Generative Models

Saeed Saremi

Assigned reading: 20.3[1]

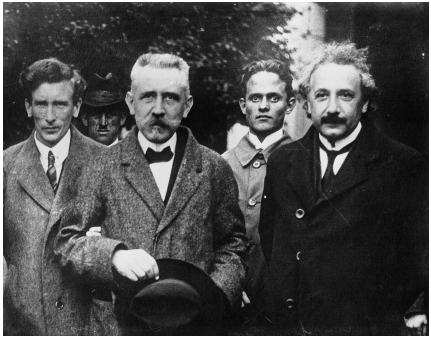
November 12, 2024

[1] The primary source of material is the lecture notes.

BROAD OUTLINE

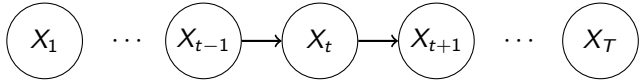
Lecture 1: LANGEVIN MCMC uses the SCORE FUNCTION

Lecture 2: Learn the SCORE FUNCTION via DENOISING



SUMMARY OF LECTURE 1

- MCMC is about coming up with Markov chains



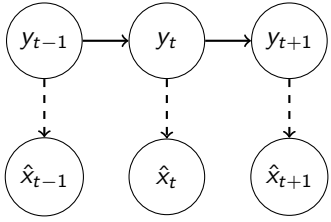
  such that $X_T \sim p(x)$ for large-enough T for any initialization $X_1 \sim \pi(x)$.
- A celebrated example is Langevin MCMC:

$$x_{t+1} = x_t + h \underbrace{\nabla \log p(x_t)}_{\text{score function}} + \sqrt{2h}\,\varepsilon_t,\ \varepsilon_t \sim N(0, I),$$

  which we can run by choosing a step size h and knowing the score function.
- We played with tuning Langevin MCMC for 1D and 2D Gaussians.

OUTLINE OF LECTURE 2

- Langevin MCMC likes smooth distributions which we can "engineer" using Gaussian noise
- Tweedie-Miyasawa formula for denoising
  - › Learning the score function via denoising
- Walk-Jump Sampling:[2]

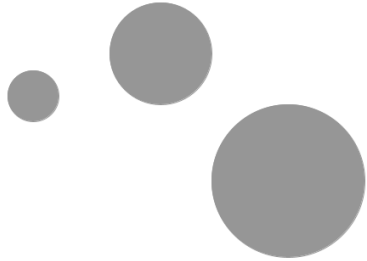


[2]Saremi, S., & Hyvärinen, A. (2019). "Neural Empirical Bayes," *Journal of Machine Learning Research*.

☹ GETTING TRAPPED in a MODE

😐 remember our discussion from the previous lecture that ill-conditioned multivariate Gaussians are difficult to sample from using Langevin MCMC (🎞)

☹ However, the biggest challenge for sampling is when the distribution is concentrated in (many) pockets, a.k.a. *modes*, separated by vast empty spaces. The large regions with small probability mass make navigating the space via Langevin Markov chains very slow.



MIXTURE OF GAUSSIANS

Let's consider a simple example of mixtures of Gaussians in 1D:

$$p(x) = \sum_{k=1}^{K} \pi_k \exp\left(-\frac{(x-\mu_k)^2}{2\tau^2}\right)$$

It's straightforward to find the score function:

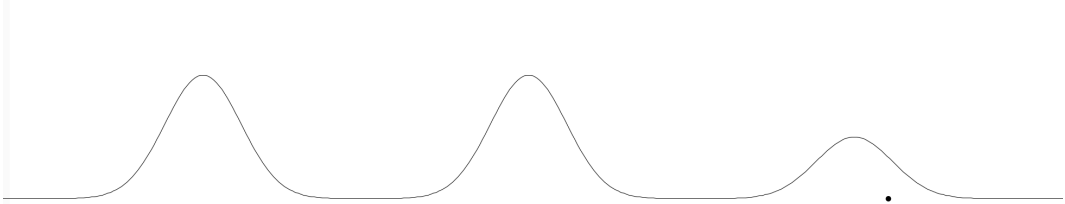
$$\nabla \log p(x) = \frac{1}{\tau^2} \frac{\sum_{k=1}^{K} \pi_k \cdot (\mu_k - x) \exp\left(-\frac{(x-\mu_k)^2}{2\tau^2}\right)}{\sum_{k=1}^{K} \pi_k \exp\left(-\frac{(x-\mu_k)^2}{2\tau^2}\right)}$$

