StatML
Generative Models

2024-04-22

Contents

Notes at `akyildiz.me/teaching/cdt/notes/Lecture-1.pdf`.

1 Monday 22 April 2024: Imperial

Today: MCMC sampling and Energy based models?

1.1 Recap

Take home: *when we don't really know which distribution we are targeting, we can trade off computational cost for accuracy.*

1.1.1 Sampling problem

Target distribution $\pi(x) = \frac{\Pi(x)}{Z}$ where Z is unknown.
What's the difference between sampling and generative modelling? In sampling, we have a target distribution, in generative modelling we want to generate samples but we don't have the target available. First need to get information on the object and then run the sampler. Want something such that $X_1, X_2, \ldots, X_n \xrightarrow{\infty} X_\infty \sim \pi$.

1.1.2 Metropolis-Hastings

Propose and accept as a function of acceptance ratio α.

- Propose $X' \sim q(\cdot|X_{n-1})$.
- $\alpha(X', X_{n-1}) = \min\left(1, \frac{\Pi(X')q(X_{n-1}|X')}{\Pi(X_{n-1})q(X'|X_{n-1})}\right)$.

The problem with MH is the choice of proposal and dealing with correlations.

1.1.3 Metropolis adjusted Langevin

$$X' = X_{n-1} + \gamma \nabla \log \Pi(X_{n-1}) + \sqrt{2\gamma} Z, \tag{1}$$

we can also write $U = \log \Pi$.
In super-high dimensional problems computing α is hard and it may just be close to 0 most of the times. As such, in machine learning, **unadjusted LA (ULA)**:

$$X_n = X_{n-1} - \gamma \nabla U(X_{n-1}) + \sqrt{2\gamma} Z \tag{2}$$

which targets π_γ. Note that this is a discretisation of a stochastic differential equation:

$$dX_t = -\nabla U(X_t)dt + \sqrt{2} dB_t. \tag{3}$$

why is this a good proposal? The stationary measure of the SDE is the target distribution: $\pi(x) \propto \exp U(x)$.

We can also prove that the "distance" between the discretisation and the SDE is low?

$$W_2(\pi_{\gamma_n}, \pi_n^\gamma) \leq O(\gamma^{\frac{1}{2}}) \tag{4}$$

can control the error by the square root of the step size γ. Further,

$$W_2(\pi_{\gamma_n}, \pi) \leq e^{-\mu\gamma_n} \left(\frac{1}{\sqrt{(n)} + \|x - x^n\|} \right). \tag{5}$$

(The correct result is in the slides. from Durmus and Moulines)

1.1.4 Annehaling

Start by looking at the modification of eq. (3), where we change the noise term and instead write:

$$dX_t = -\nabla U(X_t)dt + \sqrt{\frac{2}{\beta}} dB_t, \tag{6}$$

where the SDE has a new stationary measure:

$$\pi_\beta(x) \propto \exp(-\beta U(x)). \tag{7}$$

and the stationary meausre concentrates on the minima of E? as $\beta \to \infty$.

As such, β controls the "peakiness" of the distribution (e. g. *simulated annehaling*).

1.2 Enery Based Models and Variational Autoencoders

mem vs gen difference between $\hat{p}_{\text{data}} = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ and p_{data}.

pot en $\pi(x) = \exp(-U(x))$

1.3 Energy Based Models

We would like to estimate p_{data} from $\hat{p}_{\text{data}}$: classical density estimation problem **but** very high dimensions(so e. g. kernel doesn't work).
What would be a sensible parametric model for p_{data}? Neural networks make sense to approximate continuous functions, so for example write $(U_\theta(x))$. (rationale: universal approximators theorem guarantees there exists a NN which can approximate a continuous function arbitrarily well).

Let $U \colon X \to \mathbf{R}$ and U_θ be a NN. Then:

$$\|\pi - \pi_\theta\|_{TV} \leq \frac{m(X)}{2} \|U - U_\theta\|. \tag{8}$$

So EBM are universal approximators on bounded spaces. (Atchade 2023). **Training EBM** is hard and there is a VAST literature. We review MLE and Score matching and variations of these

1.4 Maximum Likelihood Estimation

Want to maximise the expected likelihood of the data under the model. The *expected* log-likelihood of the data is given by:

$$l(\theta) := \mathbf{E}_{x\sim p_{\text{data}}} \left[\log p_\theta(x)\right]. \tag{9}$$

Maximising this is equivalent to minimising the KL divergence between the model and the data distribution.

if the data are diracs, the KL is not defined.

If we expand, we get:

$$l(\theta) = \int \log \pi_\theta(x) p_{\text{data}}(x) dx = \dots \tag{10}$$

depends on an integral we can't compute ...

Assuming we can use Fubini:

$$\nabla_\theta \log Z_\theta = -\mathbf{E}_{x\sim p_\theta} \left[\nabla_\theta U_\theta(x)\right]. \tag{11}$$

This is easy to prove through derivative of log, swap integral and recognize pdf. Thus:

$$\nabla_\theta l(\theta) = \mathbf{E}_{x\sim p_{\text{data}}} \left[\nabla_\theta U_\theta(x)\right] - \mathbf{E}_{x\sim p_\theta} \left[\nabla_\theta U_\theta(x)\right], \tag{12}$$

where the first term can be "filled in" with the empirical data.

