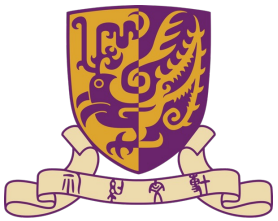


AIR5101 / CIE6021

Generative AI

Lecture 4: Energy-based Models

Instructor: Zhen Liu

Spring 2025, CUHK-Shenzhen

Agenda

- Energy-based models (EBMs) and their training
- Score Matching
- Bridge to Diffusion Models

Recall: MLE Training

MLE for general distributions

$$\max_{\theta} \mathop{\mathbb{E}}_{x \sim \mathcal{D}} \log p_\theta(x)$$

Explicit constraints:

- Normalized density: $\int_x p_\theta(x) dx = 1$
- Non-negative $p_\theta(x)$

Challenge in Modeling Distributions

Easy to guarantee non-negative $p_\theta(x)$

Simply use ReLU-like activations on the scalar output

Other choices:

- Square function
- Absolute value
- etc.

The hard one: $\int_x p_\theta(x)dx = 1$

Challenge in Modeling Distributions

Implications of $\int_x p_\theta(x)dx = 1$

- Increasing the likelihood of some point = decreasing the likelihood elsewhere

How can you find a neural net of which outputs are guaranteed to sum to one?

Possible solution:

to compute the normalization constant on unnormalized $p_\theta(x)$

Energy-based Model

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

- E(x) is called the energy function
- Z also known as the partition function
- Roots in statistical mechanics

Energy-based Model

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

Why exponential?

- Log-probability captures large variations of probability
- Connections to exponential families $p(x|\theta) = h(x)\exp\left(\eta(\theta)^T T(x) - A(\theta)\right)$
  - Captures lots of common distributions
- Connections to Boltmann distribution in physics – energy of a state

$$p_i \propto \exp\left(-\frac{\varepsilon_i}{kT}\right)$$

Energy-based Model

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

Pros:

- Learning an energy without worrying the normalization constant
- Flexible enough to represent complex distributions

Cons:

- Hard to sample from
- Evaluating the likelihood is hard (requires computing Z)
- No latent features learned

Energy-based Model

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

Curse of Dimensionality

- Z = volume of the numerator
- Cost of computing a volume in high dimensional space scales exponentially with the number of dimensions

But in many cases, you do not care about the exact Z

Energy-based Model

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

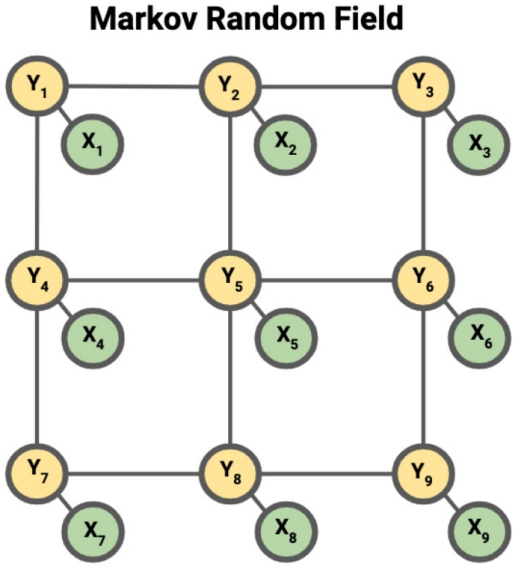
Comparing likelihoods of two points

$$p(x)/p(x') = \exp(E(x') - E(x))$$

No explicit Z involved

Example: Anomaly detection

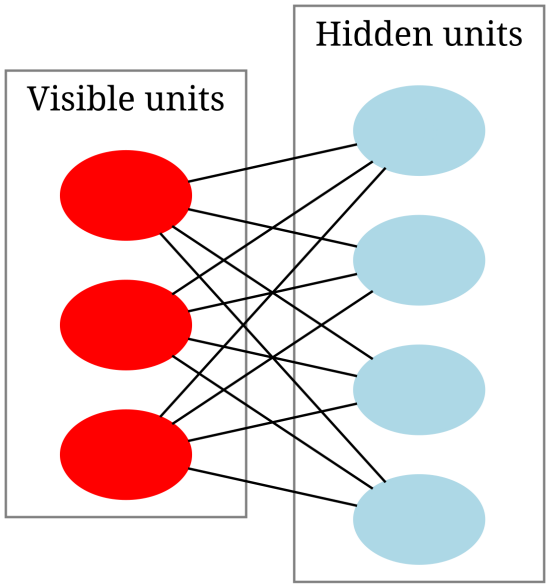
Example – Ising Model



$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i \psi_i(x_i, y_i) + \sum_{(i,j) \in E} \psi_{ij}(y_i, y_j) \right)$$

https://deepgenerativemodels.github.io/assets/slides/cs236_lecture11.pdf

Example – Restricted Boltzmann Machine (RBM)

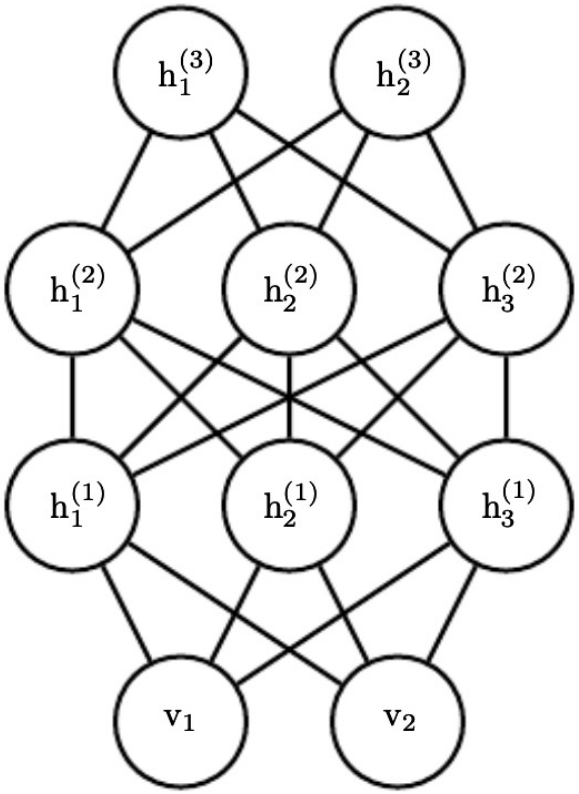


Learning latents h from inputs v

$$E(v, h) = -a^{\mathrm{T}} v - b^{\mathrm{T}} h - v^{\mathrm{T}} W h$$

“Restricted”: a bipartite graph, instead of a fully connected one

Example – Deep Belief Network



Deep Boltzmann Machine (DBM): Stacks of RBMs

Algorithm:

1. Train the shallowest untrained RBM
2. Freeze the trained ones
3. Repeat

After training DBM: deep features on the top

Can concat a linear layer and finetuning on a labeled dataset with backprop

This is how people trained deep nets around 2008!

Training an EBM

$$\nabla_\theta \mathop{\mathbb{E}}_{x \sim \mathcal{D}} \log p_\theta(x)$$

Training an EBM

$$\begin{aligned}
&\nabla_\theta \mathop{\mathbb{E}}_{x \sim \mathcal{D}} \log p_\theta(x) \\
&= - \nabla_\theta \mathop{\mathbb{E}}_{x \sim \mathcal{D}} E(x) - \nabla_\theta \log Z(\theta)
\end{aligned}$$

Training an EBM

$$
\begin{aligned}
&\nabla_\theta \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \log p_\theta(x) \\
=& -\nabla_\theta \mathop{\mathbb{E}}_{x\sim\mathcal{D}} E(x) - \nabla_\theta \log Z(\theta) \\
=& -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)}
\end{aligned}
$$

Training an EBM

$$
\begin{aligned}
&\nabla_\theta \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \log p_\theta(x) \\
=& -\nabla_\theta \mathop{\mathbb{E}}_{x\sim\mathcal{D}} E(x) - \nabla_\theta \log Z(\theta) \\
=& -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)} \\
=& -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\int_X \nabla_\theta \exp(-E_\theta(x))dx}{Z(\theta)}
\end{aligned}
$$

Training an EBM

$$
\begin{aligned}
&\nabla_\theta \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \log p_\theta(x) \\
=& -\nabla_\theta \mathop{\mathbb{E}}_{x\sim\mathcal{D}} E(x) - \nabla_\theta \log Z(\theta) \\
=& -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)} \\
=& -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\int_X \nabla_\theta \exp(-E_\theta(x))dx}{Z(\theta)} \\
=& -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\int_X -\exp(-E_\theta(x))\nabla_\theta E_\theta(x))dx}{Z(\theta)}
\end{aligned}
$$

Training an EBM

$$
\begin{aligned}
&\nabla_\theta \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \log p_\theta(x) \\
&= -\nabla_\theta \mathop{\mathbb{E}}_{x\sim\mathcal{D}} E(x) - \nabla_\theta \log Z(\theta) \\
&= -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)} \\
&= -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\int_X \nabla_\theta \exp(-E_\theta(x))dx}{Z(\theta)} \\
&= -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\int_X -\exp(-E_\theta(x))\nabla_\theta E_\theta(x))dx}{Z(\theta)} \\
&= -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) + \int_X \frac{\exp(-E_\theta(x))}{Z(\theta)} \nabla_\theta E_\theta(x)dx
\end{aligned}
$$

Training an EBM

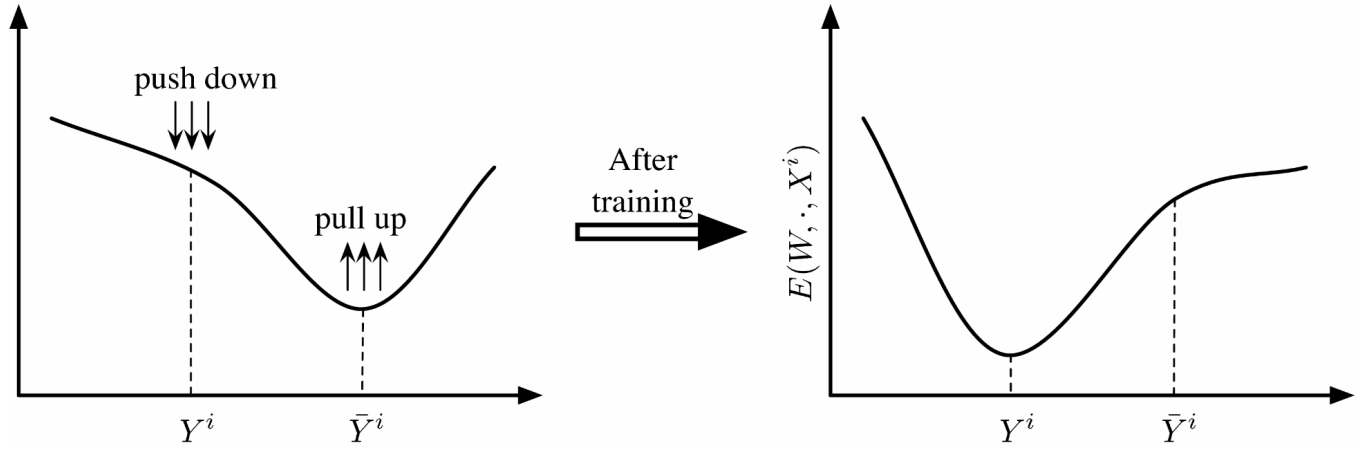
$$
\begin{aligned}
&\nabla_\theta \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \log p_\theta(x) \\
&= -\nabla_\theta \mathop{\mathbb{E}}_{x\sim\mathcal{D}} E(x) - \nabla_\theta \log Z(\theta) \\
&= -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\nabla_\theta Z(\theta)}{Z(\theta)} \\
&= -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\int_X \nabla_\theta \exp(-E_\theta(x))dx}{Z(\theta)} \\
&= -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) - \frac{\int_X -\exp(-E_\theta(x))\nabla_\theta E_\theta(x))dx}{Z(\theta)} \\
&= -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) + \int_X \frac{\exp(-E_\theta(x))}{Z(\theta)} \nabla_\theta E_\theta(x)dx \\
&= -\mathop{\mathbb{E}}_{x\sim\mathcal{D}} \nabla_\theta E(x) + \mathop{\mathbb{E}}_{x\sim p_\theta(x)} \nabla_\theta E(x)
\end{aligned}
$$