Now if x is close to a μ_{k^*} and if modes are well separated, the score function is approximately

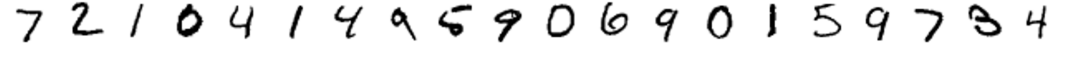
$$\nabla \log p(x) \approx \frac{\mu_{k^*} - x}{\tau^2},$$

which is the same as a unimodal Gaussians: the chain gets TRAPPED in the mode $k = k^*$.

LANGEVIN CHAIN (MIXTURE OF GAUSSIANS)



📢 GAUSSIAN NOISE SMOOTHS ANY DISTRIBUTION, MAKING IT EASIER TO SAMPLE.



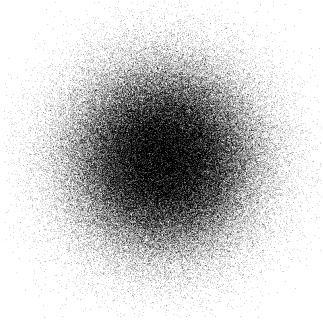
(a) X

(b) $Y = X + \sigma N$

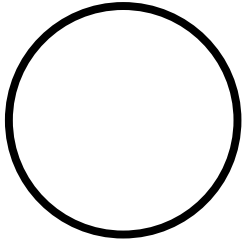
GEOMETRIC VIEW

GAUSSIAN DISTRIBUTION IN HIGH DIMENSIONS

- $X_{1:d} \sim \mathcal{N}(0, \sigma^2 I_d)$ in high dimensions is concentrated on S_{d-1} with radius $\sigma\sqrt{d}$
- one-line proofs:
  - $\|X_{1:d}\|^2 = \sum_{i=1}^{d} X_i^2 = d \cdot \frac{1}{d} \sum_{i=1}^{d} X_i^2 \approx d\,\mathbb{E}[X^2] = d\sigma^2 \Rightarrow \|X_{1:d}\| \approx \sigma\sqrt{d}$
  - `x = torch.randn(n, d).pow(2).sum(1).sqrt()/math.sqrt(d)`



(a) $d = 2$



(b) $d \gg 1$

📣 ISOTROPIC GAUSSIAN NOISE MAKES THE DATA MANIFOLD MORE SPHERICAL.

ALGEBRAIC VIEW[3]

[3] This part of the lecture is beyond the scope of the course. You will not be tested on this.

ADDITIVE NOISE = CONVOLUTION

- Consider additive (Gaussian) noise

$$Y = X + N, \text{ where } N \sim \mathcal{N}(0, \sigma^2 I).$$

- Equivalently,

$$y|x \sim \mathcal{N}(x, \sigma^2 I)$$

Show

$$p_Y(y) = \frac{1}{(2\pi\sigma^2)^{d/2}} \int p_X(x) \, exp\Big(-\frac{\|y-x\|^2}{2\sigma^2}\Big) dx$$

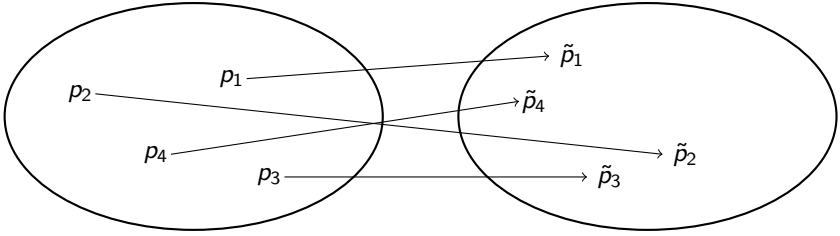
This is written more compactly, using the convolution notation:

$$p_Y = p_X * p_N$$

- Recall the definition of Fourier transform

$$\tilde{p}(\omega) = \int e^{i\omega^\top x} p(x) dx$$

- Fourier transform is a bijection:



- Fourier transform of a probability distribution is named characteristic function.

