum_j W_{j,k}^{(ii)} h_j (y_k)^2\right) dy} \\
&= \frac{\prod_k exp\left(-\frac{(x_k-c_k)^2}{2\sigma_k^2} + B_k \frac{x_k}{\sigma_k^2} + C_k x_k^2\right)}{\prod_k \underbrace{\int exp\left(-\frac{(y_k-c_k)^2}{2\sigma_k^2} + B_k \frac{y_k}{\sigma_k^2} + C_k y_k^2\right) dy_k}_{D_k}}
\end{aligned}$$

with

$$
\begin{aligned}
B_k &= \sum_j W^{(i)}_{j,k} h_j \\
C_k &= \sum_j W^{(ii)}_{j,k} h_j
\end{aligned}
$$

$$
\begin{aligned}
D_k &= \int exp\left(-\frac{(y_k - c_k)^2}{2\sigma_k^2} + B_k \frac{y_k}{\sigma_k^2} + C_k y_k^2\right) dy_k \\
&= \int exp\left(-\frac{c_k^2}{2\sigma_k^2} + (B_k + c_k)\frac{y_k}{\sigma_k^2} + \left(C_k - \frac{1}{2\sigma_k^2}\right) y_k^2\right) dy_k
\end{aligned}
$$

Thanks to the previous lemma, we obtain :

$$
\begin{aligned}
D_k &= \sqrt{\frac{\pi}{\left(\frac{1}{2\sigma_k^2} - C_k\right)}} exp\left(\frac{\left(\frac{(B_k+c_k)}{\sigma_k^2}\right)^2}{4\left(\frac{1}{2\sigma_k^2} - C_k\right)} - \frac{c_k^2}{2\sigma_k^2}\right) \\
&= \sqrt{2\pi}\frac{\sigma_k}{\sqrt{1 - 2\sigma_k^2 C_k}} exp\left(\frac{(B_k + c_k)^2}{2\sigma_k^2\left(1 - 2\sigma_k^2 C_k\right)} - \frac{c_k^2}{2\sigma_k^2}\right) \\
&= \sqrt{2\pi}\frac{\sigma_k}{\sqrt{1 - 2\sigma_k^2 C_k}} exp\left(\frac{B_k^2 + 2B_k c_k + c_k^2}{2\sigma_k^2\left(1 - 2\sigma_k^2 C_k\right)} - \frac{c_k^2}{2\sigma_k^2}\right) \\
&= \sqrt{2\pi}\frac{\sigma_k}{\sqrt{1 - 2\sigma_k^2 C_k}} exp\left(\frac{B_k^2 + 2B_k c_k + 2\sigma_k^2 C_k c_k^2}{2\sigma_k^2\left(1 - 2\sigma_k^2 C_k\right)}\right)
\end{aligned}
$$

let's note $\tilde{\sigma} = \frac{\sigma_k}{\sqrt{1-2\sigma_k^2 C_k}}$

$$\begin{aligned}
P(x|h) &= \prod_k \frac{1}{\tilde{\sigma}\sqrt{2\pi}} \cdot exp\left(-\frac{(x_k - c_k)^2}{2\sigma_k^2} + B_k \frac{x_k}{\sigma_k^2} + C_k x_k^2 - \frac{B_k^2 + 2B_k c_k + 2\sigma_k^2 C_k c_k^2}{2\sigma_k^2\,(1 - 2\sigma_k^2 C_k)}\right) \\
&= \prod_k \frac{1}{\tilde{\sigma}\sqrt{2\pi}} \cdot exp\left(-\frac{1}{2} \cdot \left(\frac{x_k^2 - 2x_k c_k + c_k^2}{\sigma_k^2} - 2B_k \frac{x_k}{\sigma_k^2} - 2C_k x_k^2 + \frac{B_k^2 + 2B_k c_k + 2\sigma_k^2 C_k c_k^2}{\sigma_k^2(1 - 2\sigma_k^2 C_k)}\right)\right) \\
&= \prod_k \frac{1}{\tilde{\sigma}\sqrt{2\pi}} \cdot exp\left(-\frac{1}{2} \cdot \left(\frac{x_k^2(1 - 2C_k\sigma^2)}{\sigma_k^2} - 2(B_k + c_k)\frac{x_k}{\sigma_k^2} + \frac{B_k^2 + 2B_k c_k + c_k^2}{\sigma_k^2(1 - 2\sigma_k^2 C_k)}\right)\right) \\
&= \prod_k \frac{1}{\tilde{\sigma}\sqrt{2\pi}} \cdot exp\left(-\frac{1}{2} \cdot \left(\frac{x_k^2}{\tilde{\sigma}^2} - 2(B_k + c_k)\frac{x_k}{\sigma_k^2} + \frac{(B_k + c_k)^2}{\sigma_k^2(1 - 2\sigma_k^2 C_k)}\right)\right) \\
&= \prod_k \frac{1}{\tilde{\sigma}\sqrt{2\pi}} \cdot exp\left(-\frac{1}{2} \cdot \left(\left(\frac{x_k}{\tilde{\sigma}} - \tilde{\sigma}\frac{B_k + c_k}{\sigma_k^2}\right)^2 - \tilde{\sigma}^2\frac{(B_k + c_k)^2}{\sigma_k^4} + \frac{(B_k + c_k)^2}{\sigma_k^2(1 - 2\sigma_k^2 C_k)}\right)\right) \\
&= \prod_k \frac{1}{\tilde{\sigma}\sqrt{2\pi}} \cdot exp\left(-\frac{1}{2} \cdot \left(\left(\frac{x_k}{\tilde{\sigma}} - \tilde{\sigma}\frac{B_k + c_k}{\sigma_k^2}\right)^2\right)\right) \\
&= \prod_k \frac{1}{\tilde{\sigma}\sqrt{2\pi}} \cdot exp\left(-\frac{1}{2} \cdot \left(\frac{x_k - \mu}{\tilde{\sigma}}\right)^2\right)
\end{aligned}$$