Training an EBM

Contrastive Divergence (CD)

$$\mathcal{L}_{\mathrm{CD}} = \mathbb{E}_{p_{\mathrm{data}}(x)}[E_\theta(x)] - \mathbb{E}_{p_\theta(x)}[E_\theta(x)]$$

GAN-like objective



https://cs.nyu.edu/~yann/talks/lecun-20060816-ciar-1-ebm.pdf#page=9.00

Training an EBM

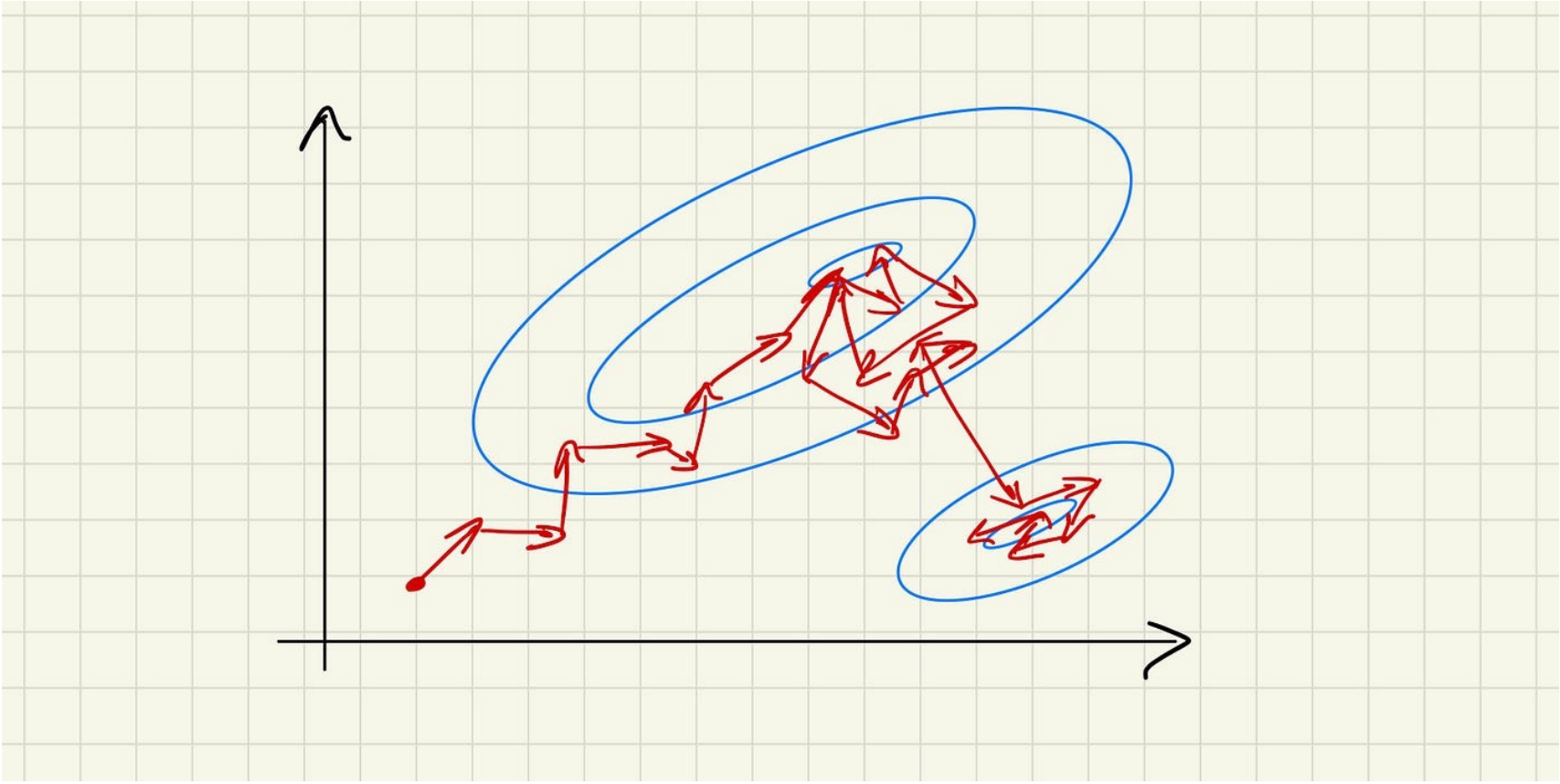
Contrastive Divergence (CD)

$$\mathcal{L}_{\mathrm{CD}} = \mathbb{E}_{p_{\mathrm{data}}(x)}[E_\theta(x)] - \mathbb{E}_{p_\theta(x)}[E_\theta(x)]$$

How to get samples from $p_\theta(x)$?

Monte Carlo Markov Chain (MCMC) sampling

MCMC



https://towardsdatascience.com/introduction-to-mcmc-1c8e3ea88cc9

Training an EBM

Suppose we get negative samples with k-step MCMC

CD-K

$$\mathcal{L}_{\mathrm{CD}-k} = \mathbb{E}_{p_{\mathrm{data}}(x)}[E_\theta(x^{(0)})] - \mathbb{E}_{p_\theta(x^{(k)})}[E_\theta(x^{(k)})]$$

Which MCMC sampler?

MCMC in continuous space

Langevin dynamics

$$x_{t+1} = x_t - \frac{\eta}{2}\nabla_x E(x_t) + \sqrt{\eta}\,\xi_t$$

where:

"Gradient descent with Gaussian noise"

- x_t: Current state of the Markov chain.
- η: Step size (learning rate).
- $\nabla_x E(x_t)$: Gradient of the energy function with respect to x.
- ξ_t: Gaussian noise sampled as $\xi_t \sim \mathcal{N}(0, I)$.