ADDITIVE GAUSSIAN NOISE (THE FOURIER ANGLE)

- Recall the definition of convolution of two functions $p_3 = p_1 * p_2$:

$$(p_1 * p_2)(y) = \int p_1(x) p_2(y - x) dx$$

- In Fourier space, convolution=multiplication: $\tilde{p}_3(\omega) = \tilde{p}_1(\omega)\tilde{p}_2(\omega)$
- The Fourier transform of the Gaussian distribution is of particular interest!

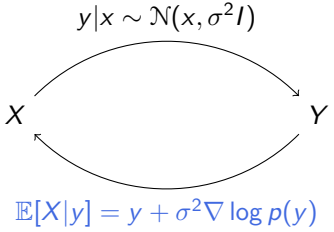
$$\tilde{p}_N(\omega) = \exp\left(-\frac{\sigma^2}{2}\|\omega\|^2\right)$$

- Gaussian noise $Y = X + N$ has the effect of smoothing out high frequencies in p_X:

$$\tilde{p}_Y(\omega) = \tilde{p}_X(\omega) \exp\left(-\frac{\sigma^2}{2}\|\omega\|^2\right),$$

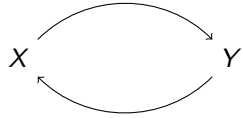
since for large $\sigma\|\omega\| \gg 1$, we have $\tilde{p}_Y(\omega) \ll \tilde{p}_X(\omega)$.

DENOISING via BAYES ESTIMATORS



HERBERT ROBBINS (1951, 1952, 1956)

$$y|x \sim \mathcal{N}(x, \sigma^2 I)$$



$$\mathbb{E}[X|y] = y + \sigma^2 \nabla \log p(y)$$

AN EMPIRICAL BAYES APPROACH TO STATISTICS

HERBERT ROBBINS
COLUMBIA UNIVERSITY

Let X be a random variable which for simplicity we shall assume to have discrete values x and which has a probability distribution depending in a known way on an unknown real parameter Λ,

(1) $$p(x|\lambda) = Pr[X = x|\Lambda = \lambda],$$

Λ *itself being a random variable* with *a priori* distribution function

(2) $$G(\lambda) = Pr[\Lambda \leqq \lambda].$$

The unconditional probability distribution of X is then given by

(3) $$p_G(x) = Pr[X = x] = \int p(x|\lambda)\, dG(\lambda),$$

Robbins invented stochastic optimization, bandits, and empirical Bayes, in the 50s.

- Denote $X \in \mathbb{R}^d$ to be the clean data
- We do not observe clean data but only its noisy version[4]
- We know the noise model/kernel/conditional probability, e.g.,

$$Y|x \sim \mathcal{N}(x; \sigma^2 I)$$

- We like to estimate the clean data X given a noisy observation $Y = y$ by least-squares

$$\mathcal{L}(\varphi(y)) = \int \|x - \varphi(y)\|^2 p(x)p(y|x)dx$$

Recall from Lecture 18 that the minimizer of the above is $\hat{x}(y) = \mathbb{E}[X|y]$:

$$\hat{x}(y) = \frac{\int x\, p(y|x)p(x)dx}{\int p(y|x)p(x)dx}.$$

[4] The setting of [Robbins1956]

It turns out for many types of noise models $\hat{x}(y)$ can be written purely in terms of distribution of noisy data p_Y:

$$p(y) = \int p(y|x)p(x)dx.$$

- For Gaussian noise

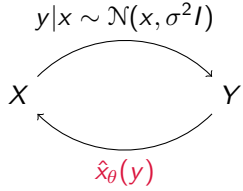
$$p(y|x) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\Big(-\frac{1}{\sigma^2}\|y - x\|^2\Big),$$

we arrive at[5]

$$\hat{x}(y) = \frac{\int x\, p(y|x)p(x)dx}{\int p(y|x)p(x)dx} = y + \sigma^2 \nabla \log p(y).$$

