Restricted Boltzmann Machines

Summer Research Internship

Advisor: Prof Bruno Ribeiro

During the summer months of 2017, I had the privilege of working with Prof. Bruno Ribeiro from the Computer Science department of Purdue University. I was working with him to improve the training method for RBMs. He proposed a Las Vegas transformation of the MCMC estimators of RBMs to obtain unbiased gradient estimates and hence better parameter estimates for the model.

The first one-two weeks of the internship were packed with learning the theory of Energy-based models, Markov Random Field, Markov Chain Monte Carlo techniques and Gibbs Sampling. A summary can be found here.

Log-Likelihood Gradient of MRFs with Latent Variables

A typical problem in training RBMs is the intractability of its log-likelihood function:

$$-\frac{\partial \log p(x^{(t)})}{\partial \theta} = E_h\left[\frac{\partial E(x^{(t)}, h)}{\partial \theta} \mid x^{(t)}\right] - E_{x,h}\left[\frac{\partial E(x, h)}{\partial \theta}\right]$$

It consists of two terms: positive and negative phase. The negative phase is the expected value of the energy function under the model distribution. Directly calculating this sum, which runs over all the values of the respective variables, leads to a computational complexity which is in general exponential in the number of variables of the MRF.
To avoid this computational burden, the expectations can be approximated by samples drawn from the corresponding distributions based on MCMC techniques.

Markov Chain Monte Carlo and Gibbs Sampling:

Markov chains play an important role in RBM training because they provide a method to draw samples from complex probability distributions like the Gibbs distribution of an MRF. From the theory of Markov Chains, we know that an irreducible and aperiodic Markov chain on a finite state space is guaranteed to converge to its stationary distribution. That is, for an arbitrary starting distribution μ it holds that,

$$\lim_{k \to \infty} d_V(\mu^T P^k, \pi^T) = 0$$

where P is the transition matrix, π is the stationary distribution and d_V is the distance of variation. Markov chain Monte Carlo methods make use of this convergence theorem for producing samples from certain probability distribution by setting up a Markov chain that converges to the desired distributions. Suppose you want to sample from a distribution q with a finite state space. Then you construct an irreducible and aperiodic Markov chain with stationary distribution $\pi = q$. If t is large enough, a sample $X^{(t)}$ from the constructed chain is then approximately a sample from π and therefore from q.
Gibbs Sampling is a simple MCMC algorithm for producing samples from the joint probability distribution of multiple random variables. The basic idea is to update each variable subsequently based on its conditional distribution given the state of the others.

Restricted BoltzMann Machines

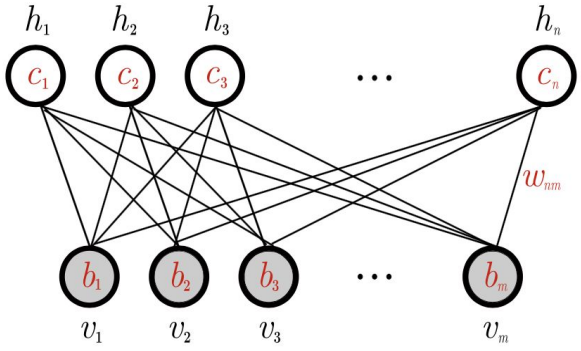
A RBM is an MRF associated with a bipartite undirected graph. It consists of m visible units $V = (V_1, \ldots, V_m)$ to represent observable data and n hidden units $H = (H_1, \ldots, H_n)$ to capture dependencies between observed variables. In binary RBMs the random variables (V, H) take values $(v, h) \in \{0, 1\}^{m+n}$ and the joint probability distribution under the model is given by the Gibbs distribution,

$$p(v, h) = \frac{1}{Z} e^{-E(v, h)}$$

with the energy function

$$E(v, h) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$

for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, w_{ij} is a real valued weight associated with the edge between units V_j and H_i and b_j and c_i are real valued bias terms associated with the j^{th} visible and the i^{th} hidden variable, respectively.