MCMC in continuous space

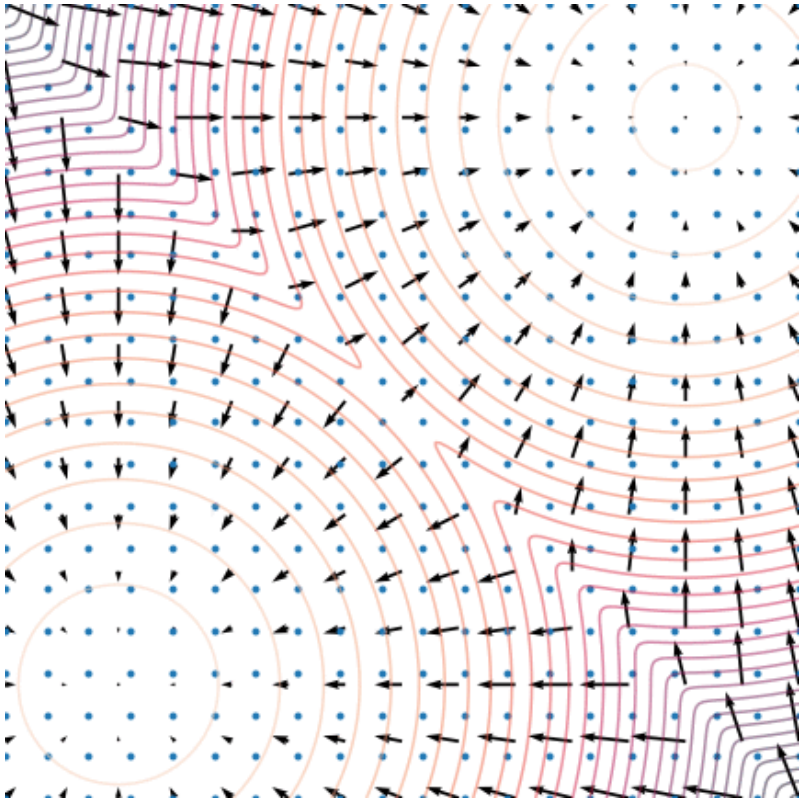
Algorithm: Langevin Monte Carlo (LMC)

Input: Energy function $E(x)$, step size $\eta > 0$, number of iterations T

Output: Samples x_T from the target distribution $p(x) \propto \exp(-E(x))$

1. Initialize x_0 (e.g., randomly or with prior knowledge).
2. **For** $t = 0$ to $T - 1$:
   a. Compute the gradient of the energy:
   $\nabla_x E(x_t)$.
   b. Update the state using the Langevin dynamics:
   $x_{t+1} = x_t - \frac{\eta}{2} \nabla_x E(x_t) + \sqrt{\eta}\, \xi_t$,
   where $\xi_t \sim \mathcal{N}(0, I)$ (Gaussian noise).
3. **End For**
4. Return x_T (final sample) or the sequence $\{x_t\}_{t=0}^{T}$ (full trajectory).

Demo



https://yang-song.net/blog/2021/score/

MCMC in continuous space

MCMC can take long time to converge

Why restart MCMC sampling every time?

CD-K ⇒ Persistent CD (PCD)

Basic idea:

- Maintain a replay buffer that stores previously sampled negative samples
- For each CD iteration:
  randomly choose to restart randomly, or pick one from the buffer to start MCMC

Connection to GAN

It can be proved that

$$\log Z = \max_q \mathbb{E}_q[-E(x)] - H(q)$$

where:

- Z is the normalization constant, defined as $Z = \int \exp(-E(x))dx$.
- $E(x)$ is the energy function.
- $H(q)$ is the entropy of $q(x)$, given by:

$$H(q) = -\int q(x) \log q(x)\, dx.$$

- The maximization is over all valid probability distributions $q(x)$, i.e., $q(x) \geq 0$ and $\int q(x)dx = 1$.

Connection to GAN

$$\log Z = \max_q \mathbb{E}_q[-E(x)] - H(q)$$

Recall the KL objective:

$$\max_\theta \mathbb{E}_{x\sim D} \log p_\theta(x) = \max_\theta \mathbb{E}_{x\sim D}\big[-E_\theta(x)\big] - \log Z_\theta$$

Therefore,

$$\max_\theta \mathbb{E}_{x\sim D} \log p_\theta(x) = \max_\theta \min_q \left(H(q) + \mathbb{E}_q\big[E_\theta(x)\big] - \mathbb{E}_{x\sim D}\big[E_\theta(x)\big] \right)$$

Connection to GAN

$$\max_{\theta} \min_{q} \left(H(q) + \mathbb{E}_q \left[E_\theta(x) \right] - \mathbb{E}_{x \sim D} \left[E_\theta(x) \right] \right)$$

What if we use a generator for q?

Ignoring the entropy term ⇒ W-GAN objective (with Wasserstein distance)

$$\mathbb{E}_{x \sim \mathbb{P}_r} [f_w(x)] - \mathbb{E}_{z \sim p(z)} [f_w(g_\theta(z)]$$

Noise Contrastive Estimation

Learns an EBM by distinguishing data from a (known) noise distribution

$$\mathcal{L}_{\mathrm{NCE}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\left[\log\sigma\big(f_\theta(x_i) - \log p_{\mathrm{noise}}(x_i)\big) + \sum_{j=1}^{k}\log\left(1 - \sigma\big(f_\theta(x'_j) - \log p_{\mathrm{noise}}(x'_j)\big)\right)\right]$$

Analogy: training a GAN, but

- with noise as the “generator”

- only train the “discriminator” (the energy function)

Modern EBMs

https://arxiv.org/pdf/2012.01316

Score Matching

Score Matching

Suppose you already have access to the target distribution..