with $\mu = \tilde{\sigma}^2 \frac{B_k + c_k}{\sigma_k^2} = \frac{B_k + c_k}{1 - 2\sigma_k^2 C_k}$

as $P(x|h) = \prod_i P(x_i|h)$, we get the desired equation by identification. □

We now are able to generate point with our model. But we need to do some more calculations in order to train it with a SGD on the negative log-likelihood.

3.2 parameter's update

Now let's find the parameter's update, by computing the positive part of the gradient of the log likelihood.

Remember that :

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

With our current energy function, we obtain :

$$\mathbb{E}_h\left[\frac{\partial - E(x^{(t)}, h, \theta)}{\partial W_{j,k}^{(i)}}|x\right] = \frac{x_k}{\sigma_k^2} \cdot P(h_j = 1|v, \theta)$$

$$\mathbb{E}_h\left[\frac{\partial - E(x^{(t)},h,\theta)}{\partial W^{(ii)}_{j,k}}|x\right] = x_k^2 \cdot P(h_j=1|v,\theta)$$

$$\mathbb{E}_h\left[\frac{\partial - E(x^{(t)},h,\theta)}{\partial c_k}|x\right] = \frac{x_k - c_k}{\sigma_k^2}$$

$$\mathbb{E}_h\left[\frac{\partial - E(x^{(t)},h,\theta)}{\partial b_j}|x\right] = P(h_j=1|v,\theta)$$

$$\mathbb{E}_h\left[\frac{\partial - E(x^{(t)},h,\theta)}{\partial \sigma_k}|x\right] = \frac{1}{\sigma_k^3}\left((x_k-c_k)^2 - 2x_k\sum_j W^{(i)}_{j,k}P(h_j=1|x,\theta)\right)$$

which leads to the following approximate gradient of the log likelihood:

$$\frac{\partial}{\partial W^{(i)}_{j,k}}log(p(x^{(t)}|\theta) \approx \frac{1}{\sigma_k^2}\cdot\left(P(h_j=1|x^{(t)},\theta)\cdot x_k^{(t)} - P(h_j=1|\tilde{x},\theta)\tilde{x}_k\right)$$

$$\frac{\partial}{\partial W^{(ii)}_{j,k}}log(p(x^{(t)}|\theta) \approx P(h_j=1|x^{(t)},\theta)\cdot(x^{(t)})^2 - P(h_j=1|\tilde{x},\theta)\cdot\tilde{x}^2$$

$$\frac{\partial}{\partial c_k}log(p(x^{(t)}|\theta) \approx \frac{1}{\sigma_k^2}(x_k^{(t)} - \tilde{x}_k)$$

$$\frac{\partial}{\partial b_j}log(p(x^{(t)}|\theta) \approx P(h_j=1|x^{(t)},\theta) - P(h_j=1|\tilde{x},\theta)$$

$$\begin{aligned}\frac{\partial}{\partial \sigma_k}log(p(x^{(t)}|\theta) \approx \frac{1}{\sigma_k^3}\left((x_k^{(t)}-c_k)^2 - (\tilde{x_k}-c_k)^2\right) - \\ \frac{2}{\sigma_k^3}\left(x_k^{(t)}\sum_j W^{(i)}_{j,k}P(h_j=1|x^{(t)},\theta) - \tilde{x_k}\sum_j W^{(i)}_{j,k}P(h_j=1|\tilde{x},\theta)\right)\end{aligned}$$

Proof.

