

Energy-Based Models

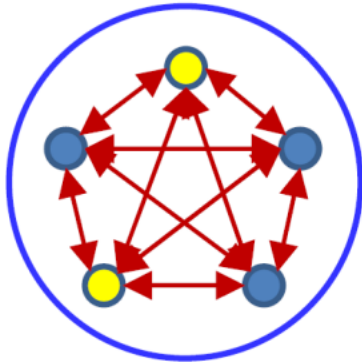
W

Energy-based Models

- Goal of generative models:
 - a probability distribution of data: $P(x)$
- Requirements
 - $P(x) \geq 0$ (non-negative)
 - $\int_x P(x)dx = 1$
- Energy-based model:
 - Energy function: $E(x; \theta)$, parameterized by θ
 - $P(x) = \frac{1}{z} \exp(-E(x; \theta))$ (why exp?)
 - $z = \int_x \exp(-E(x; \theta))dx$

Boltzmann Machine

- Generative model
 - $E(y) = -\frac{1}{2}y^\top W y$
 - $P(y) = \frac{1}{z} \exp(-\frac{E(y)}{T})$, T : temperature hyper-parameter
 - W : parameter to learn
- When y_i is binary, patterns are affecting each other through W



$$z_i = \frac{1}{T} \sum_j w_{ji} s_j$$

$$P(s_i = 1 | s_{j \neq i}) = \frac{1}{1 + e^{-z_i}}$$

Boltzmann Machine: Training

- Objective: maximum likelihood learning (assume $T = 1$):
 - Probability of one sample:

$$P(y) = \frac{\exp(\frac{1}{2}y^T y)}{\sum_{y'} \exp(y'^T W y')}$$

- Maximum log-likelihood:

$$L(W) = \frac{1}{N} \sum_{y \in D} \frac{1}{2} y^T W y - \log \sum_{y'} \exp(\frac{1}{2} y'^T W y')$$