Score matching

$$\mathcal{L}(\theta) = \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\|\nabla_x \log p_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|^2\right]$$

$-\nabla_x E_\theta(x)$

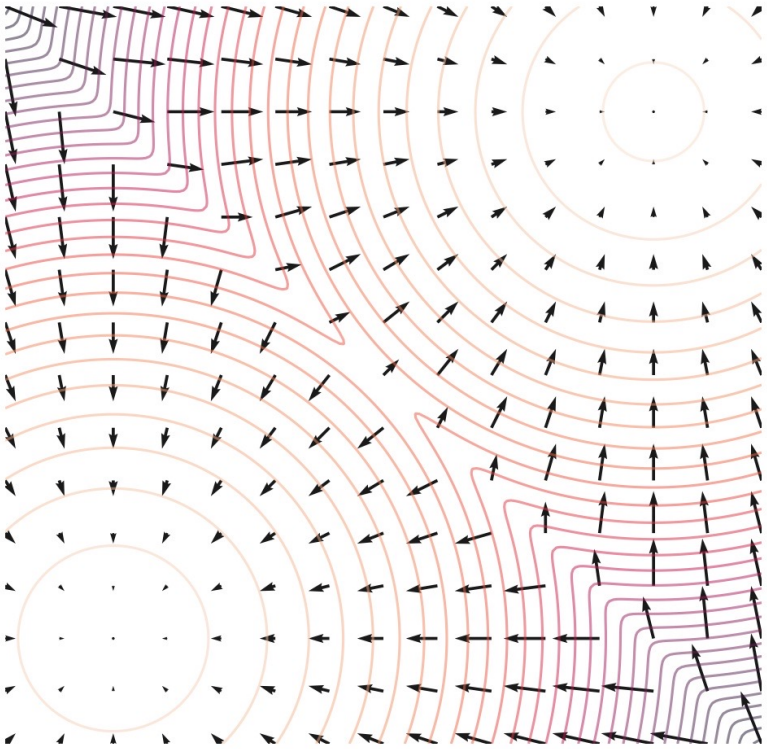
(Fisher) score function

Not the typical score function in statistics
(which takes derivative w.r.t. theta)

Fisher divergence between p_data and p_theta

Score Matching

Score matching = matching the gradient field of energy



Score Matching

But we do not have data score

Score matching

$$\mathcal{L}(\theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[\text{Tr}(\nabla_x^2 \log p_\theta(x)) + \frac{1}{2} \| \nabla_x \log p_\theta(x) \|^2 \right]$$

Computing the trace of a large Hessian is often infeasible

The trace operator can be estimated by

$$\text{Tr}(A) = \mathbb{E}_{v \sim \mathcal{N}(0,I)} \left[v^\top A v \right]$$

Proof

$$\mathcal{L}_{\text{SM}}(\theta) = \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}\left\|\nabla_x \log p_\theta(x) - \nabla_x \log p_{\text{data}}(x)\right\|^2$$

$$= \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\|\nabla_x \log p_\theta(x)\|^2 - 2\nabla_x \log p_\theta(x)^\top \nabla_x \log p_{\text{data}}(x) + \|\nabla_x \log p_{\text{data}}(x)\|^2\right]$$

Notice that the last term is constant w.r.t. theta

Instead, write the equivalent one

$$\mathcal{L}_{\text{SM}}(\theta) = \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\|\nabla_x \log p_\theta(x)\|^2 - 2\nabla_x \log p_\theta(x)^\top \nabla_x \log p_{\text{data}}(x)\right]$$

Proof

$$\mathcal{L}_{\text{SM}}(\theta) = \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\|\nabla_x \log p_\theta(x)\|^2 - 2\nabla_x \log p_\theta(x)^\top \nabla_x \log p_{\text{data}}(x)\right]$$

Since $\nabla_x \log p_{\text{data}}(x) = \dfrac{\nabla_x p_{\text{data}}(x)}{p_{\text{data}}(x)}$,

$$\mathcal{L}_{\text{SM}}(\theta) = \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\|\nabla_x \log p_\theta(x)\|^2\right] - \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\frac{\nabla_x \log p_\theta(x)^\top \nabla_x p_{\text{data}}(x)}{p_{\text{data}}(x)}\right]$$

Let's see if the second term matches $\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\text{Tr}(\nabla_x^2 \log p_\theta(x))\right]$

Proof

$$\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\frac{\nabla_x \log p_\theta(x)^\top \nabla_x p_{\text{data}}(x)}{p_{\text{data}}(x)}\right] = \int \nabla_x \log p_\theta(x)^\top \nabla_x p_{\text{data}}(x)\, dx$$

Notice that $\int \nabla_x \cdot (v(x)p_{\text{data}}(x))dx = 0$ if $v(x)p_{\text{data}}(x)$ vanishes as $x \to \infty$

(Proof by divergence theorem)

By the property of divergence,

$$\nabla_x \cdot (\nabla_x \log p_\theta(x) p_{\text{data}}(x)) = (\nabla_x^2 \log p_\theta(x)) p_{\text{data}}(x) + \nabla_x \log p_\theta(x)^\top \nabla_x p_{\text{data}}(x)$$

Therefore,

$$\int \left[(\nabla_x^2 \log p_\theta(x)) p_{\text{data}}(x) + \nabla_x \log p_\theta(x)^\top \nabla_x p_{\text{data}}(x)\right] dx = 0$$

Proof

$$\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\frac{\nabla_x \log p_\theta(x)^\top \nabla_x p_{\text{data}}(x)}{p_{\text{data}}(x)}\right] = \int \nabla_x \log p_\theta(x)^\top \nabla_x p_{\text{data}}(x)\, dx$$

With the following,

$$\int \left[(\nabla_x^2 \log p_\theta(x)) p_{\text{data}}(x) + \nabla_x \log p_\theta(x)^\top \nabla_x p_{\text{data}}(x)\right] dx = 0$$

We have

$$\begin{aligned}\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\frac{\nabla_x \log p_\theta(x)^\top \nabla_x p_{\text{data}}(x)}{p_{\text{data}}(x)}\right] &= -\int (\nabla_x^2 \log p_\theta(x)) p_{\text{data}}(x)\, dx \\ &= -\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\text{Tr}(\nabla_x^2 \log p_\theta(x))\right]\end{aligned}$$