[5] The key to this derivation is $\sigma^2 \nabla_y p(y|x) = (x - y)p(y|x)$

LEARNING THE SCORE FUNCTION



- We can use a neural network (with parameters θ) for denoising:

$$\hat{x}_\theta : \mathbb{R}^d \to \mathbb{R}^d,$$

which we optimize with SGD using the following denoising objective:

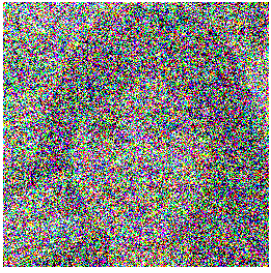
$$\mathcal{L}(\theta) = \mathbb{E}_{x \sim p_X, \varepsilon \sim \mathcal{N}(0,I)} \|x - \hat{x}_\theta(x + \sigma\varepsilon)\|^2.$$

- learning the optimal denoiser = approximating the score function $\nabla \log p(y)$:

$$\nabla \log p(y) \approx \frac{1}{\sigma^2}(\hat{x}_\theta(y) - y)$$

📢 NEURAL NETWORKS ARE EXCELLENT DENOISERS!

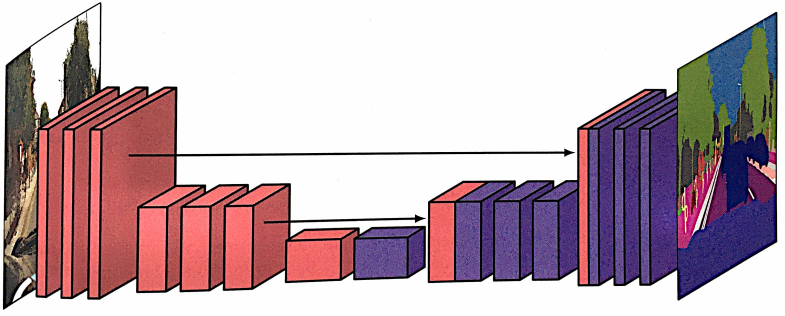
(a) x



(b) $y = x + \sigma\,\varepsilon,\ \varepsilon \sim \mathcal{N}(0, I)$

(c) $\hat{x}_\theta(y)$

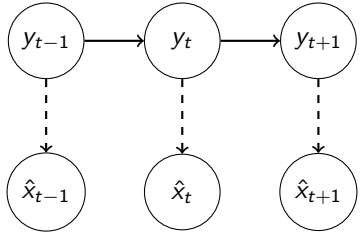
THE U-NET ARCHITECTURE[6]



[6]Ronneberger, Fischer, & Brox (2015). *U-net: Convolutional networks for biomedical image segmentation.*

WALK-JUMP SAMPLING

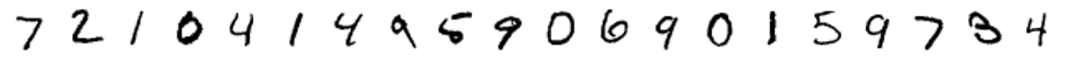
$$y_{t+1} = y_t + h \underbrace{\frac{1}{\sigma^2}(\hat{x}_\theta(y_t) - y_t)}_{\text{score function}} + \sqrt{2h}\,\varepsilon_t,\ \ \varepsilon_t \sim N(0, I),$$



$$\hat{x}_t = \hat{x}_\theta(y_t)$$

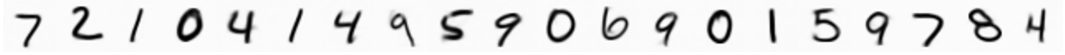
⚠ walks and jumps are decoupled! ⚠

MNIST EXAMPLE (TRAINING)



(a) x_i

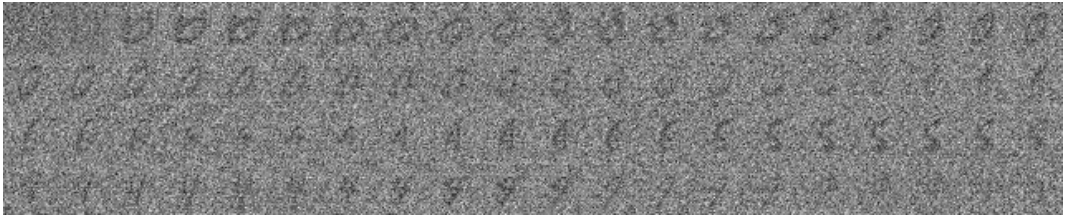
(b) $y_i = x_i + \sigma\,\varepsilon_i, ; \varepsilon_i \sim \mathcal{N}(0, I)$