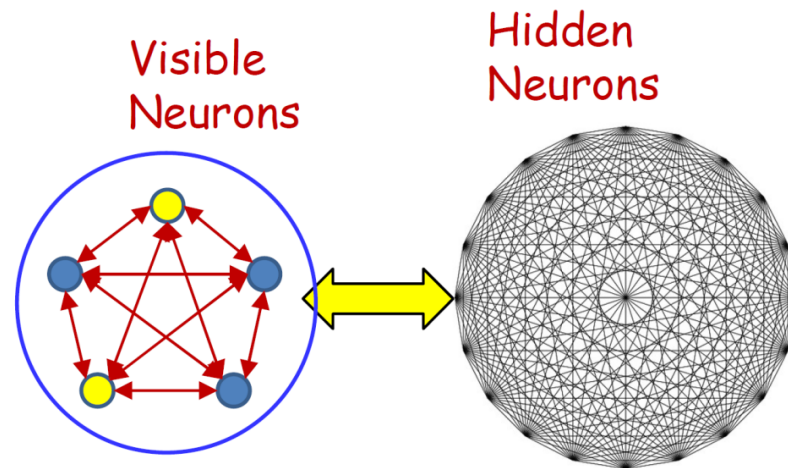
Boltzmann Machine: Training

Boltzmann Machine: Training

Boltzmann Machine with Hidden Neurons

- Visible and hidden neurons:
 - y : visible, h : hidden

- $$P(y) = \sum_h P(y, v)$$

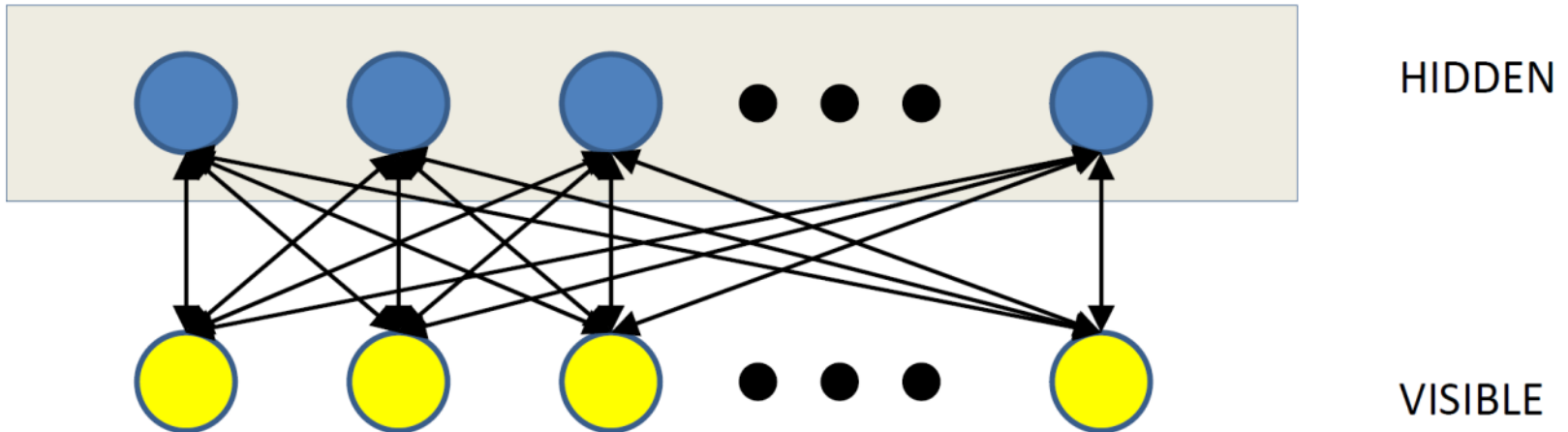


Boltzmann Machine with Hidden Neurons: Training

Boltzmann Machine with Hidden Neurons: Training

Restricted Boltzmann Machine

- A structured Boltzmann Machine
 - Hidden neurons are only connected to visible neurons
 - No intra-layer connections
 - Invented by Paul Smolensky in '89
 - Became more practical after Hinton invested fast learning algorithms in mid 2000



