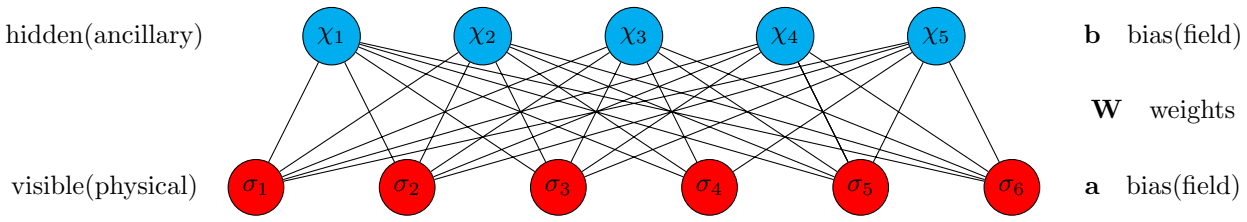
Restricted Boltzmann Machine

May 16, 2020

Contents



1 Introduction

The main task of Machine Learning(ML) is learning from data and making predictions. There are three main steps in ML:

1) Collecting the training data set, usually from experiments, our daily life, etc;
2) Finding a set of functions(also called model, dependent on a set of parameters) that is used to represent the law behind the data set;
3) Defining a loss function to measure the badness of your function, given certain parameter value, and find the best parameter value to minimize the loss function.

Considerations on each step can lead to methods with very different models and task-oriented algorithms.

2 Basic notation & Application[1, 2]

2.1 Basic Notion

Restricted Boltzmann Machine(RBM) is a natural model for studing statistical physics, which can be used to learn the probability distribution of a data set. For a particular configuration $(\vec{\sigma}, \vec{\chi}) \in \{0,1\}^{N+M}$, the RBM gives the joint distribution:

$$p(\vec{\sigma}, \vec{\chi}) = \frac{1}{Z} \exp\left(-E(\vec{\sigma}, \vec{\chi})\right), \tag{1}$$

where the corresponding energy is defined as:

$$E(\vec{\sigma}, \vec{\chi}) = -\sum_{i=1}^{M} b_i \chi_i - \sum_{j=1}^{N} a_j \sigma_j - \sum_{i,j} W_{ij} \chi_i \sigma_j, \tag{2}$$

and $Z = \sum_{\vec{\sigma},\vec{\chi}} \exp\left(-E(\vec{\sigma}, \vec{\chi})\right)$ is the so-called partition function which is essential to meet the normalization condition. Then we can get the marginal distribution for the physical configuration $\{\vec{\sigma}\}$:

$$p(\vec{\sigma}) = \sum_{\vec{\chi}} p(\vec{\sigma}, \vec{\chi}) = \frac{1}{Z} \exp\left(\sum_j a_j \sigma_j + \sum_i \ln\left(1 + \exp\left(b_i + \sum_j W_{ij}\sigma_j\right)\right)\right) = \frac{1}{Z} \exp\left(-\mathcal{E}(\vec{\sigma})\right), \tag{3}$$

and the red part is just the mean field term, while the cyan part, which is nonlinear, can capture the correlations. This makes RBM a powerful representation of a very complex probability distribution(indeed, any distribution with enough hidden units). Further more, we can calculate conditional distributions using Bayes's theorem as follows:

$$\begin{aligned} p(\vec{\sigma}|\vec{\chi}) &= \frac{\exp\left(-E(\vec{\sigma}, \vec{\chi})\right)/Z}{\sum_{\vec{\sigma}} \exp\left(-E(\vec{\sigma}, \vec{\chi})\right)/Z} \quad \text{insert the definition of energy } E(\vec{\sigma}, \vec{\chi}) \\ &= \frac{\prod_{j=1}^{N} \exp\left(a_j\sigma_j + \sum_i W_{ij}\chi_i\sigma_j\right)}{\prod_{j=1}^{N}(1 + \exp(a_j + \sum_i W_{ij}\chi_i))} \\ &= \prod_{j=1}^{N} \frac{\exp\left(a_j\sigma_j + \sum_i W_{ij}\chi_i\sigma_j\right)}{1 + \exp\left(a_j + \sum_i W_{ij}\chi_i\right)}. \end{aligned}$$

