CS7140: Advanced Machine Learning **Spring 2020**

Lecture 11: Boltzmann Machines

Lecturer: Rose Yu *Scribes: Armand Comas*

11.1 Boltzmann Machines

11.1.1 Background: Hopfield Network

From the previous lecture we know that the Hopfield Network Minimizes the following Energy function:

$$E(x; W) = -\frac{1}{2}\sum_{m,n} W_{mn} x_m x_n \tag{11.1}$$

We also know that when a continuous Hopfield network with activation function $x_n = \tanh(a_n)$ can be seen as approximating the probability distribution associated with the previous energy function:

$$P(x|\beta, W) = \frac{1}{Z}\exp\left[-\beta E(x; J)\right] \tag{11.2}$$

A Boltzmann machine is, in short, a stochastic version of a Hopfield network. Therefore, it is an Energy-Based Model where $P(x) \propto \exp\left[-\beta E(x)\right]$ can be described as the *Temperature* of such model.

11.1.2 Stochastic Hopfield Network - Boltzmann Machine principles

The Boltzmann Machine, or as described before, a Stochastic Hopfield Network, has the following activity rule:

- After computing the activation as: $a_i(t) = \sum_j w_{ij} x_j(t)$,
- Set $x_i = +1$ with probability $\frac{1}{1+e^{-2a_i}}$
- Set $x_i = -1$ with probability $1 - \frac{1}{1+e^{-2a_i}}$
- Sample from the Boltzmann Machine with an MCMC method from the probability distribution defined in Equation 11.2.

Gibbs sampling is therefore proposed for estimating the state-dependent probability distribution $P(x|\beta, W)$. Details on Gibbs sampling method have been previously explained in class. However, it is important to highlight that MCMC methods in general have the tendency to get stuck in low-energy regions, i.e. local minima.

11.1.3 Boltzmann Machine Learning

Our interest is clear. We want to adjust the weights $\mathbf{W}$ such that our generative model is well fitted to certain real world distributions. The generative model is given by:

$$P(x|\mathbf{W}) = \frac{1}{Z(\mathbf{W})}\exp\left[-\frac{1}{2}\mathbf{x^T M x}\right], \tag{11.3}$$

which is a case of equation 11.2. Here, the input data is given as $\left\{\mathbf{x}^{(n)}\right\}_{n=1}^{N}$. Through Bayes' theorem we obtain:

$$P(\mathbf{W}|\left\{\mathbf{x}^{(n)}\right\}_{n=1}^{N}) = \frac{\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}|\mathbf{W})\right] P(\mathbf{W})}{P(\left\{\mathbf{x}^{(n)}\right\}_{1}^{N})}. \tag{11.4}$$

We focus on the likelihood and derive the maximum likelihood estimation algorithm. The log-likelihood will therefore be given by:

$$\left[\prod_{n} P(\mathbf{x}^{(n)}|\mathbf{W})\right] = \sum_{n}\left[\frac{1}{2}\mathbf{x}^{(n)T}\mathbf{W}\mathbf{x}^{(n)} - \log Z(\mathbf{W})\right] \tag{11.5}$$

, to maximize it, as usual, we make use of its logarithm to obtain the direction of the optimization. The gradient of the log-likelihood is:

$$\frac{\delta}{\delta w_{ij}} \log P(\left\{\mathbf{x}^{(n)}\right\}|\mathbf{W}) = \sum_{n}^{N}\left[\left\langle x_i^{(n)} x_j^{(n)}\right\rangle - \left\langle x_i x_j\right\rangle_{P(\mathbf{x}^{(n)})}\right]. \tag{11.6}$$

This equation can be divided in two main terms, which define very well the objective function:

- Data correlation: $\langle x_i x_j\rangle_{\text{Data}} = \sum_n x_i^{(n)} x_j^{(n)}$, which is often interpreted as the *"Wake"* term: increases the weights in the *"real world"*.
- Model Correlation $\langle x_i x_j\rangle_{P(\mathbf{x}^{(n)})} = \sum_{\mathbf{x}} x_i x_j P(\mathbf{x}|\mathbf{W})$, which can be interpreted as the *"Sleep"* term. It decreases the weights in the *"dream"*. This term will be estimated by Monte Carlo Methods. Those terms will have to balance each other for the expression to converge.