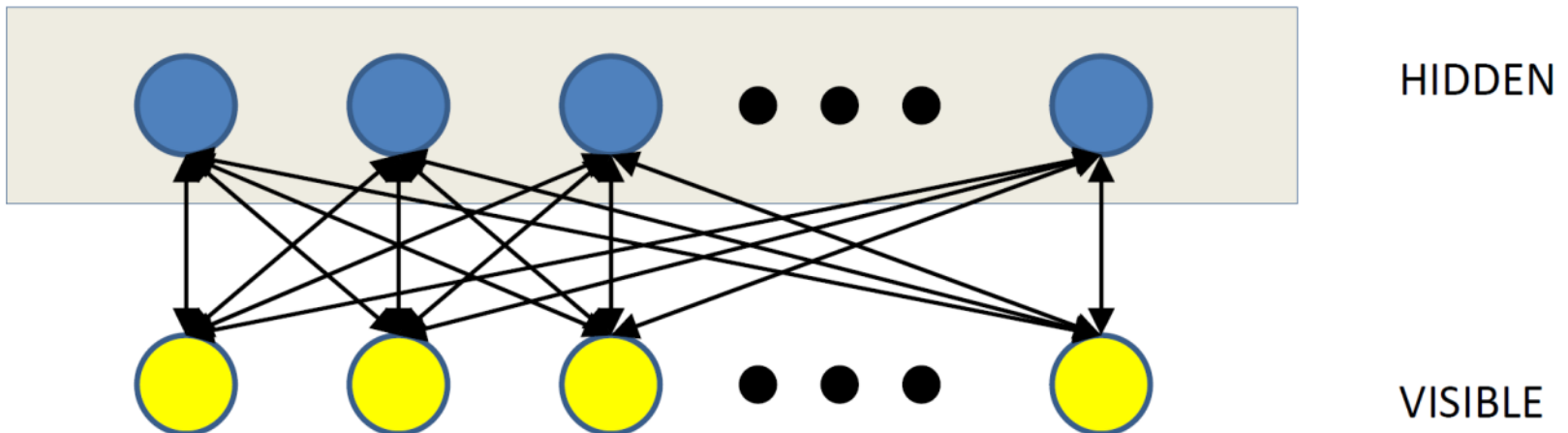
Restricted Boltzmann Machine

- Computation Rules

- Iterative sampling

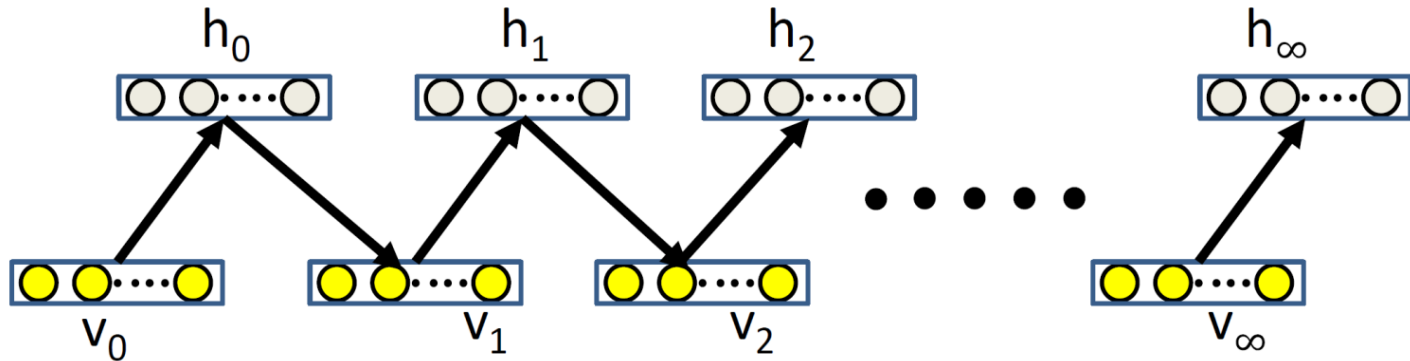
- Hidden neurons h_i : $z_i = \sum_j w_{ij}v_j$, $P(h_i | v) = \frac{1}{1 + \exp(-z_i)}$

- Visible neurons v_j : $z_j = \sum_i w_{ij}h_i$, $P(v_j | h) = \frac{1}{1 + \exp(-z_j)}$



Restricted Boltzmann Machine

- Sampling:
 - Randomly initialize visible neurons v_0
 - Iterative sampling between hidden neurons and visible neurons
 - Get final sample (v_∞, h_∞)

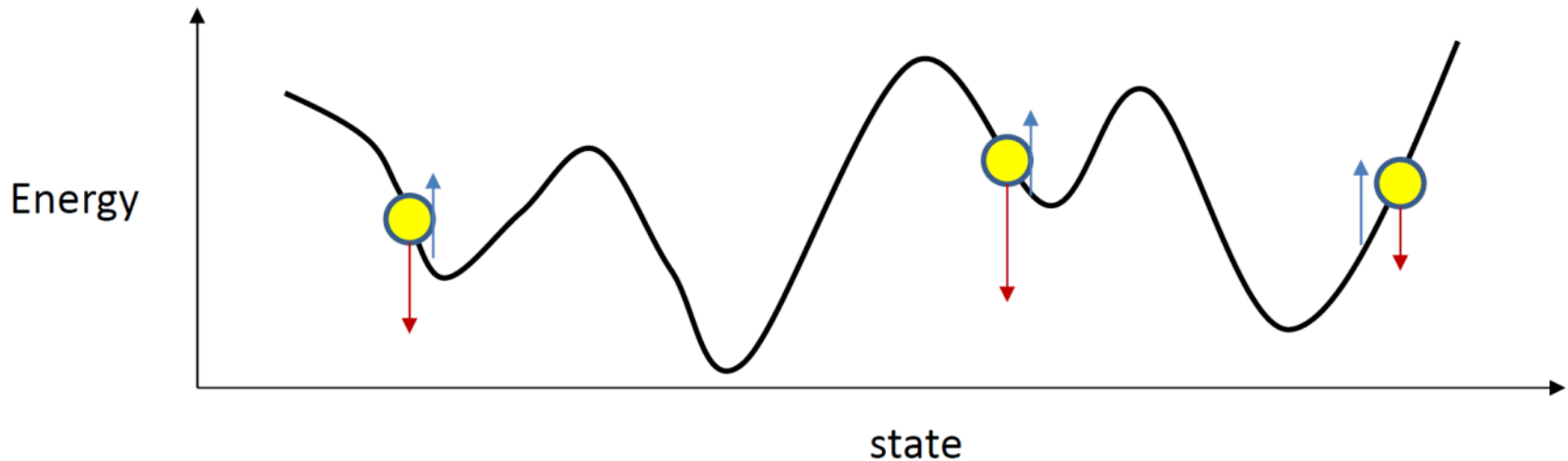


Restricted Boltzmann Machine

- Maximum likelihood estimated:

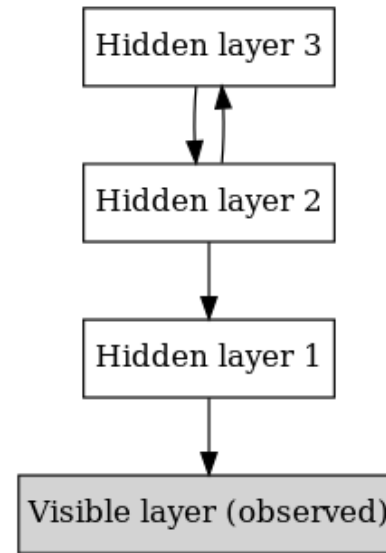
- $$\nabla_{w_{ij}} L(W) = \frac{1}{N_p K} \sum_{v \in P} v_{0i} h_{0j} - \frac{1}{M} \sum v_{\infty i} h_{\infty j}$$

- No need to lift up the entire energy landscape!
 - Raising the neighborhood of desired patterns is sufficient

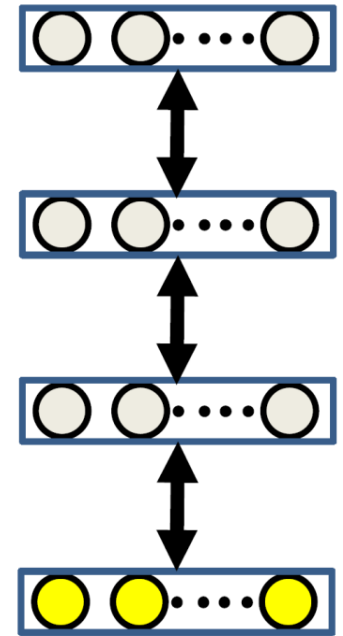


Deep Boltzmann Machine

- Can we have a **deep** version of RBM?
 - Deep Belief Net ('06)
 - Deep Boltzmann Machine ('09)
- Sampling?
 - Forward pass: bottom-up
 - Backward pass: top-down
- Deep Boltzmann Machine
 - The very first deep generative model
 - Salakhudinov & Hinton