Denoising Score Matching (DSM)

Estimating the trace of the Hessian is still tedious, even with approximations

Denoising Score Matching

$$\mathcal{L}_{\text{DSM}}(\theta) = \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x),\epsilon\sim\mathcal{N}(0,\sigma^2 I)}\left[\|\nabla_{\tilde{x}}\log p_\theta(\tilde{x}) + \frac{\tilde{x}-x}{\sigma^2}\|^2\right]$$

With $\tilde{x} = x + \epsilon$ the input but corrupted Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$

Proof

We may rewrite the DSM loss

$$L_{\text{DSM}}(\theta) = \mathbb{E}_{p_{\text{data}}(x_0)p_\sigma(x|x_0)}\left[\frac{1}{2}\|s_\theta(x) + \frac{x - x_0}{\sigma^2}\|^2\right]$$

into

$$L_{\text{DSM}}(\theta) = \mathbb{E}_{p_{\text{data}}(x_0)p_\sigma(x|x_0)}\left[\frac{1}{2}\|s_\theta(x)\|^2 + \frac{1}{2}\left\|\frac{x - x_0}{\sigma^2}\right\|^2 + \langle s_\theta(x), \frac{x - x_0}{\sigma^2}\rangle\right]$$

Reorganizing terms

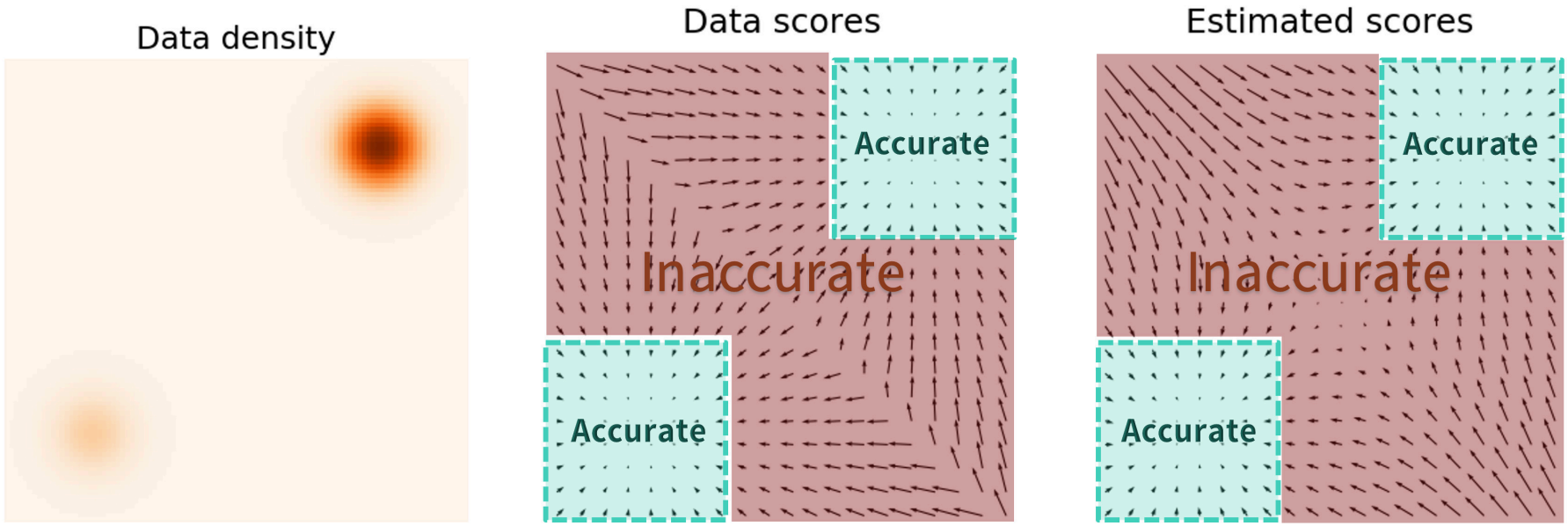
$$L_{\text{DSM}}(\theta) = \mathbb{E}_{p_{\text{data}}(x)}\left[\frac{1}{2}\|s_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|^2\right] + \text{constant}$$

Optimizing this loss is equivalent to optimizing

$$L_{\text{SM}}(\theta) = \mathbb{E}_{p_{\text{data}}(x)}\left[\frac{1}{2}\|s_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|^2\right]$$

Issue with DSM

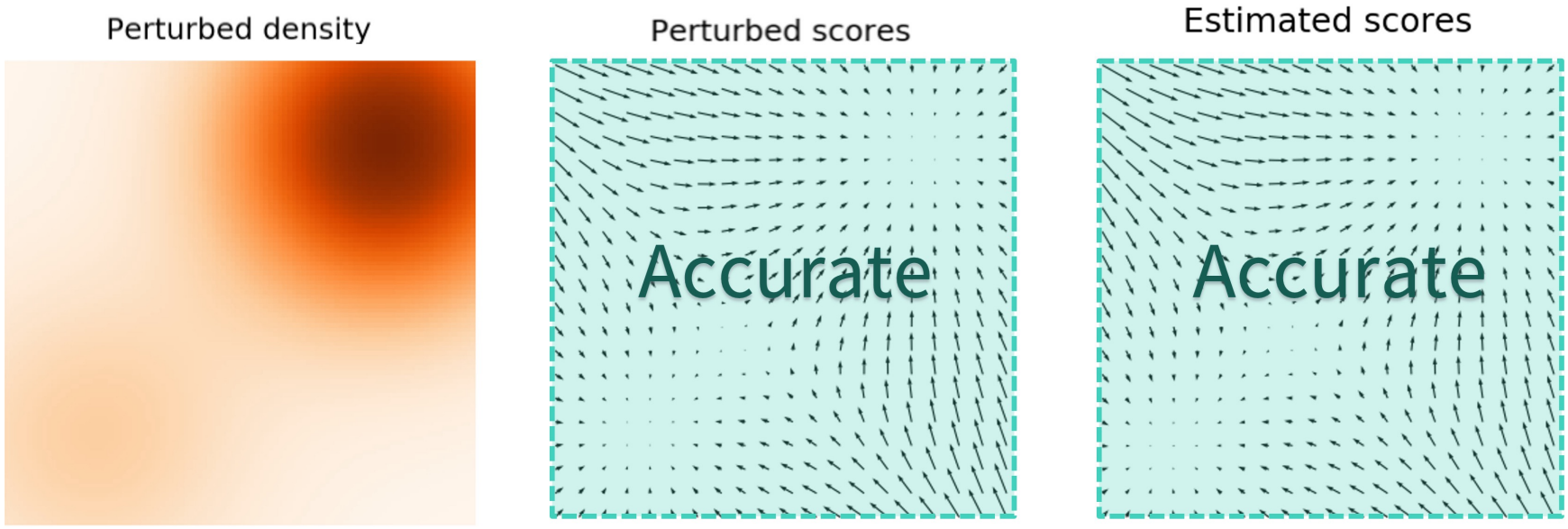
Places less likely to be sampled are less trained