11.1.3.1 Estimation with MCMC

Generally, taking the derivative with respect to the log-likelihood is difficult, as the normalization term itself depends on the weights $\mathbf{W}$ and it's intractable. The gradient of $Z(\mathbf{W})$ will be:

$$\Delta_{\mathbf{W}} \log Z = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \Delta_{\mathbf{W}} \log P^*(\mathbf{x}) \tag{11.7}$$

and MCMC methods will be used to approximate the maximum likelihood problem under this apparently intractable partition function.

11.1.3.2 Naive MCMC

A first approach for maximum Likelihood estimation is to take a Naive MCMC method for the energy based model. This is done as follows:

1. We sample a batch of data $\left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(B)}\right\}$.
2. Compute gradient (the expected value) $\mathbf{g} = \frac{1}{B}\sum_b \Delta_{\mathbf{W}} \log P^*(\mathbf{x}^{(b)}; \mathbf{W}$. Namely, we compute the average of the gradients of the unnormalized distribution for each data point, and normalize it by the batch size B. This is an estimation of the gradient of $\log Z(\mathbf{W})$.
3. We randomly initialize the states $\left\{\widetilde{\mathbf{x}}^{(1)}, \ldots, \widetilde{\mathbf{x}}^{(B)}\right\}$ using MCMC.

4. We update those states with Gibbs sampling.
5. Finally we update $\mathbf{g} = \mathbf{g} - \frac{1}{B}\sum_b \Delta_{\mathbf{W}} \log P^*(\widetilde{\mathbf{x}}^{(b)}; \mathbf{W})$.
6. Iterate until convergence

This is a naive way of performing gradient descent. What makes it problematic, is that the second, third and fourth bullet points take a great amount of computation, as it is known to be the case when we approximate complex probability functions by means of MCMC methods. Figure 11.1 shows how the optimization algorithm has two differenciated phases during learning which need to be balanced. The positive phase and the negative phase.

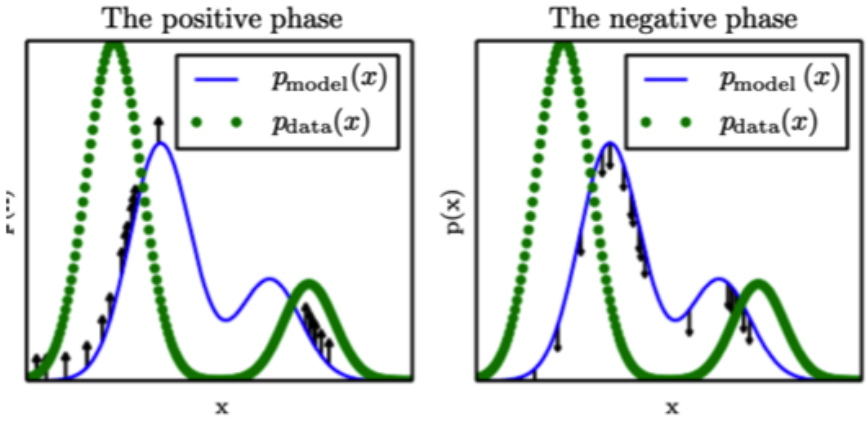

Figure 11.1: Positive and negative phases of Naive MCMC method for maximum likelihood estimation. The data distribution and the model distribution need to be balanced for the algorithm to converge.

We identify the burn-in operation that occurs for every mini-batch as the source of the computation charge. This is, in order to sample from the model we need to randomly initialize the states for every minibatch.

To address this issue, we will make use of the Contrastive Divergence technique.

11.1.3.3 Contrastive Divergence

Random initialization of the states is very inefficient. Instead, contrastive divergence changes a single step of the Naive MCMC algorithm. The state initialization of step 3., is done by actual data as follows:

- Initiate states $\left\{\widetilde{\mathbf{x}}^{(1)}, \ldots, \widetilde{\mathbf{x}}^{(B)}\right\}$ with data $\widetilde{\mathbf{x}}^b \leftarrow \mathbf{x}^b$.

11.1.4 Criticism

The main problem with the previously described algorithms is that they only capture second-order statistics (i.e. pair-wise relationships). In many cases those statistics are not enough to capture realistic relationships between random variables. For this reason we need higher-order concepts. For example, we couldn't learn the distribution of complex objects in real-world, which highly rely on complex contours and geometry, with second-order-only statistics.