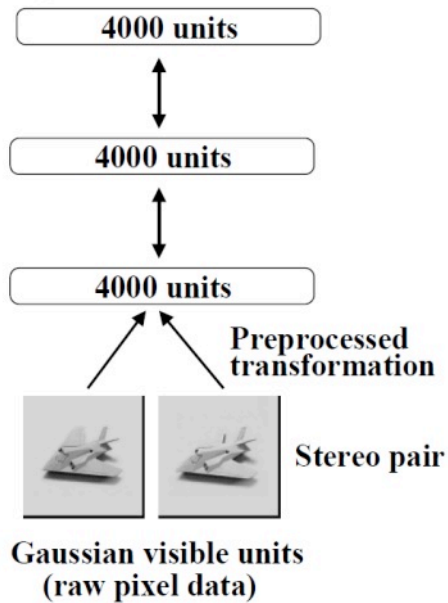
deep belief net



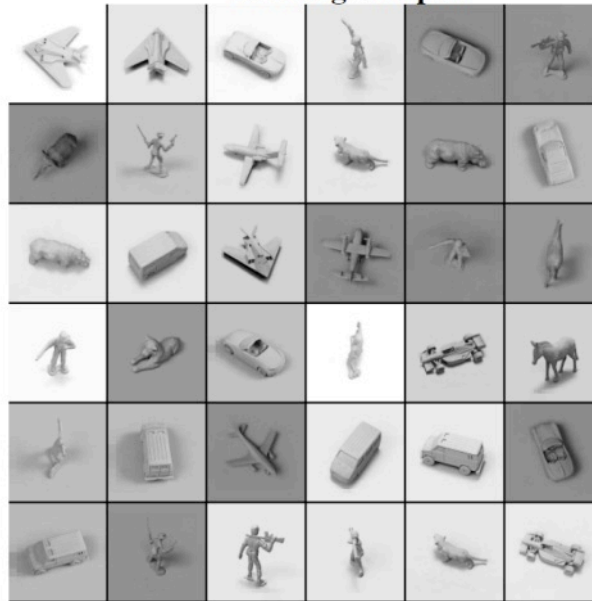
Deep Boltzmann Machine

Deep Boltzmann Machine

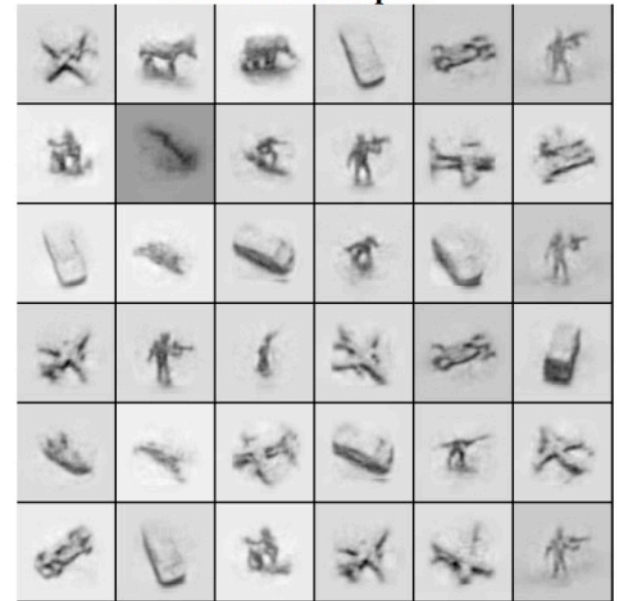
Deep Boltzmann Machine



Training Samples



Generated Samples



Summary

- Pros: powerful and flexible

- An arbitrarily complex density function $p(x) = \frac{1}{z} \exp(-E(x))$

- Cons: hard to sample / train

- Hard to sample:
 - MCMC sampling
 - Partition function
 - No closed-form calculation for likelihood
 - Cannot optimize MLE loss exactly
 - MCMC sampling

Normalizing Flows

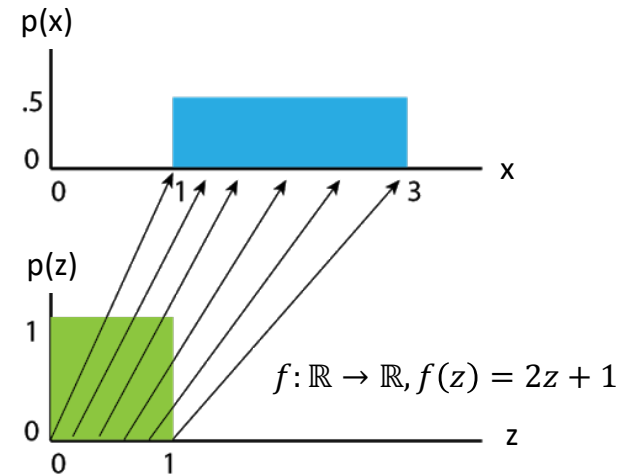


Intuition about easy to sample

- Goal: design $p(x)$ such that
 - Easy to sample
 - Tractable likelihood (density function)
- Easy to sample
 - Assume a continuous variable z
 - e.g., Gaussian $z \sim N(0,1)$, or uniform $z \sim \text{Unif}[0,1]$
 - $x = f(z)$, x is also easy to sample