We can hence reformulate the conditional distributions in a 'decomposed' way(similar for $p(\vec{\chi}|\vec{\sigma})$)[1]:

$$p(\vec{\sigma}|\vec{\chi}) = \prod_{j=1}^{N} p(\sigma_j|\vec{\chi}) \quad \text{with} \quad p(\sigma_j = 1|\vec{\chi}) = \text{sigmoid}(a_j + \sum_i W_{ij}\chi_i) \tag{4}$$

$$p(\vec{\chi}|\vec{\sigma}) = \prod_{i=1}^{M} p(\chi_i|\vec{\sigma}) \quad \text{with} \quad p(\chi_i = 1|\vec{\sigma}) = \text{sigmoid}(b_i + \sum_j W_{ij}\sigma_j) \tag{5}$$

The sigmoid function is defined as: $\text{sigmoid}(x) = 1/(1 + \exp(-x))$[2]. An attractive thing about RBM is that: given any configuration $\vec{\sigma}$ as input, we can generate a data set(obey the distribution 3) from Markov Chain Monte Carlo(MCMC) using the update rules 4 and 5.

[1] Attention: the units in the same layer are independent variables.

[2] In practical programming, $\exp(-x)$ may cause overflow. There is another safe version for the sigmoid function as: $(1+\tanh(x/2))/2$, which is preferred.

2.2 Application

Our aim is learning the thermodynamics of a classical spin system, and all we need to know is the partition function(Ising model for example):

$$Z(\beta) = \sum_{\vec{\sigma}} \exp\left(-\beta E(\vec{\sigma})\right) \quad \text{with} \quad E(\vec{\sigma}) = -J \sum_{<i,j>} \sigma_i \sigma_j. \tag{6}$$

However, it is not practical to calculate it for a general model. One can alternatively sample a data set from the corresponding distribution $q(\vec{\sigma}) = \exp\left(-\beta E(\vec{\sigma})\right)/Z(\beta)$ and average the quantity interested over the data set. Here, we generate the data set(training set $\mathcal{D}$) via conventional Monte Carlo Simulation and train the RBM to learn the probability distribution $q(\vec{\sigma})$ hidden in the data set, finally we sample from the trained RBM and calculate the physical quantities. Thus, we need to find parameter value(namely $\mathbf{W}$, $\mathbf{b}$, $\mathbf{c}$) which makes the RBM represent the distribution $q(\vec{\sigma})$ best. Kullback Leibler(KL) divergence is a measure of the distance between two distributions:

$$\begin{aligned}
\mathbb{KL}(q||p) &= \sum_{\vec{\sigma} \text{ for all}} q(\vec{\sigma}) \ln \frac{q(\vec{\sigma})}{p(\vec{\sigma})} \quad \text{be zero iff } q(\vec{\sigma}) = p(\vec{\sigma}) \text{ for all } \vec{\sigma} \\
&= \sum_{\vec{\sigma} \text{ for all}} q(\vec{\sigma}) \ln q(\vec{\sigma}) - \sum_{\vec{\sigma} \text{ for all}} q(\vec{\sigma}) \ln p(\vec{\sigma}) \quad \text{the first term is fixed by the true distribution} \\
&\sim -\sum_{\vec{\sigma} \text{ for all}} \frac{1}{|\mathcal{D}|} \sum_{\sigma' \in \mathcal{D}} \delta_{\sigma,\sigma'} \ln p(\vec{\sigma}) \quad \text{note that } q(\vec{\sigma}) \text{ can be estimated from the data set } \mathcal{D} \\
&= -\frac{1}{|\mathcal{D}|} \sum_{\vec{\sigma} \in \mathcal{D}} \ln p(\vec{\sigma}) \quad \text{note that the term } \sum_{\vec{\sigma} \in \mathcal{D}} \ln p(\vec{\sigma}) = \ln \prod_{\vec{\sigma} \in \mathcal{D}} p(\vec{\sigma}) \text{ is also known as log-likelihood.}
\end{aligned}$$

The most common strategy to minimize the KL divergence(our loss function) is using Stochastic Gradient Descent(SGD). Thus we need to calculate the derivatives of KL divergence with respect to RBM parameters $\vec{\lambda}$(attention: we write $\{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ as $\vec{\lambda}$):

$$\nabla_{\vec{\lambda}} \mathbb{KL}(q||p) = -\frac{1}{|\mathcal{D}|} \sum_{\vec{\sigma} \in \mathcal{D}} \nabla_{\vec{\lambda}} \ln p_{\vec{\lambda}}(\vec{\sigma}). \tag{7}$$

