Energy Based Models

Amaires@June 2024

1 Motivation

Essential to generative learning is the modeling of the probability density function (PDF) of given data. In theory, a deep neural network f_θ is capable of approxmating any function. f_θ in general, however, is not a valid PDF which has two fundamental requirements:

Non-negativity: $f_\theta(x) \geq 0$

Normalization: $\int f_\theta(x) = 1$

The non-negativity requirement is not hard to satisfy with simple transformations applied to f_θ. For example, $\exp(f_\theta(x))$ and $f_\theta^2(x)$ are both non-negative.

The normalization requirement, however, is much harder to satisfy. There are a few approaches to this problem.

1. Generative Adversarial Network (GAN) does not model the PDF or rely on the PDF for training. Instead, it only creates a model that can draw samples from.
2. Autoregressive models break the PDF into the product of a series of conditional PDFs.
3. Normalizing flow models use a sequence of bijective mappings to transform relatively simple distributions to the desired PDF.
4. Variational AutoEncoders (VAE) optimizes the upper bound of likelihoods. Like GAN, it does not produce a true PDF at the end either.

Energy Based Models (EBM) take a different approach. EBM only models a non-normalized function $E_\theta(x)$ with the expecation that the actual PDF will be

$$p_\theta = \frac{\exp(E_\theta(x))}{Z_\theta}, \text{ where } Z_\theta = \int \exp(E_\theta(x))$$

Z_θ, the normalization numerator and a function of θ but not x , is also called the *partition function*. EBM has some of its roots in statistical physics and hence the name Energy Based Models. $E_\theta(x)$, or in literature $-E_\theta(x)$, is called the *energy function*. Without the normalization requirement, and unlike autoregressive models and normalizing flow models, EBM can give E_θ more flexibity and potentially make it more powerful.

Since EBM only explicitly model E_θ, but not Z_θ or p_θ, so any task that strictly requires p_θ is out of the question. E_θ, however, is sufficient for comparing $p_\theta(x_1)$ and $p_\theta(x_2)$ since

$$p_\theta(x_1) > p_\theta(x_2) \iff \exp(E_\theta(x_1)) > \exp(E_\theta(x_2)) \iff E_\theta(x_1) > E_\theta(x_2) \tag{1}$$

This property is enough to enable a lot of practical deep learning applications such as object recognition, paining restoration and sequence labeling etc.

2 Sampling

Since EBMs do not explicitly model p_θ, how are samples drawn given E_θ?

The Metopolis-Hastings Markov Chain Monte Carlo (M-H MCMC) method described in Algorithm 1 is a relatively simple solution. The * step ensures that sufficient space of x is sampled and that the algorithm does not get stuck with local maximum. The M-H MCMC method works in theory, but can take a very long time to converge.

One obvious way to speed up the M-H MCMC method is to take advantage of the gradient of p_θ with respect to x and use it to find x with higher probability. That gradient still depends on Z_θ however as

$$\nabla_x p_\theta(x) = \frac{1}{Z_\theta} \exp(E_\theta(x)) \nabla_x E_\theta(x).$$

Fortunately the gradient of $\log p_\theta(x)$, also called the *score function* $s_\theta(x)$ only depends on $E_\theta(x)$ because

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x E_\theta(x) - \nabla_x \log Z_\theta = \nabla_x E_\theta(x).$$

The last step of the derivation works because Z_θ does not depend on x and hence $\nabla_x \log Z_\theta = 0$. The Langevin MCMC method, described in Algorithm 2, works exactly by levaraging $s_\theta(x)$. Again, the randomization in the * step helps the algorithm get out of local maximum and sample more space of x.

Algorithm 1 Metropolis-Hastings Markov Chain Monte Carlo method

x:=norm_random()
until convergence:
  y :=x+ϵ·norm_random()
  if $E_\theta(y) > E_\theta(x)$:
    $x := y$
  else:
    with probability $\exp(E_\theta(y) - E_\theta(x))$:
      $x := y$ [*]
return x

Algorithm 2 Langevin MCMC method

x :=norm_random()
until convergence:
  $x := x + \epsilon \cdot s_\theta(x) + \sqrt{2\epsilon}$·norm_random() [*]
return x

3 Training

There are multiple different ways of training EBMs. Some require sampling from the model being trained, and others do not.