$$\begin{aligned}
\mathbb{E}_h\left[\frac{\partial - E(x^{(t)}, h, \theta)}{\partial W^{(i)}_{j,k}}|x\right] &= \sum_h P(h|x,\theta)\cdot\frac{\partial}{\partial W^{(i)}_{j,k}}(-E(v,h,\theta))\\
&= \sum_h P(h|x,\theta)\cdot h_j\frac{x_k}{\sigma_k^2}\\
&= \frac{x_k}{\sigma_k^2}\sum_h P(h|x,\theta)\cdot h_j\\
&= \frac{x_k}{\sigma_k^2}\cdot P(h_j=1|v,\theta)
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}_h\left[\frac{\partial - E(x^{(t)}, h, \theta)}{\partial W^{(ii)}_{j,k}}|x\right] &= \sum_h P(h|x,\theta)\cdot\frac{\partial}{\partial W^{(ii)}_{j,k}}(-E(v,h,\theta))\\
&= \sum_h P(h|x,\theta)\cdot h_j x_k^2\\
&= x_k^2\sum_h P(h|x,\theta)\cdot h_j\\
&= x_k^2\cdot P(h_j=1|v,\theta)
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}_h\left[\frac{\partial - E(x^{(t)}, h, \theta)}{\partial c_k}|x\right] &= \sum_h P(h|x,\theta)\cdot\frac{\partial}{\partial c_k}(-E(v,h,\theta))\\
&= \sum_h P(h|x,\theta)\cdot\frac{x_k-c_k}{\sigma_k^2}\\
&= \frac{x_k-c_k}{\sigma_k^2}\cdot\sum_h P(h|x,\theta)\\
&= \frac{x_k-c_k}{\sigma_k^2}
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}_h\left[\frac{\partial - E(x^{(t)}, h, \theta)}{\partial b_j}|x\right] &= \sum_h P(h|x,\theta)\cdot\frac{\partial}{\partial b_j}(-E(v,h,\theta))\\
&= \sum_h P(h|x,\theta)\cdot h_j\\
&= P(h_j=1|v,\theta)
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}_h\left[\frac{\partial - E(x^{(t)},h,\theta)}{\partial \sigma_k}|x\right] &= \sum_h P(h|x,\theta)\cdot\left(\frac{(x_k-c_k)^2}{\sigma_k^3} - 2\sum_j W^{(i)}_{j,k} h_j \frac{x_k}{\sigma_k^3}\right)\\
&= \frac{1}{\sigma_k^3}\sum_h P(h|x,\theta)\cdot\left((x_k-c_k)^2 - 2\sum_j W^{(i)}_{j,k} h_j x_k\right)\\
&= \frac{1}{\sigma_k^3}\left((x_k-c_k)^2 - 2x_k\sum_j W^{(i)}_{j,k}\sum_h P(h|x,\theta)\cdot h_j\right)\\
&= \frac{1}{\sigma_k^3}\left((x_k-c_k)^2 - 2x_k\sum_j W^{(i)}_{j,k} P(h_j=1|x,\theta)\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial W^{(i)}_{j,k}} log(p(x^{(t)}|\theta) &= \frac{x_k}{\sigma_k^2}\cdot P(h_j=1|x^{(t)},\theta) - \mathbb{E}_x\left[\frac{x_k}{\sigma_k^2}\cdot P(h_j=1|v,\theta)\right]\\
&\approx \frac{1}{\sigma_k^2}\cdot\Big(P(h_j=1|x^{(t)},\theta)\cdot x_k^{(t)} - P(h_j=1|\tilde{x},\theta)\tilde{x}_k\Big)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial W^{(ii)}_{j,k}} log(p(x^{(t)}|\theta) &= (x_k^{(t)})^2\cdot P(h_j=1|x^{(t)},\theta) - \mathbb{E}_x\left[x_k^2\cdot P(h_j=1|x,\theta)\right]\\
&\approx P(h_j=1|x^{(t)},\theta)\cdot(x^{(t)})^2 - P(h_j=1|\tilde{x},\theta)\cdot\tilde{x}^2
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial c_k} log(p(x^{(t)}|\theta) &= \frac{x_k^{(t)}-c_k}{\sigma_k^2} - \mathbb{E}_x\left[\frac{x_k-c_k}{\sigma_k^2}|v,\theta)\right]\\
&= \frac{x_k^{(t)}}{\sigma_k^2} - \mathbb{E}_x\left[\frac{x_k}{\sigma_k^2}|v,\theta)\right]\\
&\approx \frac{1}{\sigma_k^2}(x_k^{(t)} - \tilde{x}_k)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial b_j} log(p(x^{(t)}|\theta) &= P(h_j=1|x^{(t)},\theta) - \mathbb{E}_x\left[P(h_j=1|x,\theta)|x,\theta)\right]\\
&\approx P(h_j=1|x^{(t)},\theta) - P(h_j=1|\tilde{x},\theta)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial \sigma_k} log(p(x^{(t)}|\theta) &= \frac{1}{\sigma_k^3}\left((x_k^{(t)} - c_k)^2 - 2x_k^{(t)}\sum_j W_{j,k}^{(i)} P(h_j = 1|x^{(t)}, \theta)\right) \\
&- \frac{1}{\sigma_k^3}\mathbb{E}_x\left[(x_k - c_k)^2 - 2x_k \sum_j W_{j,k}^{(i)} P(h_j = 1|x, \theta)\right] \\
&\approx \frac{1}{\sigma_k^3}\left((x_k^{(t)} - c_k)^2 - (\tilde{x_k} - c_k)^2\right) \\
&- \frac{2}{\sigma_k^3}\left(x_k^{(t)}\sum_j W_{j,k}^{(i)} P(h_j = 1|x^{(t)}, \theta) - \tilde{x_k}\sum_j W_{j,k}^{(i)} P(h_j = 1|\tilde{x}, \theta)\right)
\end{aligned}$$