Here we use $p_{\vec{\lambda}}(\vec{\sigma})$ to emphasize the parameter dependence of $p(\vec{\sigma})$. For each data $\vec{\sigma}^*$ in data set $\mathcal{D}$, we have:

$$\begin{aligned}
\nabla_{\vec{\lambda}} \ln p_{\vec{\lambda}}(\vec{\sigma}^*) &= \nabla_{\vec{\lambda}} \ln \frac{1}{Z_{\vec{\lambda}}} \sum_{\vec{\chi}} \exp\left(-E_{\vec{\lambda}}(\vec{\sigma}^*, \vec{\chi})\right) \\
&= \nabla_{\vec{\lambda}} \ln \sum_{\vec{\chi}} \exp\left(-E_{\vec{\lambda}}(\vec{\sigma}^*, \vec{\chi})\right) - \nabla_{\vec{\lambda}} \ln Z_{\vec{\lambda}} \\
&= -\frac{\sum_{\vec{\chi}} \exp\left(-E_{\vec{\lambda}}(\vec{\sigma}^*, \vec{\chi})\right) \nabla_{\vec{\lambda}} E_{\vec{\lambda}}(\vec{\sigma}^*, \vec{\chi})}{\sum_{\vec{\chi}} \exp\left(-E_{\vec{\lambda}}(\vec{\sigma}^*, \vec{\chi})\right)} + \frac{\sum_{\vec{\sigma},\vec{\chi}} \exp\left(-E_{\vec{\lambda}}(\vec{\sigma}, \vec{\chi})\right) \nabla_{\vec{\lambda}} E_{\vec{\lambda}}(\vec{\sigma}, \vec{\chi})}{\sum_{\vec{\sigma},\vec{\chi}} \exp\left(-E_{\vec{\lambda}}(\vec{\sigma}, \vec{\chi})\right)} \\
&= -\sum_{\vec{\chi}} p_{\vec{\lambda}}(\vec{\chi}|\vec{\sigma}^*) \nabla_{\vec{\lambda}} E_{\vec{\lambda}}(\vec{\sigma}^*, \vec{\chi}) + \sum_{\vec{\sigma},\vec{\chi}} p_{\vec{\lambda}}(\vec{\sigma}, \vec{\chi}) \nabla_{\vec{\lambda}} E_{\vec{\lambda}}(\vec{\sigma}, \vec{\chi}) \\
&= -\langle \nabla_{\vec{\lambda}} E_{\vec{\lambda}}(\vec{\sigma}^*, \vec{\chi}) \rangle_{p_{\vec{\lambda}}(\vec{\chi}|\vec{\sigma}^*)} + \langle \nabla_{\vec{\lambda}} E_{\vec{\lambda}}(\vec{\sigma}, \vec{\chi}) \rangle_{p_{\vec{\lambda}}(\vec{\sigma},\vec{\chi})},
\end{aligned}$$

the first term is averaged over the conditional distribution for hidden layer given a certain data which is easy to compute, while the second term is averaged over the distribution of the RBM which is difficult to calculate. Fortunately, the ML community have already proposed an efficient algorithm called contrastive divergence(CD) to estimate the second term. Here, we give more detail informations about the derivatives of KL divergence:

$$\nabla_{W_{ij}} \ln p_{\vec{\lambda}}(\vec{\sigma}^*) = \langle \chi_i \sigma_j^* \rangle_{p_{\vec{\lambda}}(\vec{\chi}|\vec{\sigma}^*)} - \langle \chi_i \sigma_j \rangle_{p_{\vec{\lambda}}(\vec{\chi},\vec{\sigma})} \tag{8}$$

$$\nabla_{a_j} \ln p_{\vec{\lambda}}(\vec{\sigma}^*) = \langle \sigma_j^* \rangle_{p_{\vec{\lambda}}(\vec{\chi}|\vec{\sigma}^*)} - \langle \sigma_j \rangle_{p_{\vec{\lambda}}(\vec{\chi},\vec{\sigma})} \tag{9}$$

$$\nabla_{b_i} \ln p_{\vec{\lambda}}(\vec{\sigma}^*) = \langle \chi_i \rangle_{p_{\vec{\lambda}}(\vec{\chi}|\vec{\sigma}^*)} - \langle \chi_i \rangle_{p_{\vec{\lambda}}(\vec{\chi},\vec{\sigma})}. \tag{10}$$