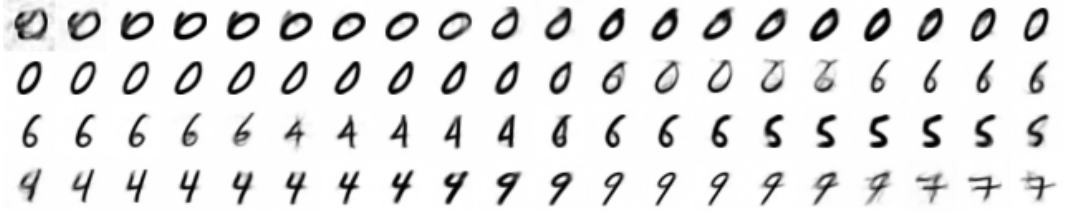
(c) $\hat{x}_\theta(y_i)$

MNIST EXAMPLE (SAMPLING)

Below, it's a chain in real time (all steps are shown). Step size is set to $h = 1$.



(a) WALK



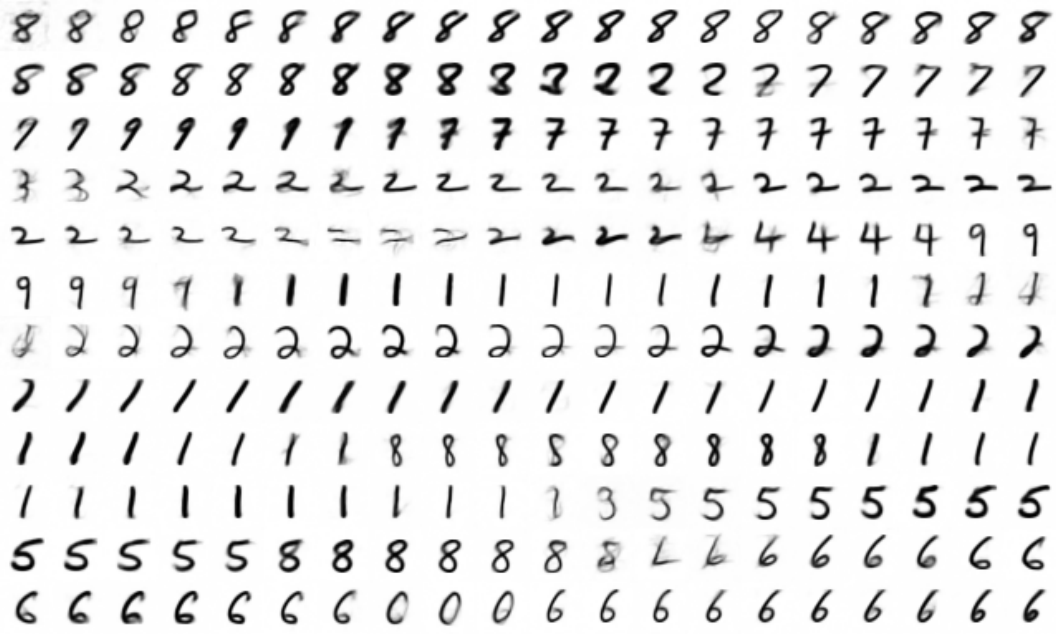
(b) JUMP

MORE LANGEVIN CHAINS

(a) CIFAR-10

(b) FFHQ-256

MIXING TIME-SAMPLE QUALITY TRADE-OFF



MIXING TIME-SAMPLE QUALITY TRADE-OFF → DENOISING DIFFUSION

- large noise: sampling is easier, but the distribution of denoised samples $\hat{p}_X$ is "far from" p_X
- small noise: the distribution of denoised samples is "close to" p_X, but sampling is harder.
- We can address this problem by learning many score functions.
- This leads to denoising diffusion models.