□

3.3 Fast increasing variance

Experimentally, σ tends to increase very fast. To solve that problem, we replace σ_k^2 by $exp(z_k)$. The energy becomes :

$$E(x, h|\theta) = \sum_k \frac{(x_k - c_k)^2}{2exp(z_k)} - \sum_j\sum_k W_{j,k}^{(i)} h_j \frac{x_k}{exp(z_k)} - \sum_j b_j h_j - \sum_j\sum_k W_{j,k}^{(ii)} h_j x_k^2 \qquad (16)$$

And we get the following derivatives, all other remaining unchanged:

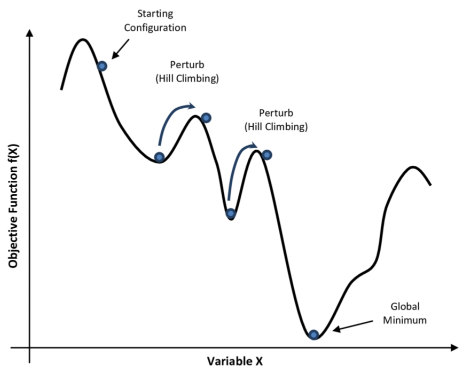
$$\begin{aligned}
\mathbb{E}_h\left[\frac{\partial - E(x^{(t)}, h, \theta)}{\partial z_k}|x\right] &= \frac{1}{exp(z_k)}\sum_h P(h|x, \theta)\cdot\left(\frac{1}{2}(x_k - c_k)^2 - x_k\sum_j W_{j,k}^{(i)} h_j\right) \\
&= \frac{1}{exp(z_k)}\cdot\left(\frac{1}{2}(x_k - c_k)^2 - x_k\sum_j W_{j,k}^{(i)}\sum_h h_j P(h|x, \theta)\right) \\
&= \frac{1}{exp(z_k)}\cdot\left(\frac{1}{2}(x_k - c_k)^2 - x_k\sum_j W_{j,k}^{(i)} P(h_j = 1|x, \theta)\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial z_k} log(p(x^{(t)}|\theta) &= \frac{1}{exp(z_k)}\cdot\left(\frac{1}{2}((x_k^{(t)} - c_k)^2 - (\tilde{x}_k - c_k)^2\right) \\
&- \left(x_k^{(t)}\sum_j W_{j,k}^{(i)} P(h_j = 1|x^{(t)}, \theta) - \tilde{x}_k\sum_j W_{j,k}^{(i)} P(h_j = 1|\tilde{x}, \theta))\right)
\end{aligned}$$

Our model becomes increasingly complex, and through, harder to train. We would like to maximise more efficiently the likelihood of our data. And solution is to use a common heuristic, annealing.

3.4 Annealing

Let's suppose that you're throwing a ball down a hill, and you hope it'll fall as low as possible. The ball could end in a local minimum. To avoid that, we'll give the ball a "temperature", so that it rarely moves randomly. This heuristic helps to find the global minimum, or at least a lower one.