A very intuitive example for this is the shift ensemble (Figure 11.2). We can see in the figure that using a Boltzman Machine will fail because if we compute the pair-wise correlation between pixels we will find that they are independent (correlation will be 0), namely, no meaningful information will be captured. However, the correlation is very clear when we look at it, and could be captured with higher-order statistics.

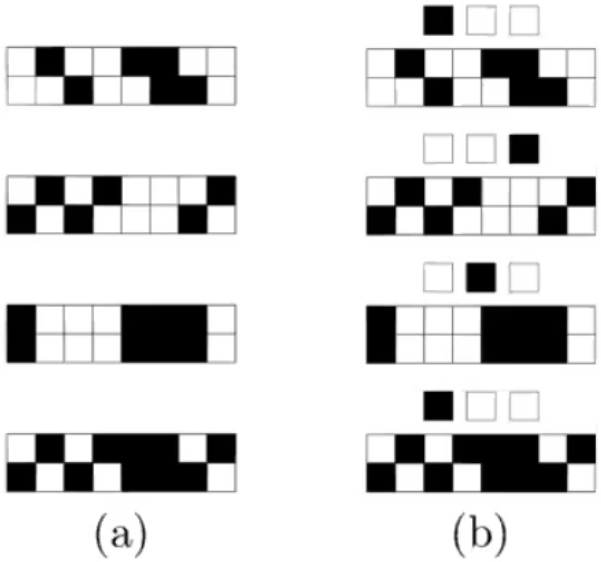

Figure 11.2: Column (a) shows 2-rowed images, in which one row is the shifted version (by 1 pixel) of the other row. Column (b) shows the labeled shift ensemble. It is the same as (a) with the difference that each image will have a label above showing whether the upper row is shifted left, right or not shifted with respect to the lower row.

11.1.4.1 Higher-Order Correlations

We can add terms to the expression we know for the Boltzmann Machine probability function, in a way that we also take into account sample triplets, quadruplets, etc.:

$$P(\mathbf{x}|\mathbf{W},\mathbf{V},\dots) = \frac{1}{Z}\exp\left(\frac{1}{2}\sum_{ij} w_{ij}x_i x_j + \frac{1}{6}\sum_{ij} v_{ijk}x_i x_j x_k + \dots\right) \tag{11.8}$$

The learning algorithm is the same, but in this case we need more parameters to learn the new higher-order dependencies. This at its time requires a higher amount of data.

To avoid those negative implications to grow into an intractable situation, we can include hidden variable, which will describe those high-order correlations.

In this case, the expression of the activity rule and probability function associated to the Boltzmann machine will vary slighly as:

$$\text{activity rule: } \mathbf{y}^{(n)} \equiv \left(\mathbf{x}^{(n)}, \mathbf{h}\right) \tag{11.9}$$

$$P(\mathbf{x}^{(n)}|\mathbf{W}) = \sum_h P(\mathbf{x}^{(n)}, \mathbf{h}|\mathbf{W}\frac{1}{Z(\mathbf{W})}\exp\left[-\frac{1}{2}[\mathbf{y^{(n)}}]^{\mathbf{T}}\mathbf{M}\mathbf{y}^{(n)}\right] \tag{11.10}$$

Therefore, when we look at the gradient with respect to the weights, the expression also looks very similar to Equation 11.6:

$$\frac{\delta}{\delta w_{ij}} \log P(\left\{\mathbf{x}^{(n)}\right\} | \mathbf{W}) = \sum_{n}^{N} \left[\langle y_i y_j \rangle_{P(\mathbf{h}|\mathbf{x}^{(n)}, \mathbf{W})} - \langle y_i y_j \rangle_{P(\mathbf{x}, \mathbf{h}|\mathbf{W})} \right], \tag{11.11}$$

with the data correlation term and the model correlation term clearly differentiated. The gradients, similarly to before, are still difficult to compute.

References

[1] Yang Yuhong, "Information Theory, Inference, and Learning Algorithms." (2005). Chapter 43

[2] Ian G Goodfellow and Yoshua Bengio and Aaron C. Courville "Deep Learning." (Nature, 2005). Chapter 17