The idea is to use MCMC to sample from π_θ to approximate the second term. At iteration k the expectation needs to be approximated. For this, we run a separate ULA chain at each k. General algorithms on the slides which use stochastic gradients.

Cases: *CD-1*, *PCD* (persistent Contrastive Divergence), *Short-Run MCMC* (Nijkamp et al. 2019).

Exercise: Understand the differences between these models.

1.5 Latent variables EBMs and MLE.

Until now, we haven't used any latent variables Let $p_\theta(x, z)$ be a probabilistic model with parameters θ, data x and latent variables z.

Given the data x, the typical goal is to build MLE of the parameters θ by maximising the marginal likelihood:

$$\theta \in \arg\max_\theta \log p_\theta(x). \tag{13}$$

So the difference with the previous case is that we have to marginalise over the latent variables, and so we are maximising over an integral as well.

In literature for LVMs, the standard approach is the EM algorithm:

- E-step: computing an expectation $Q(\theta, \theta_n) = \mathbf{E}_{p_{\theta_n}(z|x)} \left[\log p_\theta(x, z)\right]$.
- M-step: maximising the expectation.

To compute E-Step, we would run ULA for a pre-determined number of steps. We can use *Fisher's identity*:

$$\nabla_\theta \log p_\theta(x) = \mathbf{E}_{p_{\theta_n}(z|x)} \left[\nabla_\theta \log p_\theta(x, z) \right]. \tag{14}$$

(same as before but with "log-trick" in reverse).

SOUL method ...

e. g. Consider the latent variable EBM model:

$$p_\theta(x, z) = p_\beta(x|z) p_\alpha(z), \tag{15}$$

where

$$p_\alpha(z) = \frac{e^{U_\alpha(z)} p_0(z)}{Z_\alpha}. \tag{16}$$

Can you derive MLE training algorithm for this model?

So what is the intuition here? If we use Fisher's identity, what do we get?

$$\begin{aligned}
\nabla_\theta \log p_\theta(x) &= \mathbf{E}_{p_{\theta_n}(z|x)} \left[\nabla_\theta \log p_\theta(x, z) \right] && (17)\\
&= \mathbf{E}_{p_\theta(z|x)} [\nabla \log p_\theta(x|z) + \nabla \log p_\alpha(z)] && (18)\\
&= \mathbf{E}_{p_\theta(z|x)} [\nabla \log p_\theta(x|z) + \nabla U_\alpha(z)] - \nabla \log Z_\alpha && (19)\\
&= \mathbf{E}_{p_\theta(z|x)} [\nabla \log p_\theta(x|z) + \nabla U_\alpha(z)] - \mathbf{E}_{p_\alpha(z)} [\nabla U_\alpha(z)]. && (20)
\end{aligned}$$

Now we have to compute two expectations and we will set up two MCMC schemes to approximate these. My problem was understanding what we can compute and what we cannot. When we don't know, use trick to translate into something estimable

Solution:

- Learning EBM prior via short-run MCMC.
- **Mini-batch** Sample observed examples $x_{i i=1}^{m}$.
- **Prior sampling** ...

aaaaa

1.6 Score Matching

- *Idea that made diffusion models work.*
- The idea is to minimise the difference between the score of the model and the score of the data.
- Similarity: MLE minimises KL divergence, while score matching minimises Fisher divergence.

Recall ULA(eq. (2)): notice we only need the gradient of the (unnormalised) log density for sampling: $\nabla \log \Pi(X_t^\gamma)$.

Score matching methods are based on Fisher divergence: differences in grad-logs:

$$F(\pi_1||\pi_2) = \frac{1}{2}\mathbf{E}_{\pi_1}[||\nabla \log \pi_1 - \nabla \log \pi_2||^2]. \tag{21}$$

We are interested in computing θ where:

$$\theta \in \arg\, \min_\theta F(\pi_{\text{data}}||\pi_\theta). \tag{22}$$

but again, we do not know π_{data} and we do not know the grad-log. However, we can rewrite:

$$F(\pi_{\text{data}}||\pi_\theta) = \mathbf{E}_{p_{\text{data}}}\left[\text{Tr}\nabla^2 \log \pi_\theta(X) + \frac{1}{2}\|\nabla \log \pi_\theta(X)\|^2\right]. \tag{23}$$

This means we don't need to know p_{data} to compute the Fisher divergence.
Proof comes from expansion, realising one term=0, log-trick, integration by part,

What you are estimating as a score is a vector field that points towards the "mass" of the data. Image from Murphy: 2d samples from score-matching.

Mentioned one problem with score-matching not dealing well with parts where you don't see data. → you are not going to estimate well the proportions between two modes.

Possible solutions:

- Noise your data. $p^\sigma_{\text{data}}(x) = \int p_{\text{data}}(x) \cdot \ldots$.

Noising implies perturbing densities such as they have better properties ("no 0 density areas between pdfs"). There is a version of noising score matching ...

The key discovery here is that there will be a way of noising the data which will lend itself to going "the other way" and denoising noised distributions.

1.7 Variational Autoencoders

2 Wednesday 24 April 2024: Oxford