Now let's see how to implement that in practice. Let's define an inverse temperature $\beta = 1/T$. We can transform the probability as

$$p(x, h; \beta) = \frac{1}{Z_\beta} \exp(-\beta E(x, h)) \tag{17}$$

and $Z_\beta = \sum_{x,h} p(x, h; \beta)$. When $\beta = 1$, we are dealing with our current system. When $\beta \to 0$, we are converging toward a system of independent variables. We know that there is a phase transition, when $s^* = \max(s_\alpha) > 1/4$:

$$m_i^{(v)} = \sum_j w_{ij} m_j^{(h)} \tag{18}$$

$$m_j^{(h)} = \text{sigmoid}(\sum_i w_{ij} m_i^{(v)}) \tag{19}$$

when w_{ij} is small, $\text{sigmoid}(\sum_i w_{ij} m_i^{(v)}) \approx 1/2 + 1/4 \sum_i w_{ij} m_i^{(v)}$. The idea to samples configurations is to do the following. First, we have to find β_{init} such that $\beta_{init} s^* = 1/4$. Then, if we do n_{MC} steps, we adjust the value of β at each iteration, such that, when $t = 1$, $\beta = \beta_{init}$ and when $t = n_{MC}$, $\beta = 1$.
We now have :

$$P(h_j = 1|x) = Sigm\left(\beta\left(\sum_k W_{j,k}^{(i)} \frac{x_k}{\sigma_k^2} + b_j + \sum_k W_{j,k}^{(ii)} x_k^2\right)\right) \tag{20}$$

$$\begin{aligned} P(x_k|h) &= \mathcal{N}\left(\beta \cdot x_k \,|\, \beta \cdot \mu_k, \beta \tilde{\cdot} \sigma_k^2\right) \\ &= \prod_k \frac{1}{\tilde{\sigma}\sqrt{2\beta\pi}} \cdot \exp\left(-\frac{1}{2}\beta \cdot \left(\frac{x_k - \mu}{\tilde{\sigma}}\right)^2\right) \end{aligned}$$

with :

$$\mu_k = \frac{\sum_j W^{(i)}_{j,k} h_j + c_k}{1 - 2\sum_j W^{(ii)}_{j,k} h_j} \tag{21}$$

$$\tilde{\sigma}_k = \frac{1}{\sqrt{1 - \sum_j W^{(ii)}_{j,k} h_j}} \tag{22}$$

3.5 Experimentation

3.5.1 Artificial Gaussian cluster

We create artificial data in order to evaluate our model. So we generate many gaussian clusters, where each cluster has a given mean and variance.

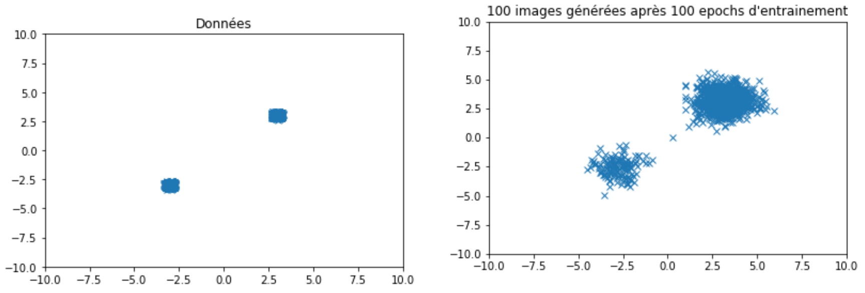
Here's how I did it in my examples :

I chose to generate $N_g = 2$ gaussians. The center of each gaussian belongs to a hyper-cube of $D = 10$ dimensions, and of length $L = 10$.

It takes more points to correctly estimate the variance of a gaussian cluster when this last one is high. This explains why I generated M data using the defined gaussian cluster. We define the ratio signal over noise as $\sigma_i/\sqrt{M_i} = cst$ *(where σ_i is the variance and M_i the number of data for the cluster number i)*. We need $\sigma_i/\sqrt{M_i}$ to be the same for all clusters. We have that $\sum_i M_i = M$.

Then, we have to center the data set such that the mean over all the data is zero. This helps the RBM, since it doesn't have to learn the center of the data.

First, I tried with 2 dimensions, 10 hidden neurons, and a learning rate: 10e-5. We have 5000 points for each clusters, and the variances are the same We can observe the accumulated data on the left, and accumulated samples produced after the training on the right.



Note that the cluster on the right always catches more points. Let's try in a space of higher dimension. But then, a question arise. . .

3.5.2 How to visualize ?

The RBM normally learns "interesting" dimensions whereto project the data set. We can use the weight matrix to find those new directions using the SVD (singular values decomposition). Basically, we can write

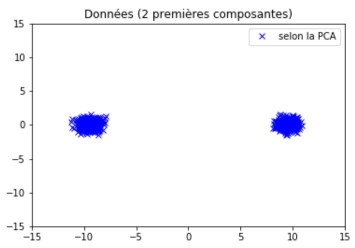
$$w_{ij} = \sum_{\alpha} u_i^{\alpha} s_{\alpha} v_j^{\alpha}$$

