Introduction to training and sampling

Justin T Chiu

July 27, 2023

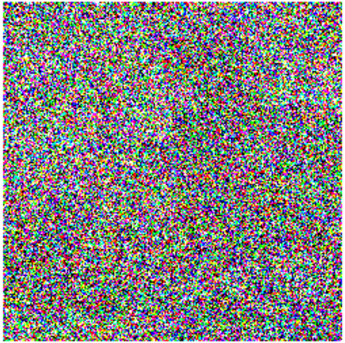
Goals

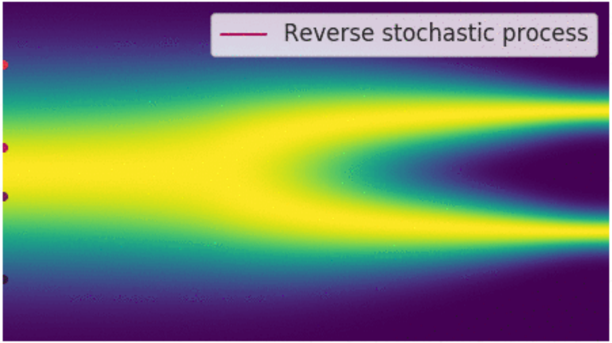
1. How do we train?

2. How do we sample?

Setup

- Sampling requires solving the reverse SDE
- Solving the reverse SDE requires training a score model





Training

Sampling

Recap

Problem setup: Energy-based models

- Observations from true model $x \sim p^*(x)$
- Ideally: Learn a model $p(x)$ that's close to $p^*(x)$
  - Capture uncertainty / variability over x
- Focus: Sampling dope images

EBM

- Globally normalized over images x

$$p(x) = \frac{\exp(E(x))}{Z}$$
$$Z = \int_x \exp(E(x))$$

- Computation of the partition function Z is hard
  - Integrate $E(x)$ over all possible images
- Don't need that to sample. Instead do Langevin dynamics
  - Start with random image x, follow score $\nabla \log p(x)$ to improve sample
  - Doesn't depend on Z

KL divergence to Fisher divergence

- KL divergence

$$E_{p^*(x)} \log \frac{p^*(x)}{p(x)} = E_{p^*(x)} \log p^*(x) - E_{p^*(x)} \log p(x)$$

- Insight: go from functional equality to equality of gradient (Fisher divergence)

$$E_{p^*(x)} \left\| \nabla_x \log \frac{p^*(x)}{p(x)} \right\|_2^2 = E_{p^*(x)} \left\| \nabla_x \log p^*(x) - \nabla_x \log p(x) \right\|_2^2$$

Score modeling

- Stein score $s(x) = \nabla_x \log p(x)$

$$E_{p^*(x)} \left\| \nabla_x \log p^*(x) - \nabla_x \log p(x) \right\|_2^2 = E_{p^*(x)} \left\| \nabla_x \log p^*(x) - s(x) \right\|_2^2$$

- Parameterize $s(x)$ directly instead of $p(x)$, avoid computing Z

New issues in score matching

$$E_{p^*(x)} \|\nabla_x \log p^*(x) - s(x)\|_2^2$$

1) Solved: Cant compute $p(x)$ b/c of Z => model Stein score $s(x) = \nabla_x \log p(x)$

2) Unknown p^*: Dont know $p^*(x)$ or its score

3) Covariate shift: $E_{p^*(x)}$ is problematic because of covariate shift

Avoiding p^*: Implicit score matching

- Rewrite the explicit score matching objective to avoid p^*. See Volo's slides (12, p. 14)

$$E_{p^*(x)}\left[\|\nabla_x \log p^*(x) - s(x)\|_2^2\right] \approx E_{p^*(x)}\left[\frac{1}{2}\|s(x)\|_2^2 + tr(\nabla_x s(x))\right]$$

- Second term is nasty: $s(x) \in R^d$, $\nabla_x s(x) \in R^{d \times d}$
- Solution: Use Hutchinson's trace estimator

$$E_{p^*(x)}\left[\frac{1}{2}\|s(x)\|_2^2 + tr(\nabla_x s(x))\right] = E_{v\sim N(0,I_d)} E_{p^*(x)}\left[\frac{1}{2}\|s(x)\|_2^2 + v^T \nabla_x s(x)) v\right]$$

- Easy to implement with pytorch

Covariate shift

- Sample via Langevin dynamics := Start with random point and follow score + noise
  - Score is trained on examples drawn from $p^*(x)$
  - Score is bad on regions of low $p^*(x)$, eg random points
  - Slow mixing and bad samples

Solution to cov shift

- Solution: sample perturbed $x \sim p^*(x)$ with multiple noise scales $\{\sigma_i\}$
  - Interpretation: Data augmention + smooth out samples
  - Need to have score model condition on noise $s(x; \sigma_i)$