https://yang-song.net/blog/2021/score/

Issue with DSM

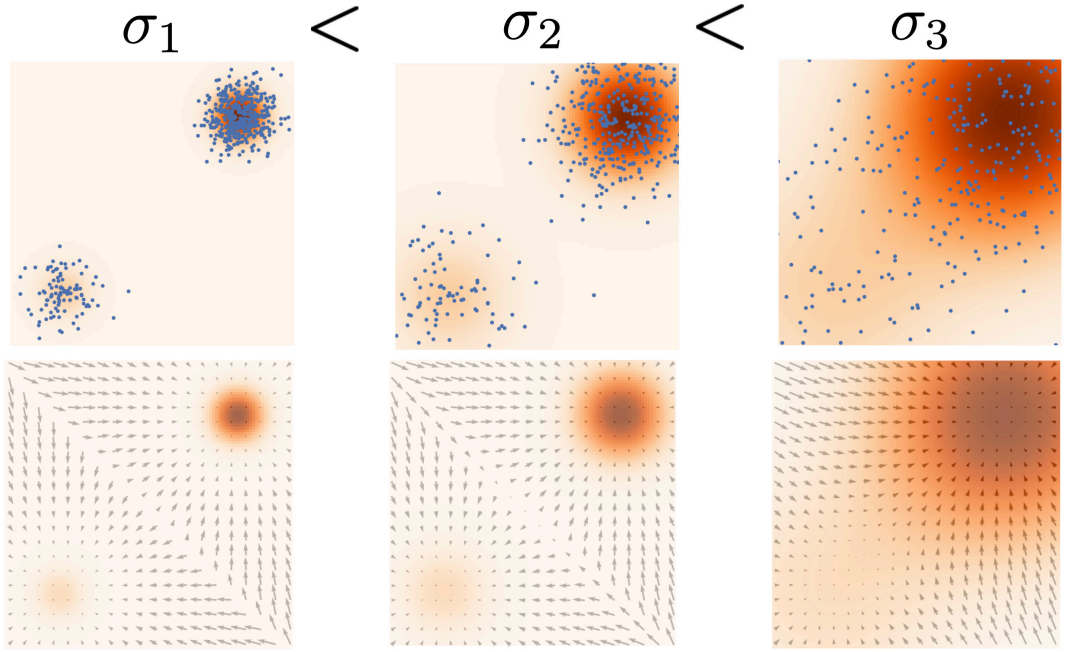
But we can increase the injected noise



https://yang-song.net/blog/2021/score/

Noise Conditional Score-Based Model

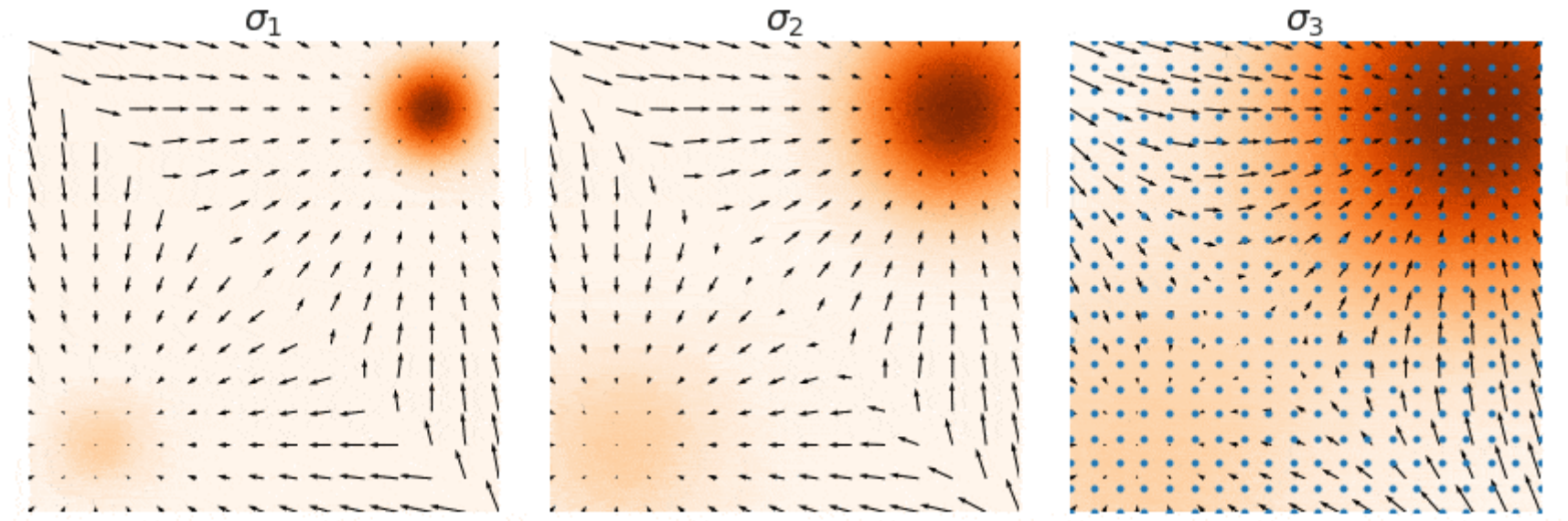
Strategy: cover the space with multiple noise scales



