CSE6243: Advanced Machine Learning **Fall 2023**

Lecture 10: EBM (CD, Score Matching)

Lecturer: Bo Dai *Scribes:*

Note: *LaTeX template courtesy of UC Berkeley EECS Department.*

10.1 Recap

Recall the definition of *energy-based model*:

Definition 10.1 (Energy-based Model (EBM))

$$p_\theta(x) = \exp(f_\theta(x) - A(\theta)) \tag{10.1}$$

$$A(\theta) = \log \int \exp(f_\theta(x))dx \tag{10.2}$$

which satisfies $P_\theta(x) \geq 1$ and $\int P_\theta(x)dx = 1$.

We also introduced the *conditional EBM*:

Definition 10.2 (Conditional Energy-based Model)

$$p_\theta(y|x) = \exp(f_\theta(x, y) - A_x(\theta)) \tag{10.3}$$

$$A_x(\theta) = \log \int \exp(f_\theta(x, y))dy \tag{10.4}$$

which also satisfies $P_\theta(y|x) \geq 1$ and $\int P_\theta(y|x)dx = 1$.

From the perspective of statistical physics, we define the energy as $g_\theta(x) = -f_\theta(x)$. In this notation, a lower value of $g(\theta)$ corresponds to a higher value of $f_\theta(x)$. Consequently, a higher $f_\theta(x)$ results in a greater probability $p_\theta(x)$, indicating that x is more likely to reside in a low-energy region.

10.2 New Content

10.2.1 Maximum Entropy Model

The maximum-entropy model assumes that the network state probability distribution is given by an exponential function of the network energy such that entropy is maximized while satisfying any statistical constraints, i.e.,

$$\begin{aligned} \max_{p \in \Delta} \quad & H(p) = -\int p(x) \log p(x) dx \\ \text{s.t.} \quad & \mathbb{E}_p[f(x)] = \mu, \int p(x)dx = 1 \end{aligned} \tag{10.5}$$

Therefore, the Lagrangian formulation is

$$L(p) = H(p) - \eta(\mu - \mathbb{E}_p[f(x)]) - \lambda(1 - \int p(x)dx) \tag{10.6}$$

with the KKT conditions

$$\nabla_p L(p, \lambda) = -\log p - 1 + \eta f(x) + \lambda = 0 \tag{10.7}$$

$$\Rightarrow \quad \log p = 1 + \eta f(x) + \lambda \tag{10.8}$$

$$\int p(x)dx = 1 \tag{10.9}$$

$$\Rightarrow \quad \exp(\lambda - 1) = \frac{1}{\int \exp(\eta f(x))dx = 1} \tag{10.10}$$

Therefore, the optimizer is in the form of $\frac{\exp(\eta f(x))}{Z}$, with Z the partition function, i.e.,

$$\frac{\exp(\eta f(x))}{Z} = \arg\max H(p), \text{ s.t. } \mathbb{E}_p[f(x)] = \mu \tag{10.11}$$

10.2.2 Application of EBM

Unsupervised learning. In unsupervised learning, EBMs can be used to capture the structure and patterns inherent in the input data without any explicit labels. The model tries to learn an energy function where data points from the true data distribution are given lower energy values, while other points in the data space receive higher energy values.

Generative models. EBMs can generate new data samples by searching for configurations of the variables that minimize the energy function. An often-used method with EBMs is the Markov Chain Monte Carlo (MCMC) method to sample from the model's distribution.

Conditional models. In conditional modeling, the goal is to learn a mapping from an input space to an output space. For structured prediction tasks, the output space is often combinatorial, meaning that there are a large number of possible outputs. EBMs can be designed to take both an input (like an image) and a potential output (like a caption) and assign an energy value to the pair. During inference, the model can search for the output that minimizes the energy given the input.

10.2.3 Restricted Boltzmann Machine (RBM)

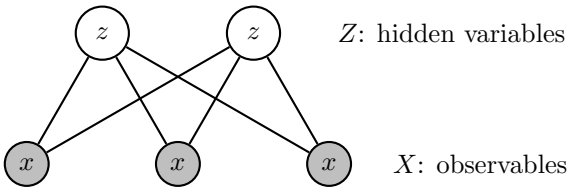

Figure 10.1: Structure of Restricted Boltzmann Machine

The RBM consists of two layers: a visible layer and a hidden layer. Each node in these layers is called a neuron or a unit. The "restricted" in RBM means that there are no intra-layer connections: neurons within

a layer do not connect with each other. They only connect with neurons in the other layer. Under these conditions, we can express

$$p(x, z) \propto \exp(x^\top W z + bX + Cz) \tag{10.12}$$

The RBM can be easily extended to its deep version – Deep Boltzmann Machine.

There are several pros & cons for the RBM:

Pros.

- RBM can clearly characterize the relations among variables and thus reveal dependence of x;
- It can represent the compositionality, i.e., the probabilistic logics. For example, let $f(x) = f_{\theta_1}(x) \bigwedge f_{\theta_2}(x)$, then $p(x) = p_{\theta_1}(x)p_{\theta_1}(x) \propto \exp(f_{\theta_1}(x) \bigwedge f_{\theta_2}(x))$.

Cons.