3.1 Maximum Likelihood Method or Contrastive Divergence

Surprisingly, it is possible to conduct maximum likelihood optimization for EBM without modeling the PDF. Let's start with a little math:

$$\begin{aligned}\max E_{x\sim p(x)} \log p_\theta(x) &= \max_\theta E_{x\sim p(x)}[E_\theta(x) - \log Z_\theta] \\ &= \max_\theta [E_{x\sim p(x)} E_\theta(x) - \log Z_\theta]\end{aligned}$$

The likelihood gradient for updating θ is

$$\begin{aligned}\nabla_\theta [E_{x\sim p(x)} E_\theta(x) - \log Z_\theta] &= E_{x\sim p(x)} \nabla_\theta E_\theta(x) - \nabla_\theta \log Z_\theta \\ &= E_{x\sim p(x)} \nabla_\theta E_\theta(x) - \frac{1}{Z_\theta} \nabla_\theta Z_\theta \\ &= E_{x\sim p(x)} \nabla_\theta E_\theta(x) - \frac{1}{Z_\theta} \nabla_\theta [\int \exp(E_\theta(x)) dx] \\ &= E_{x\sim p(x)} \nabla_\theta E_\theta(x) - \frac{1}{Z_\theta} \int \nabla_\theta \exp(E_\theta(x)) dx \\ &= E_{x\sim p(x)} \nabla_\theta E_\theta(x) - \frac{1}{Z_\theta} \int \exp(E_\theta(x)) \nabla_\theta E_\theta(x) dx \\ &= E_{x\sim p(x)} \nabla_\theta E_\theta(x) - \frac{1}{Z_\theta} \int \exp(E_\theta(x)) \nabla_\theta E_\theta(x) dx \\ &= E_{x\sim p(x)} \nabla_\theta E_\theta(x) - \int \frac{1}{Z_\theta} \exp(E_\theta(x)) \nabla_\theta E_\theta(x) dx \\ &= E_{x\sim p(x)} \nabla_\theta E_\theta(x) - \int p_\theta(x) \nabla_\theta E_\theta(x) dx \\ &= E_{x\sim p(x)} \nabla_\theta E_\theta(x) - E_{x\sim p_\theta(x)} \nabla_\theta E_\theta(x)\end{aligned}$$

Note that the first term involves $p(x)$, the second term involves $p_\theta(x)$, and neither depends on Z_θ. Also note that here the likelihood gradient is with respect to θ, the parameters, not x. Don't confuse it with the score function, which is a gradient with respect to x. The likelihood gradient points in the direction where the energy function's gradient differs the most between real samples and model samples. This is probably where the name Contrastive Divergence got its name.

The big picture here is that even though the partition function Z_θ is not modeled, it is still possible to estimate the likelihood function's gradient and conduct maximum likelihood training. Each training step though requires drawing samples from the model being trained, which can be expensive, as described in Section 2.

3.2 Score Matching

If $\nabla_x p(x)$ and $\nabla_x p_\theta(x)$ are equal everywhere, then $p(x) = p_\theta(x)$+constant. Therefore, if $\nabla_x \log p(x)$ and $\nabla_x \log p_\theta(x)$ are equal everywhere, then $\log p(x) = \log p_\theta(x)$+constant. That constant difference can be removed since both $p(x)$ and $p_\theta(x)$ have to integrate to 1.

That is the key idea behind score matching, a method that matches the scores or score functions of two distributions everywhere as an alternative to maximum likelihood based training. The objective of score matching is to minimize the Fisher Divergence between $p(x)$ and $p_\theta(x)$:

$$\begin{aligned}\min_\theta FD(p(x), p_\theta(x)) &= \min_\theta \frac{1}{2} E_{x\sim p(x)} \left\|\nabla_x \log p(x) - \nabla_x \log p_\theta(x)\right\|_2^2 \\ &= \min_\theta \frac{1}{2} E_{x\sim p(x)} \left\|\nabla_x \log p(x) - \nabla_x E_\theta(x)\right\|_2^2\end{aligned}$$

We'll show how this objective can be manipulated to not dependent on the unknown $p(x)$ in the univariate case.

$$\begin{aligned}\min_\theta \frac{1}{2} E_{x\sim p(x)} \left\|\nabla_x \log p(x) - \nabla_x E_\theta(x)\right\|_2^2 &= \min_\theta \frac{1}{2} \int p(x)[\log' p(x) - E_\theta^{'}(x)]^2 \\ &= \min_\theta \frac{1}{2} \int p(x)[(\log' p(x))^2 + (E_\theta^{'}(x))^2 - 2\log' p(x) E_\theta^{'}(x)] \\ &= \min_\theta [\frac{1}{2} \int p(x)(\log' p(x))^2 + \frac{1}{2} \int p(x)(E_\theta^{'}(x))^2 - \int p(x) \log' p(x) E_\theta^{'}(x)] \qquad (2)\end{aligned}$$

The first term does not depend on θ, and can therefore be left out. The third term still has $\log' p(x)$ in it. Recall the integration by parts formula states that

$$\int_a^b u(x)v'(x)dx = u(x)v(x)|_a^b - \int_a^b u'(x)v(x)dx$$

and it can be used the rewrite the third term:

$$\begin{aligned}\int p(x) \log' p(x) E_\theta^{'}(x) &= \int p(x) \frac{1}{p(x)} p'(x) E_\theta^{'}(x) \\ &= \int p'(x) E_\theta^{'}(x) \\ &= p(x) E_\theta^{'}(x)|_{-\infty}^{+\infty} - \int p(x) E_\theta^{''}(x) \\ &= 0 - \int p(x) E_\theta^{''}(x) \qquad (3) \\ &= -E_{x\sim p(x)} E_\theta^{''}(x) \qquad (4)\end{aligned}$$

The derivation of (3) makes the very reasonable assumption that $\lim_{+\infty} p(x) = \lim_{-\infty} p(x) = 0$.