Once we have calculated the derivatives of KL divergence given a certain parameter value, we can update the parameters with following rule:

$$\vec{\lambda}^* = \vec{\lambda} - \eta \nabla_{\vec{\lambda}} \mathbb{KL}(\vec{\lambda}),$$

where the factor η is the learning rate. The SGD will find the best parameter value automatically. Once we have trained a RBM, we can use MCMC method to sample from RBM which have learned the distribution of training data, and compute the physical related quantities as follows(denote sampled data set as: $\mathcal{S}$):

$$\langle \mathcal{O} \rangle = \frac{1}{|\mathcal{S}|} \sum_{\vec{\sigma} \in \mathcal{S}} \mathcal{O}(\vec{\sigma}).$$

3 Training Algorithm

3.1 Gibbs Sampling and Contrastive Divergence

Now we have all the ingredients needed to train a RBM, and the main task is to calculate the values of 8, 9 and 10. Before we discuss the CD algorithm, it is essential to write these expressions more explicitly. For the first one:

$$\begin{aligned}
\nabla_{W_{ij}} \ln p_{\vec{\lambda}}(\vec{\sigma}^*) &= \sum_{\vec{\chi}} p_{\vec{\lambda}}(\vec{\chi}|\vec{\sigma}^*)\chi_i\sigma_j^* - \sum_{\vec{\sigma},\vec{\chi}} p_{\vec{\lambda}}(\vec{\sigma},\vec{\chi})\chi_i\sigma_j \quad \text{using the identity 5} \\
&= (\sum_{\chi_i} p_{\vec{\lambda}}(\chi_i|\vec{\sigma}^*)\chi_i\sigma_j^*)\prod_{k\neq i}(\sum_{\chi_k} p_{\vec{\lambda}}(\chi_k|\vec{\sigma}^*)) - \sum_{\vec{\sigma},\vec{\chi}} p_{\vec{\lambda}}(\vec{\sigma},\vec{\chi})\chi_i\sigma_j \\
&= \sum_{\chi_i} p_{\vec{\lambda}}(\chi_i|\vec{\sigma}^*)\chi_i\sigma_j^* - \sum_{\vec{\sigma}} p_{\vec{\lambda}}(\vec{\sigma})\sum_{\vec{\chi}} p_{\vec{\lambda}}(\vec{\chi}|\vec{\sigma})\chi_i\sigma_j \\
&= p_{\vec{\lambda}}(\chi_i=1|\vec{\sigma}^*)\sigma_j^* - \sum_{\vec{\sigma}} p_{\vec{\lambda}}(\vec{\sigma})p_{\vec{\lambda}}(\chi_i=1|\vec{\sigma})\sigma_j \\
&= p_{\vec{\lambda}}(\chi_i=1|\vec{\sigma}^*)\sigma_j^* - \langle p_{\vec{\lambda}}(\chi_i=1|\vec{\sigma})\sigma_j\rangle_{p_{\vec{\lambda}}(\vec{\sigma})} \qquad (11)
\end{aligned}$$

Similarly, we have:

$$\nabla_{a_j} \ln p_{\vec{\lambda}}(\vec{\sigma}^*) = \sigma_j^* - \langle\sigma_j\rangle_{p_{\vec{\lambda}}(\vec{\sigma})} \qquad (12)$$

$$\nabla_{b_i} \ln p_{\vec{\lambda}}(\vec{\sigma}^*) = p_{\vec{\lambda}}(\chi_i=1|\vec{\sigma}^*) - \langle p_{\vec{\lambda}}(\chi_i=1|\vec{\sigma})\rangle_{p_{\vec{\lambda}}(\vec{\sigma})} \qquad (13)$$

In order to get an average over distribution $p_{\vec{\lambda}}(\vec{\sigma})$, we can perform a Gibbs sampling[3]. However, this is very time-consuming, for you have to do many steps(may be $\sim 10^3$ or larger) of Gibbs sampling before it reach the equilibrium(convergence). CD algorithm uses two tricks to circumvent this difficulty[4]: 1. start the Gibbs sampling with a $\vec{\sigma}$ from the training data set, not a random one; 2. do sampling after only a few steps of Gibbs sampling(in many cases, a single step is good enough!). In the spirit of CD algorithm, we can thus replace the average over $p_{\vec{\lambda}}(\vec{\sigma})$ in 11, 12 and 13 with $p_{\vec{\lambda}}(\chi_i=1|\vec{\sigma}^k)\sigma_j^k$, σ_j^k and $p_{\vec{\lambda}}(\chi_i=1|\vec{\sigma}^k)$, where $\vec{\sigma}^k$ is sampled from k-step Gibbs sampling whose starting point is $\vec{\sigma}^*$[5].