- RBM is hard to evaluate due to the intractability of partition function $Z(\theta) = \int \exp(f_\theta(x))dx$;
- It is also hard to sample, i.e., hard to set appropriate proposals in MCMC;
- It is different to learn.

We note the property of the RBM:

$$\frac{p_\theta(x)}{p_\theta(x')} = \exp(f_\theta(x) - f_\theta(x')) \tag{10.13}$$

which means it is relatively easy to calculate the ratio between the probabilities. We will leverage this point in the following MCMC design.

MCMC Deisgn. To sample $x \sim p(x) \propto \exp(f_\theta(x))$, we do

---

$x_0 \sim p_0(x)$
for $t = 1 \cdots T$ **do**
  $x = x_t$
  $y \sim q(\cdot|x)$
  $A(x, y) = \min(\frac{p(y)q(x|y)}{p(x)q(y|x)}, 1)$
  $u \sim U[0, 1]$
  if $u \leq A(x, y)$ **then**
    $x_{t+1} = y$
  else
    $x_{t+1} = x$
  end if
end for

As we mentioned, it is easy to get $\frac{p_\theta(y)}{p_\theta(x)}$. Now the question is *how to design* $q(\cdot|x)$*?*

There are several approaches:

- **Random walk:** $y = x_{t-1} + \Delta, \quad \text{with}\, \Delta \sim N(0, \sigma^2)$;
- **Langevin dynamics:** $y = x_{t-1} + \eta \nabla_x \log p_\theta(x_{t-1}) + \sqrt{\eta}\epsilon$.

Notably, although $\nabla_\theta \log p_\theta(x)$ is not tractable, $\nabla_x \log p_\theta(x)$ is.

10.2.4 Contrastive Divergence (CD)

Contrastive Divergence (CD) is an optimization algorithm primarily used for training Restricted Boltzmann Machines (RBMs) and other energy-based models. CD approximates the gradient of the log-likelihood and helps update the model's parameters to better fit the data. We may look back to the MLE.

Maximum Likelihood Estimation (MLE). Given the dataset $\mathcal{D} = x_i$, we do

$$\max_\theta \frac{1}{n}\sum_{i=1}^{n} \log P_\theta(x) = L(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_\theta(x_i) - A(f_\theta) \tag{10.14}$$

Taking the gradient of L yields

$$\nabla L(f) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_\theta(x_i) - \nabla_\theta A(f_\theta) \tag{10.15}$$

where

$$\nabla A(f_\theta) = \nabla \log \int \exp(f_\theta(x))dx \tag{10.16}$$

$$= \frac{\int \exp(f_\theta(x))\nabla f_\theta(x)dx}{\int \exp(f_\theta(x))dx} \tag{10.17}$$

$$= \mathbb{E}_{p_\theta}[\nabla_\theta f_\theta(x)] \tag{10.18}$$

Therefore,

$$\nabla \Delta L(f) = \underbrace{\hat{\mathbb{E}}_x[\nabla f_\theta(x)]}_{\text{easy to get}} - \underbrace{\mathbb{E}_{p_\theta}[\nabla_\theta f_\theta(x)]}_{\text{sample to estimate}} \tag{10.19}$$

10.2.5 Score Matching (SM)

Score Matching (SM) is an alternative method to train energy-based models, particularly in situations where the partition function (normalizing constant) is difficult or impossible to compute directly. The idea behind Score Matching is to adjust the parameters of the model so that the gradient (or "score") of the log-density of the model matches the score of the data distribution.

We start by introducing the *Fisher divergence*.

Definition 10.3 (Fisher Divergence)

$$D_{Fisher}(p, q) = \frac{1}{2}\mathbb{E}_p[\|\nabla_x \log p(x) - \nabla_x \log q(x)\|^2] \tag{10.20}$$

Given $p_\theta(x) = \frac{\exp(f_\theta(x))}{Z(\theta)}$, we know that

$$\nabla_x \log p_f(x) = \nabla f_\theta(x) \tag{10.21}$$

Therefore,

$$\mathbb{E}_{x\sim\hat{p}}[(\nabla_x \log \hat{p}(x) - \nabla_x \log p_f(x))^2] \tag{10.22}$$

$$= \int \hat{p}(x)(\nabla_x \log \hat{p}(x) - \nabla_x \log p_f(x))^2 dx \tag{10.23}$$

$$= \int p(x)(\nabla_x \log p_f(x))^2 dx - 2\int p(x)\nabla \log p(x)\nabla_x \log p_f(x)dx \tag{10.24}$$

where the second term

$$\int p(x)\nabla \log p(x)\nabla_x \log p_f(x)dx \tag{10.25}$$

$$= \int \nabla p(x)\nabla_x \log p_f(x)dx \tag{10.26}$$

$$= p(x)\nabla_x \log p_f(x)|_{-\infty}^{+\infty} - \int \nabla_x^2 \log p_f(x)p(x)dx \tag{10.27}$$

Here we just leveraged the *integral by parts*, i.e., $\int_a^b u(x)v'(x)dx = [u(x)v(x)]_a^b - \int u'(x)v(x)dx$. In (10.27), the term $p(x)\nabla_x \log p_f(x)|_{-\infty}^{+\infty} = 0$ under some regulation condition.

Note that a second-order gradient is present in (10.27), which may significantly raise the memory cost for large-scaled decoding. Therefore, this technique is practically not adopted in LLMs.