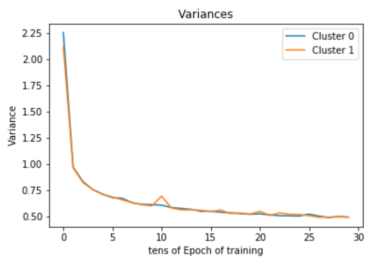
Let's assume that the index j runs over the visible nodes. We can project the data set over the matrix $\boldsymbol{v}$ (assuming $X \in \mathbb{R}^{M \times D}$, $\boldsymbol{v} \in \mathbb{R}^{D \times N_{hidden}}$)

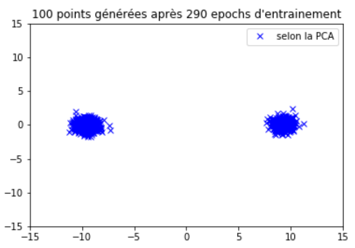
$$\boldsymbol{X}^{v} = \boldsymbol{X}\boldsymbol{v}$$

3.5.3 With only 2 similar clusters

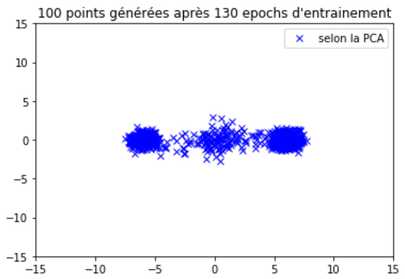
Ng = 2 ; D = 10 ; L = 10 ; learning rate = 10e-4 ; $\sigma = (0.5, 0.5)$ *We have 2500 examples for each cluster, consisting of a single point.*



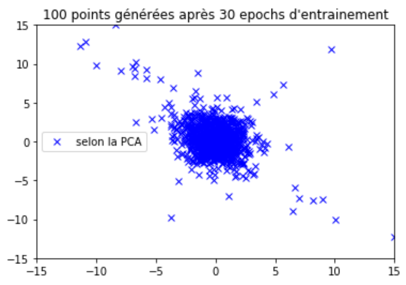




After 290 epochs of training, the estimated variances are 0.499 and 0.497.
Experimentally, the number of hidden neurons must be at least equivalent to the dimension of the data. Otherwise, another cluster appears between our two wished cluster, as we can see on the image below. *D = 100 , nb of HN = 10*



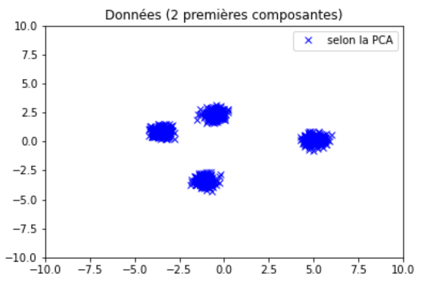
On the other hand, too many hidden neurons leads to useless research from the RBM. The system is struggling to stabilize. *D = 10 , nb of HN = 100*



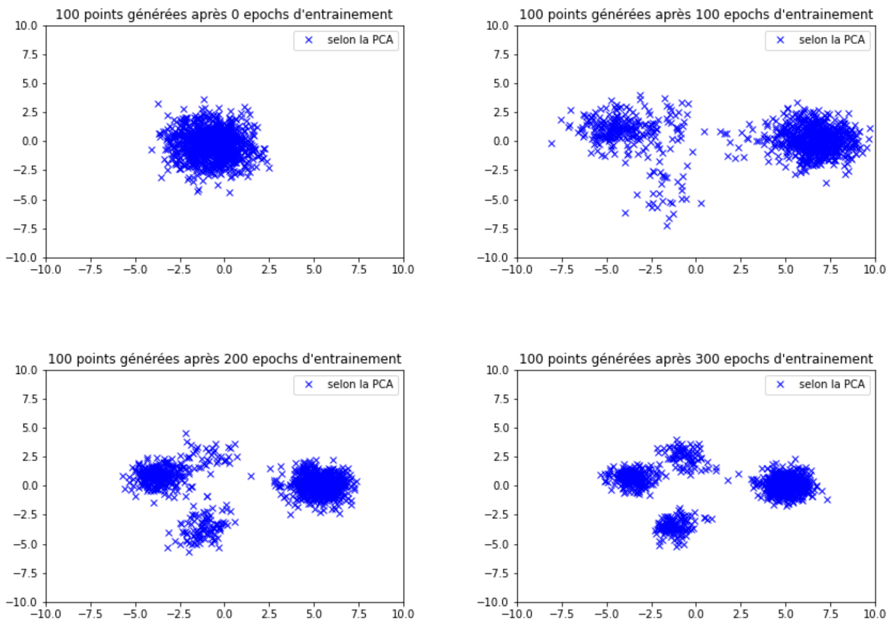
But if the initial conditions are good, the system still converges.