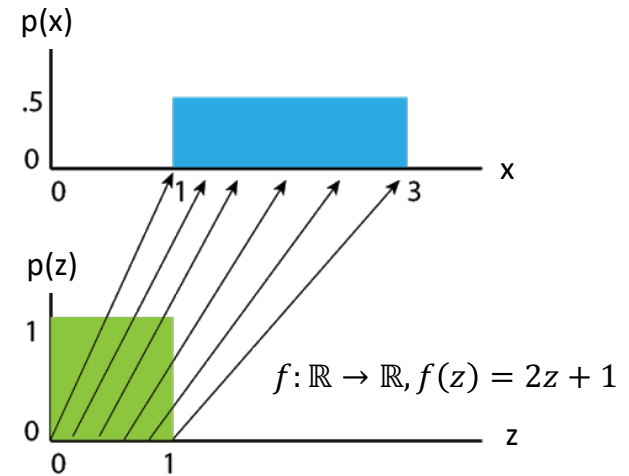
Intuition about tractable density

- Goal: design $f(z; \theta)$ such that
 - Assume z is from an “easy” distribution
 - $p(x) = p(f(z; \theta))$ has tractable likelihood
- Uniform: $z \sim \text{Unif}[0,1]$
 - Density $p(z) = 1$
 - $x = 2z + 1$, then $p(x) = ?$



Intuition about tractable density

- Goal: design $f(z; \theta)$ such that
 - Assume z is from an “easy” distribution
 - $p(x) = p(f(z; \theta))$ has tractable likelihood
- Uniform: $z \sim \text{Unif}[0,1]$
 - Density $p(z) = 1$
 - $x = 2z + 1$, then $p(x) = 1/2$
 - $x = az + b$, then $p(x) = 1/|a|$ (for $a \neq 0$)
 - $x = f(z)$, $p(x) \left| \frac{dz}{dx} \right| = |f'(z)|^{-1} p(z)$
 - Assume $f(z)$ is a bijection



Change of variable

- Suppose $x = f(z)$ for some general non-linear $f(\cdot)$
 - The linearized change in volume is determined by the Jacobian of $f(\cdot)$:

- $$\frac{\partial f(z)}{\partial z} = \begin{bmatrix} \frac{\partial f_1(z)}{\partial z_1} & \dots & \frac{\partial f_1(z)}{\partial z_d} \\ \dots & \dots & \dots \\ \frac{\partial f_d(z)}{\partial z_1} & \dots & \frac{\partial f_d(z)}{\partial z_d} \end{bmatrix}$$

- Given a bijection $f(z) : \mathbb{R}^d \rightarrow \mathbb{R}^d$
 - $z = f^{-1}(x)$

- $$p(x) = p(f^{-1}(x)) \left| \det \left(\frac{\partial f^{-1}(x)}{\partial x} \right) \right| = p(z) \left| \det \left(\frac{\partial f^{-1}(x)}{\partial x} \right) \right|$$

- Since $\frac{\partial f^{-1}}{\partial x} = \left(\frac{\partial f}{\partial z} \right)^{-1}$ (Jacobian of invertible function)

- $$p(x) = p(z) \left| \det \left(\frac{\partial f^{-1}(x)}{\partial x} \right) \right| = p(z) \left| \det \left(\frac{\partial f(z)}{\partial z} \right) \right|^{-1}$$