The graph of an RBM has only connections between the layer of hidden and visible variables but not between two variables of the same layer. In terms of probability this means that the hidden variables are independent given the state of the visible variables and vice versa:

$$p(h \mid v) = \prod_{i=1}^{n} p(h_i \mid v) \text{ and } p(v \mid h) = \prod_{j=1}^{m} p(v_j \mid h)$$

We can also show that,

$$p(H_i = 1 \mid v) = \sigma\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right)$$

$$p(V_j = 1 \mid h) = \sigma\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right)$$

The independence between the variables in one layer makes Gibbs sampling especially easy: Instead of sampling new values for all variables subsequently, the states of all variables in one layer can be sampled jointly. Thus, Gibbs sampling can be performed in just two substeps: sampling a new state h for the hidden neurons based on $p(h \mid v)$ and sampling a state v for the visible layer based on $p(v \mid h)$.

We have one problem though. In the real world, one is expected to run the Markov Chain for only K steps, hence returning a state $x \sim \widehat{\pi}(K)$, **“approximately sampled”** from the Markov chain’s true steady state distribution. Starting from a random state, K needs to be quite large for this method to work.

Contrastive Divergence

Recently it was shown that estimates obtained after running the chain for just a few steps can be sufficient for model training. This leads to contrastive divergence (CD) learning which has become a standard way to train RBMs. The idea of k-step contrastive divergence learning (CD-k) is quite simple: Instead of approximating the negative phase in the log-likelihood gradient by a sample from the RBM-distribution (which would require to run a Markov chain until the stationary distribution is reached), a Gibbs chain is run for only k steps (and usually $k = 1$). The Gibbs chain is initialized with a training example $v^{(0)}$ of the training set and yields the sample $v^{(k)}$ after k steps. Each step t consists of sampling $h^{(t)}$ from $p(h \mid v^{(t)})$ and sampling $v^{(t+1)}$ from $p(v \mid h^{(t)})$ subsequently. The gradient w.r.t θ of the log-likelihood for one training pattern $v^{(0)}$ is then approximated by:

$$CD_k(\theta,\ v^{(0)}) \ = \ - \ \sum_h p(h \mid v^{(0)}) \ \frac{\partial E(v^{(0)},\ h)}{\partial \theta} \ + \ \sum_h p(h \mid v^{(k)}) \ \frac{\partial E(v^{(k)},\ h)}{\partial \theta}$$

Empirically, CD-K works tremendously well to train RBMs with few hidden units (n_H small) even for K as low as $K \ = \ 1$. But for **high-dimensional RBMs, CD-K is less efficient** making it ineffective for modern applications.

Markov Chain Las Vegas (MCLV-K)

Prof Bruno Ribeiro proposes a Las Vegas transformation of Markov Chain Monte Carlo estimators of RBMs - Markov Chain Las Vegas (MCLV). MCLV gives statistical guarantees in exchange for random running times. MCLV uses a stopping set built from the training data and has maximum number of Markov chain steps K (referred as MCLV-K).
In standard RBM training using MCMC, the MC stops after a predefined number of K steps. In the MCLV approach, the MCMC can also stop if it reaches one of the states in the stopping set. Thus, MCMC running times are random (in average, shorter than K).
This approach is closer to a Las Vegas algorithm than a Monte Carlo algorithm: we aim to get perfect samples of a quantity with an algorithm that has random running times.
Moreover, it has been shown that by dynamically adapting K, MCLV-K can find unbiased estimates of the direction of the RBM gradient.

MCLV-K estimation with statistical guarantees:

If the Markov chain $\Phi\,(W)$ is not run until equilibrium, the gradient estimates have an unknown bias. We use Markov chain tours to take care of this bias.