3.5.4 With 4 clusters

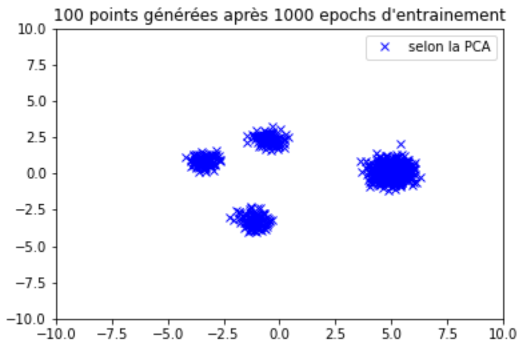
Learning rate : 10-6 , initial variances : 0.3 Here are the accumulated data :

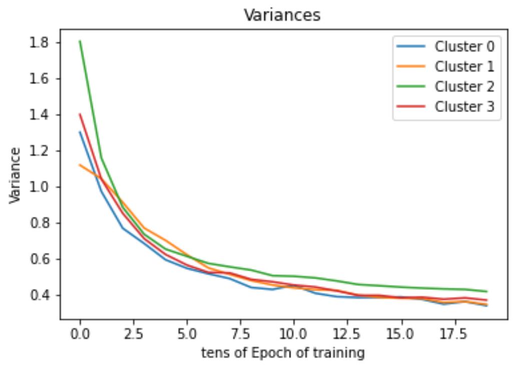


You can observe the convergence by yourself.

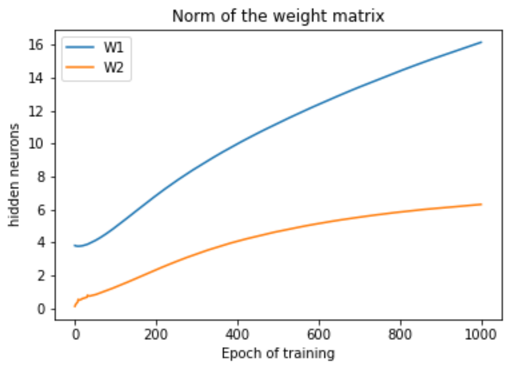


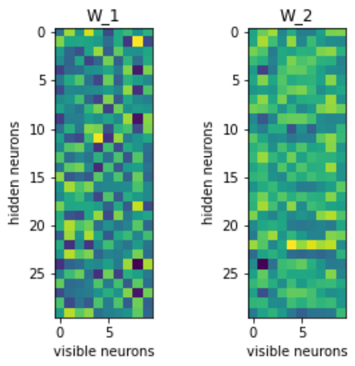
And at the end, we obtain this sample, and the following evolution of the variances.





One cluster catches more points, but the results remains very good.
We observe that the norms of the weight matrix increase through the learning process.

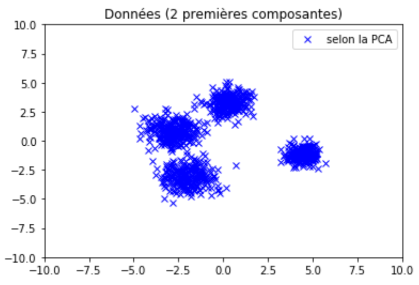




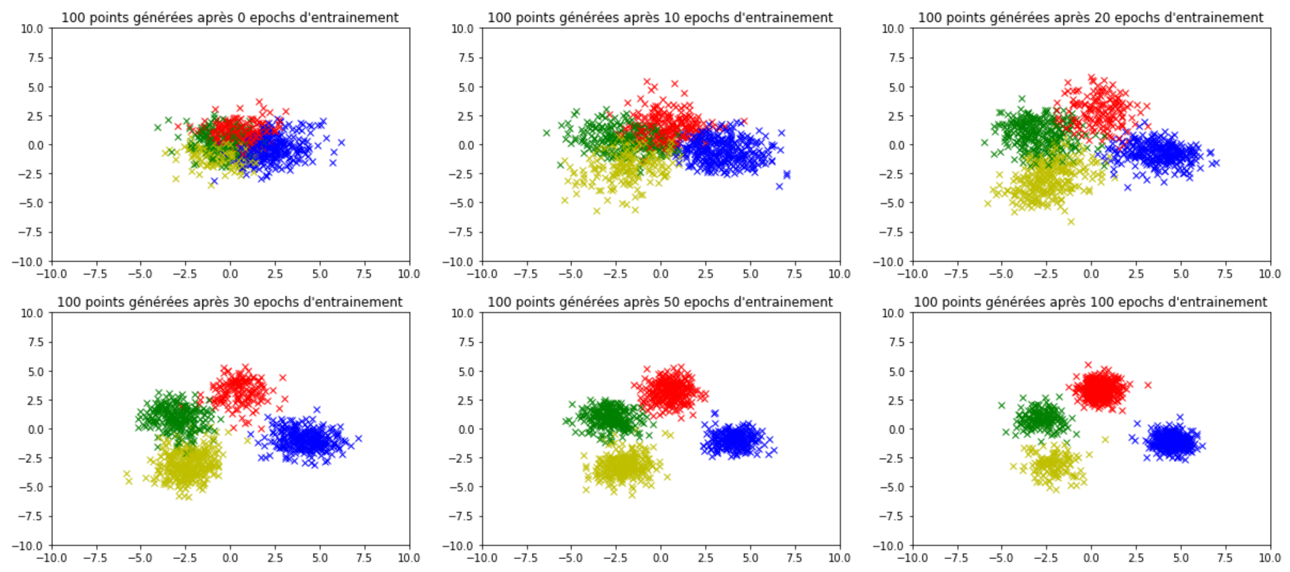
As the matrix are both used to determine the mean and variance of the clusters, it is hard to give an interpretation.