https://yang-song.net/blog/2021/score/

Noise Conditional Score-Based Model

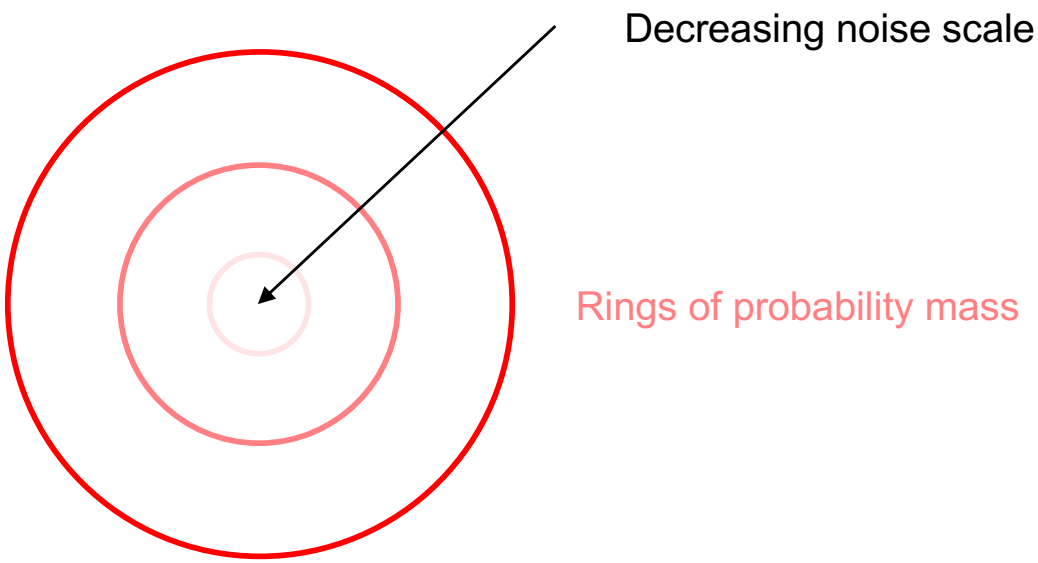
Inference with EBMs of decreasing noise scales



https://yang-song.net/blog/2021/score/

Noise Conditional Score-Based Model

In high-dimensional spaces, Gaussians have the most mass in a thin shell



Noise Conditional Score-Based Model

Objective: multi-scale denoising score matching

$$\sum_{i=1}^{L} \lambda(i) \mathbb{E}_{p_{\sigma_i}(\mathbf{x})} [\| \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, i) \|_2^2]$$

Positive weight scalar (design choice)

$$= -\frac{\tilde{x} - x}{\sigma^2}$$

What does $\mathbf{s}_\theta(\mathbf{x}, i)$ do? Denoise an input to a clean one (of the same dimension)

i.e., denoising autoencoder

Noise Conditional Score-Based Model

Objective: multi-scale denoising score matching

$$\sum_{i=1}^{L} \lambda(i) \mathbb{E}_{p_{\sigma_i}(\mathbf{x})} [\| \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, i) \|_2^2]$$

Positive weight scalar (design choice)

$$= -\frac{\tilde{x} - x}{\sigma^2}$$

What does $\mathbf{s}_\theta(\mathbf{x}, i)$ do? Denoise an input to a clean one (of the same dimension)

i.e., denoising autoencoder

Inference of NCSN

Langevin MC with annealed noises

Algorithm 1 Annealed Langevin dynamics.

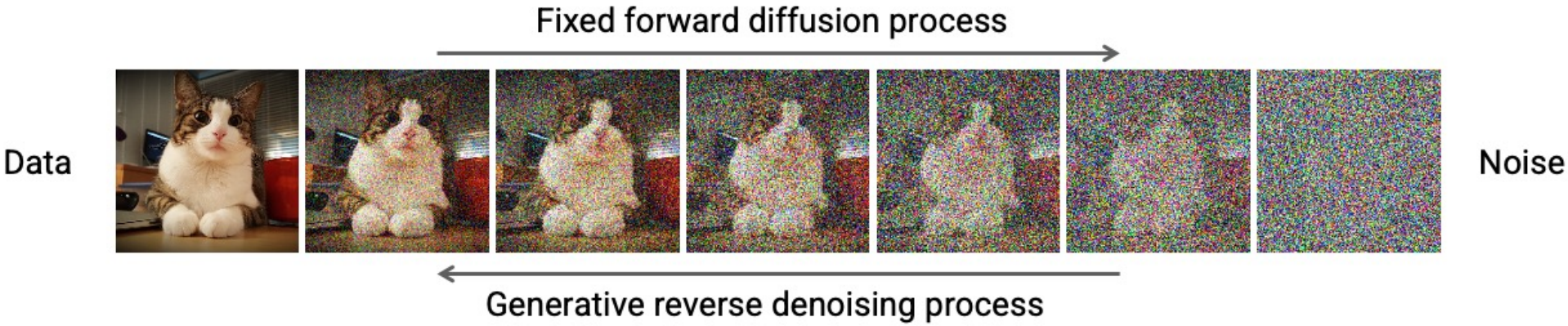
Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.

Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to L **do**
  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ ▷ α_i is the step size.
  for $t \leftarrow 1$ to T **do**
    Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\, \mathbf{z}_t$
  end for
  $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$

Diffusion Models

What if we only use one single Langevin step for each noise scale?

The typical diffusion model (DDPM, etc.) you see these days.



Why the EBM explanation?
More structured analysis, especially when you make the noise schedule continuous

Next Time

Diffusion Models (DDPM, etc.) with a more classical introduction

Move to Continuous-time: stochastic differential equation (SDE) formulation

Have a nice winter break!