3.2 Pseudocode For Training Algorithm

Here I just give a detailed description of the whole training algorithm, see Algorithm 1.

References

[1] Learning Thermodynamics with Boltzmann Machines, G. Torlai, R.G. Melko, arXiv:1606.02718v1

[2] An Introduction to Restricted Boltzmann Machines, A. Fischer, C. Igel, https://doi.org/10.1007/978-3-642-33275-3_2

[3] Start with a specific visible configuration $\vec{\sigma}^0$, we sample the hidden configuration $\vec{\chi}^0$ according to the probability 5 given $\vec{\sigma}^0$. Then we back to sample(reconstruct) the visible configuration $\vec{\sigma}^1$ according to the probability 4 given $\vec{\chi}^0$. If we do these sampling in order again and again, we will get a long sequence like(k-step Gibbs sampling): $\vec{\sigma}^0 \to \vec{\chi}^0 \to \vec{\sigma}^1 \to \vec{\chi}^1 \to \cdots \to \vec{\chi}^{k-1} \to \vec{\sigma}^k$. We refer to the ordered sampling procedure $\vec{\sigma} \to \vec{\chi} \to \vec{\sigma}$ as a single step of Gibbs sampling.

[4] The logic is that: our ultimate goal is getting a distribution $p_{\vec{\lambda}}(\vec{\sigma}) \sim q(\vec{\sigma})$. If we start Gibbs sampling with a $\vec{\sigma}$ sampled from $q(\vec{\sigma})$, it should already arrive at the equilibrium in some sense.

[5] You may be confused and ask the question that: how you can replace the average over a distribution $p_{\vec{\lambda}}(\vec{\sigma})$ just by one $\vec{\sigma}$ sampled from it? The answer lies in the fact that our derivative of KL divergence includes an average(or sum) over the whole data set or a batch in SGD(see expression 7). You may convince yourself that: this averaging process over training data set or batch indeed mimics the averaging process over the distribution $p_{\vec{\lambda}}(\vec{\sigma})$.

Algorithm 1 Training of RBM via SGD

Require: Data Set $\mathcal{D}$, RBM's Parameters $\{\mathbf{W}, \mathbf{a}, \mathbf{b}\}$, Learning Rate η, Batch Size bs, Epoch ep, CD Order k, Visible Unit Number: N and Hidden Unit Number: M.

Ensure: $\{\mathbf{W}^*, \mathbf{a}^*, \mathbf{b}^*\} = \arg\min_{\vec{\lambda}} \mathbb{KL}(q(\vec{\sigma})||p_{\vec{\lambda}}(\vec{\sigma}))$

Initialize $\{\mathbf{W}, \mathbf{a}, \mathbf{b}\}$. Each element in $\mathbf{W}$ is sampled from a gaussian distribution: $W_{ij} \sim \mathcal{N}(0,1)/\sqrt{(N+M)}$, all the elements in $\mathbf{a}, \mathbf{b}$ are set to zero: $b_i \leftarrow 0, a_j \leftarrow 0$.
Split $\mathcal{D}$ into batches $\mathcal{B}$ randomly, each is of size bs
for *epoch* in $range(1, ep)$ **do**
    for each batch B in $\mathcal{B}$ **do**
        $\Delta W_{ij}, \Delta a_j, \Delta b_i$ are set to 0, for all i, j
        for each data $\vec{\sigma}$ in batch B **do**
            sample $\vec{\sigma}^k$ via k-step Gibbs sampling with $\vec{\sigma}$ as the starting configuration
            calculate $p(\chi_i = 1|\vec{\sigma})$ and $p(\chi_i = 1|\vec{\sigma}^k)$ for all i via 4 and 5
            $\Delta W_{ij} \mathrel{+}= p(\chi_i = 1|\vec{\sigma}) * \sigma_j - p(\chi_i = 1|\vec{\sigma}^k) * \sigma_j^k$
            $\Delta a_j \mathrel{+}= \sigma_j - \sigma_j^k$
            $\Delta b_i \mathrel{+}= p(\chi_i = 1|\vec{\sigma}) - p(\chi_i = 1|\vec{\sigma}^k)$
        end for
        $W_{ij} \leftarrow W_{ij} + \eta * \Delta W_{ij}/bs$
        $a_j \leftarrow a_j + \eta * \Delta a_j/bs$
        $b_i \leftarrow b_i + \eta * \Delta b_i/bs$
    end for
end for