3.5.5 Clusters with different variances

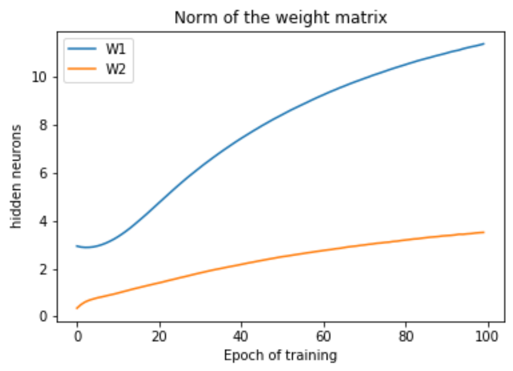
Here are the parameters I used:
Number of gaussians = 4; Dimention = 10; length = 10, 2500 examples of each cluster.
nbEpoch = 100; nbHidden = 10; learning rate = 1e-5
variances of the clusters : [0.5, 0.6, 0.7, 0.8] Here is the accumulation of the initial data:

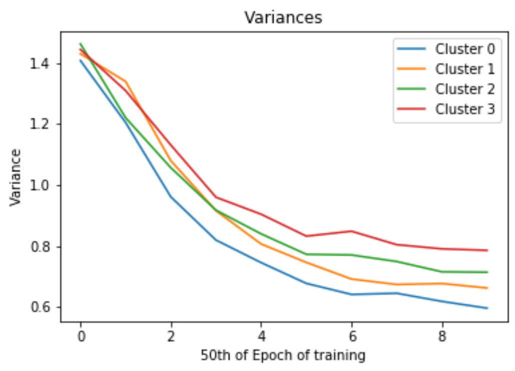


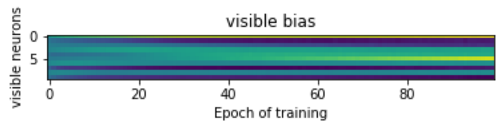
In only 100 epochs, the estimated variances are [0.60, 0.66, 0.71, 0.79], which is quite impressive.

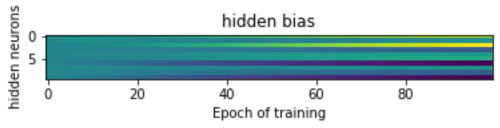


Here are the evolution of the bias, and norms of the weights.









4 Conclusion

The generalisation of the RBM trough the modification of it's energy function went one quite well. We successfully derived the conditional probability distribution and the updates needed for the learning. However, there is still much to be done, and some of the suggested track to pursue this article are the following:

- To try to dissociate the roles of the matrices. I tried, but encountered huge mathematical complications. (see in the annex)
- To ensure that clusters catches a number of point proportional to the data. It even happens that a cluster gets completely ignored by the RBM.
- To stabilise the learning process: the learning rate is hard to calibrate. We easily get overflows when it is high, or a very slow learning process in the opposite case.

I made use of the following resources : Most of the proofs of section 1 are taken from [4], which is a series of good videos on machine learning. This was a helpful introduction, as well as [2]. Also, [1] and [3] were useful resources to understand how to train a gaussian RBM.

I hope you enjoy the reading of this report. If you have any question about my work, don't hesitate to contact me by mail. Thanks again to Mr Decelle for his supervision of this paper.

Arnaud Gardille

Bibliography

[1] KyungHyun Cho. *Improved Learning Algorithms for Restricted Boltzmann Machines*. PhD thesis, 2011.

[2] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. 2010.

[3] Laurenz Wiskott Jan Melchior, Nan Wang. Gaussian-binary restricted boltzmann machines for modeling natural image statistics. 2017.

[4] Hugo Larochelle. Neural networks - restricted boltzmann machine. 2014.