Probabilistic Graphical Models, Master MVA, 2017/2018

Paper review: Deep Boltzmann Machines, R. Salakhutdinov, G. Hinton (2009)

Nidham GAZAGNADOU, Pirashanth RATNAMOGAN , Othmane SAYEM

gazagnadou@ensta.fr, ratnamogan@ensta.fr, sayem@ensta.fr

Introduction

Boltzmann Machines (BM) are a class of stochastic recurrent neural networks of binary units. It is a generative graphical model that tries to learn the latent hidden stochastic units that could produce a target input. Learning exact maximum likelihood in such models, is intractable (because computation explodes exponentially with the number of hidden units $\boldsymbol{h}$), and approximations (using MCMC) can be time consuming. In this review, the focus is put on learning the parameters of general BMs. This paper also presents several ideas to improve learning, pre-training and testing for DBMs.

Approximate learning in Boltzmann Machines (BM's)

A Boltzmann Machine is a network with stochastic binary units divided into two sets : visible units $\boldsymbol{v} \in \{0,1\}^D$ and hidden units $\boldsymbol{h} \in \{0,1\}^P$. Such a network is usually defined by its energy E :

$$E(\boldsymbol{v},\boldsymbol{h};\theta) = -\frac{1}{2}\boldsymbol{v}^T\boldsymbol{L}\boldsymbol{v} - \frac{1}{2}\boldsymbol{h}^T\boldsymbol{J}\boldsymbol{h} - \boldsymbol{v}^T\boldsymbol{W}\boldsymbol{h}$$

Where the model parameter $\theta = \{\boldsymbol{W},\boldsymbol{L},\boldsymbol{J}\}$ represents interactions between the units. And the probability factorizing in this model is given by :

$$p(\boldsymbol{v};\theta) = \frac{1}{Z(\theta)}\sum_{\boldsymbol{h}}\exp(-E(\boldsymbol{v},\boldsymbol{h};\theta)), \quad \text{with } Z(\theta) = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}}\exp(-E(\boldsymbol{v},\boldsymbol{h};\theta))$$

The goal is to maximize the log-likelihood for a dataset $\{\boldsymbol{v}^{(1)},..,\boldsymbol{v}^{(N)}\}$ of N samples : $l(\theta) = \frac{1}{N}\sum_{n=1}^N \log p(\boldsymbol{v}_n,\theta)$. This leads to compute the derivatives of the log-likelihood with respect to each component of each parameter. For instance :

$$\begin{aligned}\frac{\partial l(\theta)}{\partial \boldsymbol{W}_{ij}} &= \frac{1}{N}\sum_{n=1}^N \frac{\partial}{\partial \boldsymbol{W}_{ij}}\log\left(\sum_{\boldsymbol{h}}\exp\left[\frac{1}{2}\boldsymbol{v}_n^T\boldsymbol{L}\boldsymbol{v}_n + \frac{1}{2}\boldsymbol{h}^T\boldsymbol{J}\boldsymbol{h} + \boldsymbol{v}_n^T\boldsymbol{W}\boldsymbol{h}\right]\right) - \frac{\partial \log Z(\theta)}{\partial \boldsymbol{W}_{ij}} \\ &= \mathbb{E}_{data}\left[v_{n,i}h_j\right] - \mathbb{E}_{model}\left[v_{n,i}h_j\right]\end{aligned}$$

with the complete data distribution $p_{data}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) = p(\boldsymbol{h}|\boldsymbol{v};\theta)p_{data}(\boldsymbol{v})$ ($p_{data}(\boldsymbol{v})$ being the empirical distribution) and $p_{model}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) = p(\boldsymbol{h}|\boldsymbol{v};\theta)p(\boldsymbol{v};\theta)$ (the latter distribution coming from the model). Such derivatives

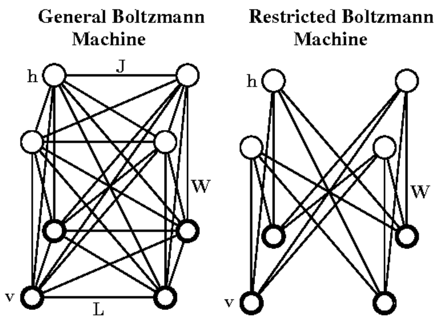

Figure 1: General (left) and Restricted (right) Boltzmann Machines (from [1]).