Normalizing Flow

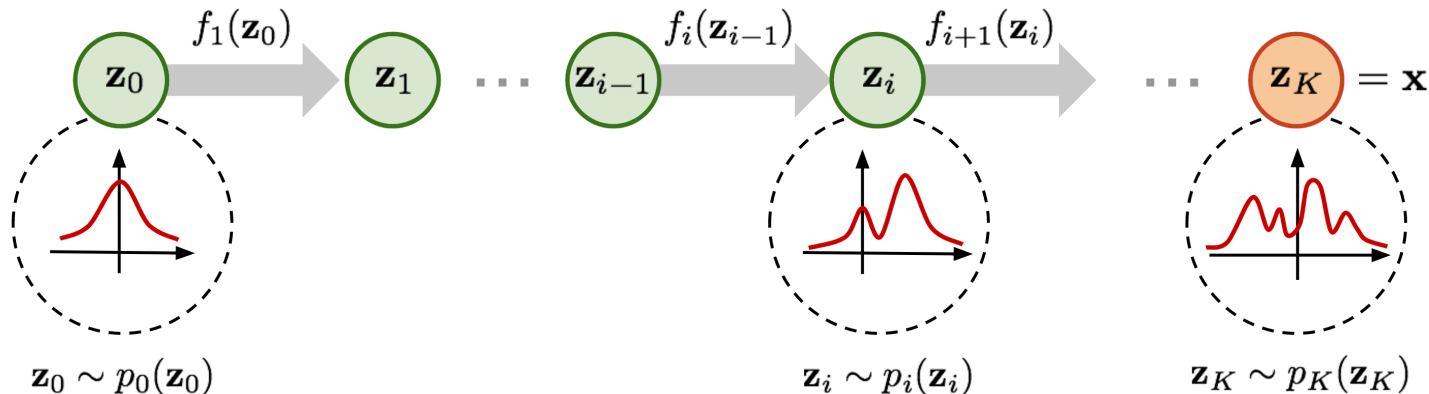
- Idea

- Sample z_0 from an “easy” distribution, e.g., standard Gaussian
- Apply K bijections $z_i = f_i(z_{i-1})$
- The final sample $x = f_K(z_K)$ has tractable density

- Normalizing Flow

- $z_0 \sim N(0, I)$, $z_i = f_i(z_{i-1})$, $x = z_K$ where $x, z_i \in \mathbb{R}^d$ and f_i is invertible
- Every reversible function produces a normalized density function

- $$p(z_i) = p(z_{i-1}) \left| \det \left(\frac{\partial f_i}{\partial z_{i-1}} \right) \right|^{-1}$$



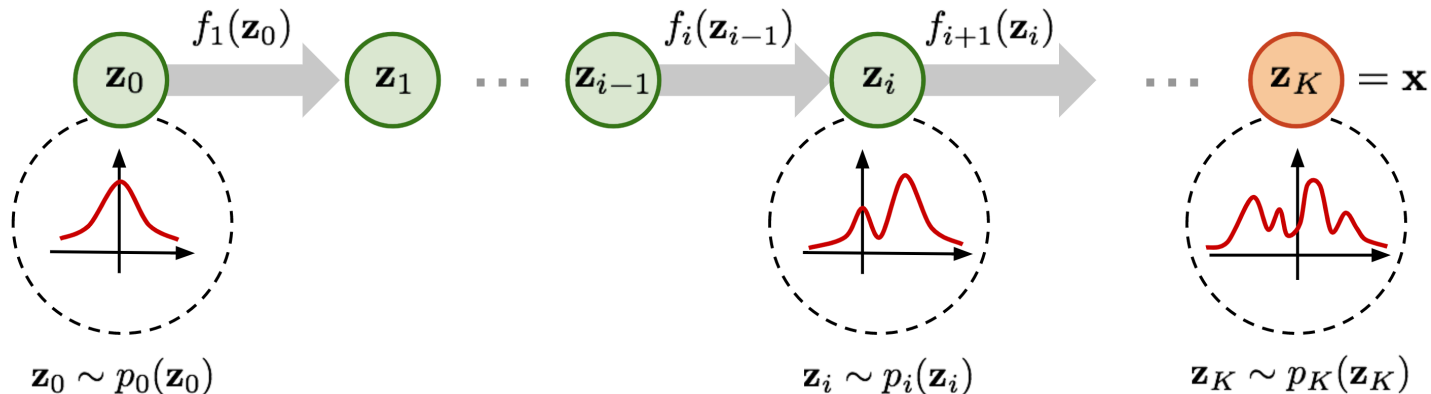
Normalizing Flow

- Generation is trivial
 - Sample z_0 then apply the transformations
- Log-likelihood

$$\bullet \log p(x) = \log p(z_{k-1}) - \log \left| \det \left(\frac{\partial f_K}{\partial z_{k-1}} \right) \right|$$

$$\bullet \log p(x) = \log p(z_0) - \sum_i \log \left| \det \left(\frac{\partial f_i}{\partial z_{i-1}} \right) \right|$$

$O(d^3)$!!!



