Basics of Energy Based Models

July 1, 2022

Useful papers - [SK21]

1 Introduction

Here we will discuss another approach to the generative modelling problem. So, our main problem is to approximate the distribution of the real data $P_{data}(X)$ having some samples from it, i.e. $x_1, ..., x_n$ - our training set. As always, we define some parameterized model $P_\theta(X)$ to approximate $P_{data}(X)$ by varying the parameter θ. The energy based approach assumes the following form of the model distribution:

$$P_\theta(X) = \frac{e^{-E_\theta(X)}}{Z_\theta} \tag{1}$$

It is possible to ask a fair question: why so? Before i give some of my stupid thoughts about this question, first of all, let's say that Z_θ is a normalizing constant

$$Z_\theta = \int e^{-E_\theta(X)} dX \tag{2}$$

Because of this we satisfy the main property of the probability distribution

$$\int P_\theta(X) dX = \int \frac{e^{-E_\theta(X)}}{Z_\theta} dX = \frac{\int e^{-E_\theta(X)} dX}{\int e^{-E_\theta(X)} dX} = 1 \tag{3}$$

Let's return to the question. The first answer is related with the physics. This distribution is the Boltzmann (Gibbs) distribution defining the probability that the system (any physic system: micro system like atom or macro sytem like gas) is in a state of with energy E and temperature T:

$$P_i = \frac{e^{\frac{-E_i}{k_B T_i}}}{Z} \tag{4}$$

As it can be seen from the formula, the probability that the system is in a low-energy state is much higher than the probability of being in a high-energy state. That is, $P \uparrow$ when $E \downarrow$. This reflects one of the basic physical principles: **every system seeks to minimum energy**. Any system, even a person :). For example, if we consider an atom, electrons inside it tend to return to their main state from energized condition. Moreover, the exponent in this formula is necessary to provide the positivity of the probability density (because of $e > 0$). Note that we could write any positive function in the numerator:

$$P_i = \frac{F_i}{Z} \tag{5}$$

But if we want the energy (to make physics analogy) instead of F_i, we have to write the exponent due to the fact that the energy might be lower that zero. Ok, we have physics analogies, besides that it is possible to find some computational advantages of such density presentation. The main advantage is that the energy E_θ can be presented by any neural network (i mean with any architecture, any form, any shape, ...). This is extremely good achievement compared to others generative models: in VAE we have to define encoder and decoder (that is, we resticted form of the model), in GAN it is necessary to consider generator and discriminator, in normalizing flows we restricted by type of neural network layers and so on. However, we have a serious problem with the normalizing constant Z_θ, i.e. this integral over the space of data from the neural network in exponent is intractable. Below different approaches to solve this problem will be considered (Maximum Likelihood Training with MCMC, Score matching, Noise Contrastive Estimation).

2 Maximum Likelihood Training with MCMC

The de facto standard for learning probabilistic models from i.i.d. data is maximum likelihood estimation:

$$\mathbb{L}(\theta) = \mathbb{E}_{P_{data}(X)} \log P_\theta(X) \tag{6}$$

Maximizing this by θ gives us approximation of the P_{data}. Lets consider it in more detail, because we still have a problem with Z_θ

$$\begin{aligned}\mathbb{L}(\theta) &= \int P_{data}(X) \log P_\theta(X) dX = \int P_{data}(X) \log \frac{e^{-E_\theta(X)}}{Z_\theta} dX \\ &= -\int P_{data}(X) E_\theta(X) dX - \int P_{data}(X) \log Z_\theta dX \\ &= -\int P_{data}(X) E_\theta(X) dX - \log Z_\theta \\ &= -\int P_{data}(X) E_\theta(X) dX - \log \int e^{-E_\theta(X)} dX\end{aligned} \tag{7}$$

Lets take the gradient, because we have to solve the optimization problem.

$$\begin{aligned}\frac{\partial \mathbb{L}}{\partial \theta}(\theta) &= -\int P_{data}(X) \frac{\partial E_\theta}{\partial \theta}(X) dX + \frac{1}{\int e^{-E_\theta(X)} dX} \int e^{-E_\theta(X)} \frac{\partial E_\theta}{\partial \theta}(X) dX \\ &= -\int P_{data}(X) \frac{\partial E_\theta}{\partial \theta}(X) dX + \int P_\theta(X) \frac{\partial E_\theta}{\partial \theta}(X) dX\end{aligned} \tag{8}$$

So, finally we have

$$\frac{\partial \mathbb{L}}{\partial \theta}(\theta) = -\mathbb{E}_{P_{data}(X)} \frac{\partial E_\theta}{\partial \theta}(X) + \mathbb{E}_{P_\theta(X)} \frac{\partial E_\theta}{\partial \theta}(X) \tag{9}$$