are computed for each parameter and allow to build a learning method based on gradient ascent. This leads to three similar equations (α being the learning rate) :

$$\Delta \boldsymbol{W} = \alpha \left(\mathbb{E}_{data}\left[\boldsymbol{v}\boldsymbol{h}^T\right] - \mathbb{E}_{model}\left[\boldsymbol{v}\boldsymbol{h}^T\right]\right) \tag{0.1}$$

$$\Delta \boldsymbol{L} = \alpha \left(\mathbb{E}_{data}\left[\boldsymbol{v}\boldsymbol{v}^T\right] - \mathbb{E}_{model}\left[\boldsymbol{v}\boldsymbol{v}^T\right]\right) \tag{0.2}$$

$$\Delta \boldsymbol{J} = \alpha \left(\mathbb{E}_{data}\left[\boldsymbol{h}\boldsymbol{h}^T\right] - \mathbb{E}_{model}\left[\boldsymbol{h}\boldsymbol{h}^T\right]\right) \tag{0.3}$$

Computing exactly those expectations is intractable because their complexity explodes with the number of hidden units. This is why the paper propose to approximate the *data-dependent expectations* $\mathbb{E}_{data}[.]$ with Persistent Markov Chains and the *model's expectations* (or *data-independent expectations*) $\mathbb{E}_{model}[.]$ with a variational approximation.

Approximate expectations using Gibbs Sampling

In order to compute the required expectations, traditionally two independent Gibbs sampling are used. One can easily apply cyclic scan Gibbs sampling or random scan Gibbs sampling in order to compute the needed expectations in the traditional Boltzmann Machine scheme.
One can compute the so-called *model's expectation* using the following conditional probabilities formulas :

$$p(h_j = 1|\boldsymbol{v}, \boldsymbol{h}_{-j}) = \sigma(\sum_{i=1}^{D} W_{ij}v_i + \sum_{\substack{m=0\\m\neq j}}^{P} J_{jm}h_j)$$

$$p(v_i = 1|\boldsymbol{h}, \boldsymbol{v}_{-i}) = \sigma(\sum_{j=1}^{P} W_{ij}h_j + \sum_{\substack{k=0\\k\neq i}}^{D} L_{ik}v_j)$$

Using those identities we can directly apply one of the Gibbs sampling scheme to sample the visible and the hidden units from those distributions.
As detailed above, data dependent distribution is given by $p_{data}(\boldsymbol{h}, \boldsymbol{v}; \theta) = p(\boldsymbol{h}|\boldsymbol{v}; \theta)p_{data}(\boldsymbol{v})$. In this setup, the visible units are independent and can be sampled easily. The hidden states conditional probability is still given by the formula above and can be computed in the same way. Theoretically, the *data-dependent expectations* could be sampled using the MCMC scheme.
Hence, it's feasible to compute both *model's expectation* and *data-dependent expectations* using multiple Gibbs sampling. However, computing that kind of MCMC based approach is not efficient enough : too much time would be required to approximate the stationary distribution. The author proposes an alternative procedure to improve the approximation of the expectations.

Persistent Markov chains to estimate the model's expectations

Given the parameters θ_t and the states X_t, Persistent Markov chains is a method that allows to speed up the traditional MCMC algorithm (see Algo 1).

```
Input : Random parameter $\theta_0$, random states $X_0$, set of decreasing learning rates $\{\alpha_t\}$
for $t \in \{1, ..., T\}$ do
    Sample $X_{t+1}$ from $X_t$ using the parameter $\theta_t$ (using Gibbs sampling for instance)
    Using $\alpha_t$, update $\theta_t$ using $X_{t+1}$ (that allowed to compute the expectations in 0.1)
end
```

Algorithm 1: Persistent Markov chains

This quite intuitive procedure works fairly well in practice. It has proof of convergence if the learning rate decreases with time i.e $\sum \alpha_t = \infty$ and $\sum \alpha_t^2 < \infty$. This is insured by setting $\alpha_t = \frac{1}{t}$. As one can see in the section about the general algorithm this scheme suits well our problem.