Eliminating the first term in (2), rewriting the second term in the expectation form, and subtituting the third term with (4), we have

$$\min_\theta [\frac{1}{2} E_{x\sim p(x)} (E_\theta^{'}(x))^2 + E_{x\sim p(x)} E_\theta^{''}(x)] = \min_\theta E_{x\sim p(x)} [\frac{1}{2} (E_\theta^{'}(x))^2 + E_\theta^{''}(x)].$$

The multivariate version of the objective can be shown to be

$$\min_\theta E_{x\sim p(x)} [\frac{1}{2} \left\|\nabla_\theta E_\theta(x)\right\|_2^2 + tr(\nabla_\theta^2 E_\theta(x))]$$

where the second term is the trace of the Hessian matrix of $E_\theta(x)$. Loosely speaking, the first term tries to find θ such that the samples x are the local maximums or minimums (with gradients as close to 0 as possible), and the second term tries to make sure it is actually local maximums (with second order gradients as negative as possible).

The score matching training method avoids the very expensive procedure of drawing samples from the model being trained. Its main expensive operation is the computation of the trace of the Hessian matrix. There are more research in this space that will be explored in future tutorials.

3.3 Noise Contrastive Estimation

Noise Contrastive Estimation (NCE) is another training method for EBMs without requiring drawing samples from the models being trained. Recall that in Generative Adversarial Networks (GAN) (`amaires.github.io/GAN`), given a fixed Generator G_ϕ, the optimal Discriminator D_θ's output is

$$D_{\theta^*}(x) = \frac{p(x)}{p(x) + p_\phi(x)}$$

The result holds if G_ϕ and p_ϕ are replaced with any static known noise distribution $p_n(x)$. That is

$$D_{\theta^*}(x) = \frac{p(x)}{p(x) + p_n(x)}$$

Note here n is not a parameter; it just means noise.

If D_θ's neural network is explicitly constructed as

$$D_\theta(x) = \frac{F_\theta(x)}{F_\theta(x) + p_n(x)}$$

then

$$D_{\theta^*}(x) = \frac{p(x)}{p(x) + p_n(x)} \simeq \frac{F_{\theta^*}(x)}{F_{\theta^*}(x) + p_n(x)}$$

solving it basically shows that $F_{\theta^*}(x) \simeq p(x)$ which also means $F_\theta(x)$ is automatically normalized if all stars are aligned. Now if $F_\theta(x)$ is replaced with an energy function based PDF function

$$\frac{\exp(E_\theta(x))}{Z}$$

where Z is an additional parameter, which is not guaranteed to be equal to $E_\theta(x)$'s partition function Z_θ, we have

$$D_{\theta,Z}(x) = \frac{\frac{\exp(E_\theta(x))}{Z}}{\frac{\exp(E_\theta(x))}{Z} + p_n(x)} = \frac{\exp(E_\theta(x))}{\exp(E_\theta(x)) + Zp_n(x)} \tag{5}$$

and

$$\frac{\exp(E_{\theta^*}(x))}{Z^*} \simeq p(x)$$

where E_{θ^*} would be our trained energy model.

With D_θ constructed as in (5), D_θ's optimization objective becomes

$$\begin{aligned}
&\max_{\theta,Z} E_{x\sim p(x)} \log D_{\theta,Z}(x) + E_{x\sim p_n(x)} \log(1 - D_{\theta,Z}(x)) \\
=&\max_{\theta,Z} E_{x\sim p(x)}[E_\theta(x) - \log(\exp(E_\theta(x)) + Zp_n(x)] + E_{x\sim p_n(x)}[\log(Zp_n(x)) - \log(\exp(E_\theta(x)) + Zp_n(x)]
\end{aligned}$$

3.4 Flow Contrastive Estimation

In theory, there are no requirements on the static noise distribution $p_n(x)$ for NCE. In practice, the closer $p_n(x)$ is to p (but not identical), the more effective NCE is. Flow Contrastive Estimation parameterizes $p_n(x)$ as $p_\phi(x)$ with a normalizing flow model because normalizing flow models are easy to sample and give tractable PDF. The discriminator is now modeled as

$$D_{\theta,Z,\phi}(x) = \frac{\exp(E_\theta(x))}{\exp(E_\theta(x)) + Zp_\phi(x)}$$

and the objective function is

$$\max_{\theta,Z} \min_{\phi} E_{x\sim p(x)} \log(D_{\theta,Z,\phi}(x)) + E_{x\sim p_\phi(x)} \log(1 - D_{\theta,Z,\phi}(x))$$