Lets discuss this final formula. First of all, we have a gradient of the energy by its parameters $\frac{\partial E_\theta}{\partial \theta}$. This term can be easily calculated by autograd frameworks (backward propagation). Also, we have the expectation by the data distribution. As always, we approximate it by Monte-Carlo estimation (we can do it, because we have samples - it is our training set):

$$\mathbb{E}_{P_{data}(X)} \frac{\partial E_\theta}{\partial \theta}(X) \approx \frac{1}{n} \sum_{i=1}^{n} \frac{\partial E_\theta}{\partial \theta}(x_i),\ i \sim \mathbb{U}(1, ..., n) \tag{10}$$

But, we have another expectation by the model distribution. Here the problem comes in. It is non trivial to obtain samples from the model distribution. So, we have to use Monte-Carlo Markov Chain to obtain these samples. Lets consider it in more detail.

2.1 Monte-Carlo Markov Chain

First of all, let's start with the Monte-Carlo. We used this approach above, to approximate the expectation. This is one of the main goal of the MC methods (approximation of the integral by a probability measure)

$$\mathbf{I} = \int P(X) f(X) dx \approx \frac{1}{n} \sum_{i=1}^{n} f(x_i) = \hat{\mathbf{I}},\ x_i \sim P(X) \tag{11}$$

Note that $\hat{I}$ is a random variable (as sum of random variables). We can consider some of its properties:

1. **Unbiasedness**. $\mathbb{E}\,\hat{\mathbf{I}} = \frac{1}{n}\sum \mathbb{E}\, f(x_i) = \mathbb{E}\, f(X) = \mathbf{I}$. This means that M-C estimation somehow converges to the true integral.

2. **Approximation error**. $\mathbb{D}\hat{\mathbf{I}} = \frac{1}{n^2}\sum \mathbb{D} f(x_i) = \frac{1}{n}\mathbb{D} f(X)$. Taking into account the central limit theorem, we can write that $\hat{\mathbf{I}} \sim \mathbb{N}(\hat{\mathbf{I}} | \mathbb{E}\, f(X), \frac{1}{n}\mathbb{D} f(X))$. Based on this, we can highlight the following important fact: The standard error of the normal distribution $\left(\sqrt{\frac{1}{n}\mathbb{D} f(X)}\right)$ is the **approximation error** of the initial integral. This follows from the fact that we want $\hat{\mathbf{I}} = \mathbf{I} = \mathbb{E}\, f(X)$, so if the variance of the normal distribution $(\mathbb{N}(\hat{\mathbf{I}} | \mathbb{E}\, f(X), \frac{1}{n}\mathbb{D} f(X)))$ is equal to zero, then we will obtain $\mathbb{E}\, f(X)$ during sampling (that's what we need). So, to reduce the approximation error we have to increase the number of samples n and also it would be nice if the function $f(X)$ is not fluctuating. The high fluctuation leads to high variance. If the function is constant ($\mathbb{D}(f(X)) = 0$), then we can take only one sample and our estimation will have 100% accuracy.

3. **Approximation error does not depend on the space dimension**. As can be seen from $\left(\sqrt{\frac{1}{n}\mathbb{D} f(X)}\right)$ that our error does not depend on the space dimension, we have only number of samples and properties of the function. It is really useful property, because if we take a look to errors of other numerical integration methods we will see:

   - Trapezoidal rule. Error $\sim n^{-\frac{2}{d}}$ (here n is the number of nodes of the quadrature formula)
   - Simpson's rule. Error $\sim n^{-\frac{4}{d}}$

- Monte-Carlo. Error $\sim n{-}\frac{1}{2}$

  As can be seen, the Monte-Carlo approach is losing to classical methods in the low dimension spaces. However, in the high dimension spaces (it is our case in the deep learning) it works much better.

So, the M-C is quite good. But we missed the most important thing: and can we even sample from $P(X)$. In the deep generative modelling (when $P(X)$ is a model) it is non-trivial problem. To do so, we have to consider Markov Chains.