Data-dependant expectation using Variational Inference

One wants an estimation of $\mathbb{E}_{\text{data}}[.]$, where $p_{\text{data}}(\boldsymbol{h}, \boldsymbol{v}; \theta) = p(\boldsymbol{h}|\boldsymbol{v}; \theta) p_{\text{data}}(\boldsymbol{v})$. As an alternative to Gibbs sampling, the authors propose a method based on a variational learning to estimate the *data-dependant expectation.*

Since it is difficult to compute the likelihood, variational learning aims to maximize a lower bound of the log-likelihood, by replacing the hidden variables' true distribution by an approximate posterior $q(h|v;\theta)$. Thus, the log-likelihood has a lower bound as following :

$$\log(p_\theta(v) \geq \sum_h q_\theta(h|v) \log(p_\theta(h,v)) - \sum_h q_\theta(h|v) \log(q_\theta(h|v)) = \mathcal{L}(\theta, q)$$

Using the mean-field technique, which consists in choosing q in a set that makes all variables independent, we define $q_\theta(h) = \prod_i q_\theta(h_i) = \prod_i (\mu_i)^{h_i}(1-\mu_i)^{1-h_i}$. Hence, the lower bound to maximize is as follows :

$$\boxed{\begin{aligned}\mathcal{L}(\theta, \mu) = \frac{1}{2}\sum_{i,k} L_{i,k} v_i v_k + \frac{1}{2}\sum_{i,l} J_{j,l}\mu_j\mu_l + \sum_{i,j} W_{i,j} v_i \mu_j - \log(Z(\theta)) \\ + \sum_j [\mu_j \log(\mu_j) + (1-\mu_j)\log(1-\mu_j)]\end{aligned}}$$

One first maximizes it, with θ fixed, with respect to μ : $\mu_j^{t+1} \in \text{argmax}_\mu \mathcal{L}(\theta_t, \mu)$. For $j \in \{1, .., H\}$, we derive with respect to θ_j, and hence the optimal μ_j is given by :

$$\mu_j^{t+1} = \sigma(\sum_i W_{i,j} v_i + \sum_{m \backslash j} J_{m,j}\mu_m)$$

Thus, for each training example $\boldsymbol{v}_n$ in the training set $(\boldsymbol{v}_n)_{n \in \mathbb{N}}$ of visible variables, we initialize randomly μ_0, then we run the mean-field method until convergence to a certain μ_n. The *data-dependant expectation* is then approximated by : $\frac{1}{N}\sum_{n=1}^N v_n \mu_n^t$. Notice that variational approach for the *model's expectation* was not possible because of the minus sign in the learning equations. One could still use variational learning and then Gibbs sampling to estimate respectively the *data-dependent* and the *model's expectations* (see Algo 2).

Learning in Deep Boltzmann Machines (DBM's)

Then, the focus is on learning Deep Boltzmann Machines (DBM's), in which there are a stack of several hidden layers with no within-layer connections as shown in Figure 2 left handside. DBM's were introduced because they can learn complex representations in an unsupervised framework (if the amount of data is large

```
**Input :** Training set of size N : $(\boldsymbol{v}_n)_{n\in\{1,..,N\}}$
**Initialize :** Random initialization of the model's parameters $\theta_0$, M fantasy particles
  $((\tilde{v}_0^1,\tilde{h}_0^1),...,(\tilde{v}_0^M,\tilde{h}_0^M))$
**for** $t \in \{1,...,T\}$ **do**
    - Apply mean-field until convergence for each training example $v_n \quad \rightarrow \mu_n^t$
    - Apply Gibbs Sampling for each fantasy particle 'm', to obtain $(\tilde{v}_{t+1}^m,\tilde{h}_{t+1}^m)$, by initializing it to
      the previous sample $(\tilde{v}_t^m,\tilde{h}_t^m)$
    - Update :
    $\boldsymbol{W}^{t+1} = \boldsymbol{W}^t + \alpha_t(\frac{1}{N}\sum_{n=1}^N v_n\mu_n^t - \frac{1}{M}\sum_{m=1}^M \tilde{v}_{t+1}^m(\tilde{h}_{t+1}^m)^T)$
        Apply same method on $\boldsymbol{L}$ and $\boldsymbol{J}$.
    - Decrease the learning rate $\alpha_t$
**end**
```

Algorithm 2: BM Learning procedure

enough) and, because of their undirected structure, DBM's allow a better propagation during approximate inference (thanks to bottom-up and top-down feedbacks). Since DMBs are considered as special cases for Boltzmann machines, we can apply the same learning procedure, which is described in the previous part.