Tours:
Define a tour to be a sequence of ξ steps of the Markov chain $(X(1),\ \ldots\ ,\ X(\xi))$ such that the state of the $(\xi \ + \ 1)$-st step is the same as the starting state, i.e. $X(1) \ = \ X(\xi \ + \ 1)$.
Let $\Xi^{(r)} \ = \ (X^r(1),\ \ldots\ ,\ X^r(\xi^{(r)}))$ denote the r-th tour, $r \geq 1$. The strong Markov property guarantees that if $s \ \neq \ r$ the sequences $\Xi^{(r)}$ are independent of $\Xi^{(s)}$. This independence guarantees that both $\Xi^{(r)}$ and $\Xi^{(s)}$ are sample paths obtained from the equilibrium distribution of the Markov chain.
However, as is, tours are not a practical concept for RBMs because in such a large state space Ω, the tour is unlikely to return to the same starting state. However, we can use a Markov chain property common to Metropolis-Hastings and Gibbs sampling Markov

chains to significantly increase the probability of return by collapsing a large number of states into a single state.

RBM stopping set:
The stopping set S uses sampled hidden states from the training data, $H_N^{(m)} = \{h_n^{(1)}, ..., h_n^{(m)} : h_n \sim p(h \mid v_n; W)\}$ where $\{v_n\}^N{}_{n=1}$ is the training data. The stopping set contains all hidden states in $H_N^{(m)}$ and all possible visible states

$$S_{HN}^{(m)} = \bigcup_{h \in H_N^{(m)}, v \in V} \{(v, h)\}$$

In practice, we do not store $S_{HN}^{(m)}$ in memory, rather we only keep $H_N^{(m)}$ in memory, as reaching a hidden state in $H_N^{(m)}$ is enough to guarantee we need to stop.

Simulating the Markov Chain:
For any MC resulting from standard Gibbs sampling MCMCs, we can simulate the transitions in and out of S by

1. $p_{\Phi'}(S, x)$, we first sample a state y with replacement from S with probability $e^{-E(y; W)} / Z_S(W)$ and then perform a transition $p_\Phi(y, x)$
2. $p_{\Phi'}(x, S)$ is also simulated by performing a transition $p_\Phi(x, y)$ and stopping the MC if $y \in S$

A tour starts by sampling the initial tour state x and stopping when the tour reaches the stopping set S.

MCLV-K gradient estimates

The following provides an estimate of the gradient of the negative log-likelihood of RBMs using MCLV-K. It has a scaling factor but the gradient direction is the same as the original gradient:

$$\nabla_W L_Z = \frac{Z(W)}{Z_S(W)} \left(\frac{1}{N} \sum_{n=1}^{N} v_n \, \mathrm{E}_W(h \mid v_n) - \mathrm{E}_W(v^T h) \right)$$

The scaling factor $Z(W) / Z_S(W)$ is constant given W and can be compensated in the learning rate.

Las Vegas Slope Estimator:

$$\widehat{\nabla_W L_{LVS}}(K,\ R) \ =\ \eta\Big(\frac{1}{N}\sum_{n=1}^{N}\frac{\partial E(x_n;\ W)}{\partial W} - \frac{1}{\sum_{k=1}^{K}|C_k|}\sum_{k=1}^{K}\ \sum_{(X(1),\ ...,\ X(k))\in C_k}\ \sum_{i=1}^{k}\frac{\partial E(X^{(r)}(k);\ W)}{\partial W}$$

k represents the states of the tour, $C_k \ =\ \{(x,\ X^{(i)}(2),\ \dots\ ,\ X^{(i)}(k))\}_i$ be the set of tours of length $k\ \leq K$, with x sampled from S.

Differences between LVS-K AND CD-K

1. LVS-K starts at a state x of $S_{HN}{}^{(m)}$ with probability proportional to $exp(-E(x;\ W))$, CD-K starts uniformly over the training examples. Thus, the negative phase of LVS-K tends to push the model away from unbalanced probabilities over the training examples.
2. At every Markov chain step, LVS-K stops early if it has reached a state in the stopping set , while CD-K will always perform all K steps.
3. The gradient estimates of LVS-K use only the completed tours, while CD-K uses all tours.
4. The gradient estimates of LVS-K use all states visited by the MC during a tour, while CD-K uses only the last visited state.