2.1.1 Basics of Markov Chain

Let's give a definition of the Markov Chain:

Definition 1 *(Markov Chain)*

The Markov Chain is a ordered process of generation of the random variables. The joint distribution of the generated variables has the following form:

$$P(x_1, ..., x_n) = P_0(x_1)P_1(x_2|x_1)\dots P_n(x_n|x_{n-1}) \tag{12}$$

Here, P_0 is a base distribution, we sample x_1 from it. After, this sample start to participate in the process (Markov Chain). In other words, we move to another sample $x_2 \sim P_1(x_2|x_1)$ and so on. Doing this process several times we obtain x_n. As was said, our main goal is to produce samples from the distribution $P(X)$, so under some conditions it can be assumed that x_n is a sample from $P(X)$:). The first condition is a homogeneity:

Definition 2 *(Homogeneous Markov Chain)*

The Markov Chain is called homogeneous if and only if

$$P_i(x_i|x_{i-1}) = P(x_i|x_{i-1}), \forall i \tag{13}$$

It turns out that such Markov Chain (if $P(x_i|x_{i-1}) > 0, \forall x_i, x_{i-1}$) has a stationary state (distribution). In simple words it means that using the process of generating the random variables corresponding to Homogeneous Markov Chain at some point we will obtain samples from stationary distribution. More formally:

$$\begin{gathered}
x_1 \sim P_0(x_1) \\
x_2 \sim P_1(x_2) = \int P(x_2|x_1)P_0(x_1)dx_1 \\
x_3 \sim P_2(x_3) = \int P(x_3|x_2)P_1(x_2)dx_2 \\
\dots \\
x_n \sim P_{n-1}(x_n) = \int P(x_n|x_{n-1})P_{n-2}(x_{n-1})dx_{n-1} \\
x_{n+1} \sim P_{n-1}(x_{n+1}) = \int P(x_{n+1}|x_n)P_{n-1}(x_n)dx_n \leftarrow \text{stationary distribution} \\
x_{n+2} \sim P_{n-1}(x_{n+2}) = \int P(x_{n+2}|x_{n+1})P_{n-1}(x_{n+1})dx_{n+1} \leftarrow \text{stationary distribution}
\end{gathered} \tag{14}$$

It is pretty good, because if the stationary distribution is equal to our target distribution, then we will obtain samples from the target distribution. But how can we make sure that stationary distribution is equal to our target distribution. We can use the following theorem:

Theorem 2.1 *(Detailed Balance Equation)*

If the distribution $P(x)$ satisfies the following equation

$$P(x)P(x'|x) = P(x')P(x|x') \tag{15}$$

then the distribution $P(x)$ is stationary

So, it means that we to have check this equation for our target distribution, then the Markov Chain will converge to samples from the target distribution. Lets consider some of the MCMC algorithms.

2.1.2 Metropolis-Hastings method

Let $P(x) = \frac{\hat{P}(x)}{Z}$ - target distribution, $q(x|y) > 0$ - Markov Chain. (The support of $q(x|y)$ should match with the support of the $P(x)$ for the algorithm, im not sure that it is necessary). The Metropolis-Hastings algorithms has the following form:

1. Draw initial value $x_0 \sim P_0(x)$
2. for i = 1, ..., m , repeat:
   - Draw candidate $x^* \sim q(x^*|x_{i-1})$
   - Calculate $\alpha = \min\left(1, \frac{\hat{P}(x^*)q(x_{i-1}|x^*)}{q(x_*|x_{i-1})\hat{P}(x_{i-1})}\right)$
   - $x_i = x^*$ with probability α else $x_i = x_{i-1}$

Starting from some $n < m$ we will obtain samples from $P(x)$. We wander through the distribution area, choosing a point with a probability of improvement over the previous point ($\alpha = \frac{\hat{P}(x^*)}{\hat{P}(x_{i-1})}$ for symmetric case). If we only took steps with improvements, then sooner or later we would have stabilized in the distribution mode, and we want samples, but not optimal values. Lets consider a toy example, as in figure below. The target

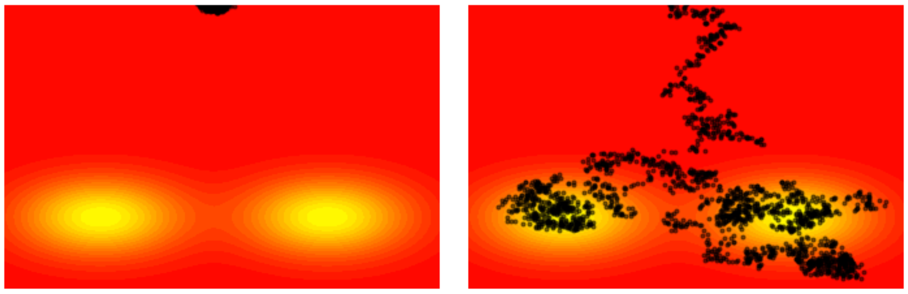

Figure 1: Target and base distribution (left). Samples from Markov Chain (right).

distribution is a mix of two gaussians. Base distribution is also gaussian, but it putted much higher than modes of the target distribution. As can be seen from the right figure, the Markov Chain converges to target distribution and gives nice samples.

2.1.3 Hamiltonian dynamics

References

[SK21] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.