Greedy Layerwise Pretraining

For a DBM, the learning method presented before can be very slow because of the depth of the network. To overcome this obstacle, the authors introduce a pretraining method : training is done layer-by-layer by decomposing blocks of three layers (see Figure 2 right-hand side) of the DBM into two stacks of Restricted Boltzmann Machines (RMB's). For learning RBM's, the authors use the algorithm introduced by Hinton et al. (2006), that we do not detail here. The initialization of a DBM in such a way shows better estimation of the parameter θ than random initialization and allows very fast (because of the single bottom-up pass through the layers) approximate inference.

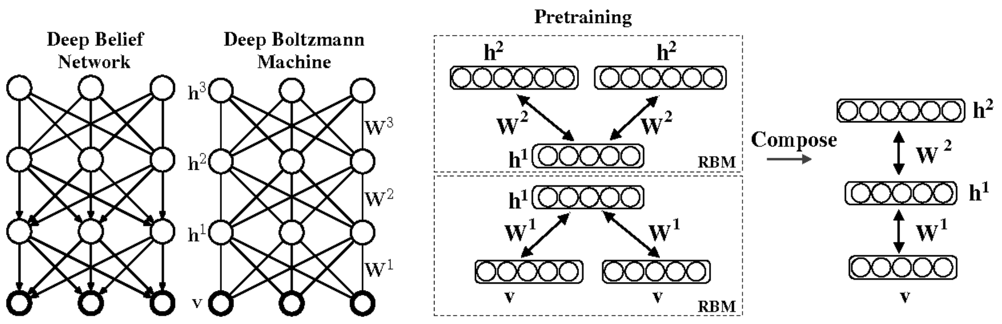

Figure 2: Left : 3-layer DBM (left) and 3-layer DBN (right). **Right** : Pretraining scheme (from [1]).

Annealed Importance Sampling (AIS) for DBM's evaluation

In order to have a better estimation of the lower bound of the likelihood on test data $\boldsymbol{v}^*$, one would like an estimate of the partition function $\hat{Z}$. A first way to proceed would be to use Simple Importance Sampling (SIS) that consists in computing : $\frac{Z_B}{Z_A} = \int p_B^*(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \int \frac{p_B^*(\boldsymbol{x})}{p_A^*(\boldsymbol{x})}p_A(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$, with known $p_A(\boldsymbol{x}) = p_A^*(\boldsymbol{x})/Z_A$ and unknown $p_B(\boldsymbol{x}) = p_B^*(\boldsymbol{x})/Z_B$. If one can sample from p_A, closed to p_B, this expectation can be estimated by Monte-Carlo (MC) . But in a DBM, if one takes p_A as the uniform distribution, SIS will not lead to a

good estimator of Z_B because p_B is more complex than a uniform distribution. Another problem might the high variance of MC estimators (variance reduction methods might help).
The authors introduce a better approximation method : Annealed Importance Sampling (AIS). It consists in sequentially applying importance sampling with intermediate distributions. That allows to go from an initial distribution (uniform model) to the targeted one (distribution of a layer $\boldsymbol{h}^i$ given its neighbours $\boldsymbol{h}^{-i}$). AIS combined with variational inference lead to an estimate of the lower bound of the log-partition function :

$$\ln p(\boldsymbol{v}^*; \theta) \geq -\sum_{\boldsymbol{h}} q(\boldsymbol{h}; \mu) E(\boldsymbol{v}, \boldsymbol{h}; \theta) + \mathscr{H}(q) - \ln Z(\theta) \approx -\sum_{\boldsymbol{h}} q(\boldsymbol{h}; \mu) E(\boldsymbol{v}, \boldsymbol{h}; \theta) + \mathscr{H}(q) - \ln \hat{Z}$$

The key point in this method leads in the fact that, thanks to the lack of intra-layer connections, one can sum out odds and layers to get a very precise estimation of the partition function.

Conclusion

Throughout the article, the authors propose a new scheme to efficiently train DBMs' parameters. They use a combination of Gibbs sampling and Variational Inference to provide an approximation of the expectations needed in the optimization update of the parameters. They also propose AIS to tighten the log-probability lower bound for test data. The authors use their learning procedure to compute both a generative and a discriminative models that uses hidden units as additional features. Pretty good outcomes as been observed.