Normalizing Flow

- Naive flow model requires extremely expensive computation
 - Computing determinant of $d \times d$ matrices
- Idea:
 - Design a good bijection $f_i(z)$ such that the determinant is easy to compute

Plannar Flow

- Technical tool: Matrix Determinant Lemma:

- $\det(A + uv^\top) = (1 + v^\top A^{-1}u) \det A$

- Model:

- $f_\theta(z) = z + u \odot h(w^\top z + b)$

- $h(\cdot)$ chosen to be $\tanh(\cdot)$ ($0 < h'(\cdot) < 1$)

- $\theta = [u, w, b], \det \left(\frac{\partial f}{\partial z} \right) = \det(I + h'(w^\top z + b)uw^\top) = 1 + h'(w^\top z + b)u^\top w$

- Computation in $O(d)$ time

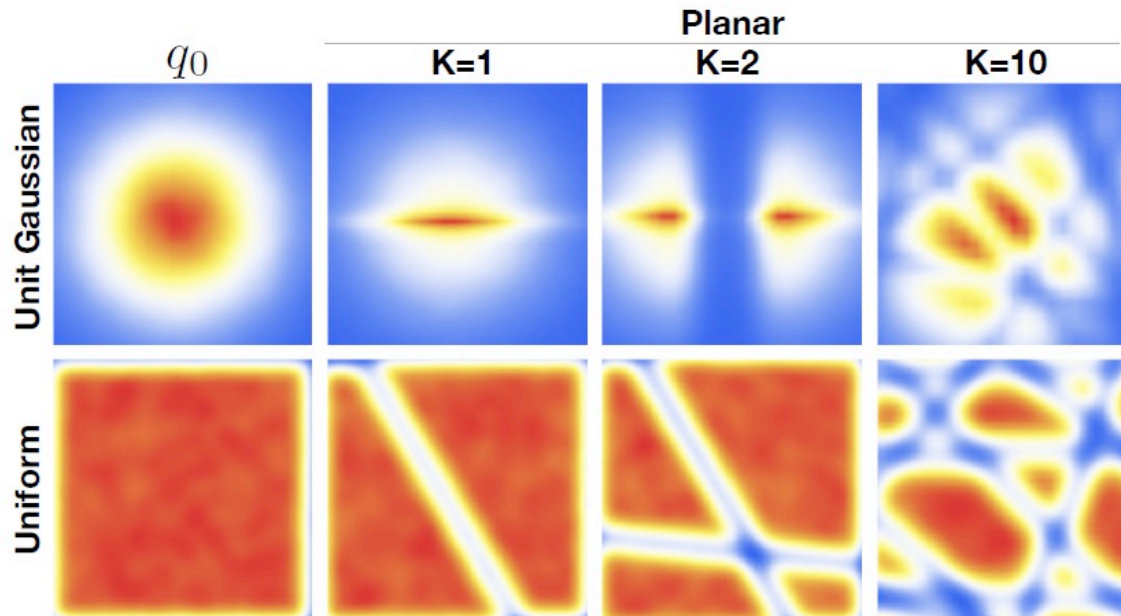
- Remarks:

- $u^\top w > -1$ to ensure invertibility

- Require normalization on u and w

Planar Flow (Rezende & Mohamed, '16)

- $f_{\theta}(z) = z + uh(w^{\top}z + b)$
- 10 planar transformations can transform simple distributions into a more complex one



Extensions

- Other flow models uses triangular Jacobian
 - Suppose $x_i = f_i(z)$ only depends on $z_{\leq i}$

Score-Based Models and Diffusion Models



Recap: Boltzmann Machine Training

- Objective: maximum likelihood learning (assume $T = 1$):
 - Probability of one sample:

$$P(y) = \frac{\exp(\frac{1}{2}y^T W y)}{\sum_{y'} \exp(\frac{1}{2}y'^T W y')}$$

- Maximum log-likelihood:

$$L(W) = \frac{1}{N} \sum_{y \in D} \frac{1}{2} y^T W y - \log \sum_{y'} \exp(\frac{1}{2} y'^T W y')$$

Can we avoid calculating the gradient of normalizing constant ($\nabla_x Z_\theta$)?

Score Matching

- Score Function
 - Definition:

$$\nabla_x \log p_{data}(x) : \mathbb{R}^d \rightarrow \mathbb{R}^d$$

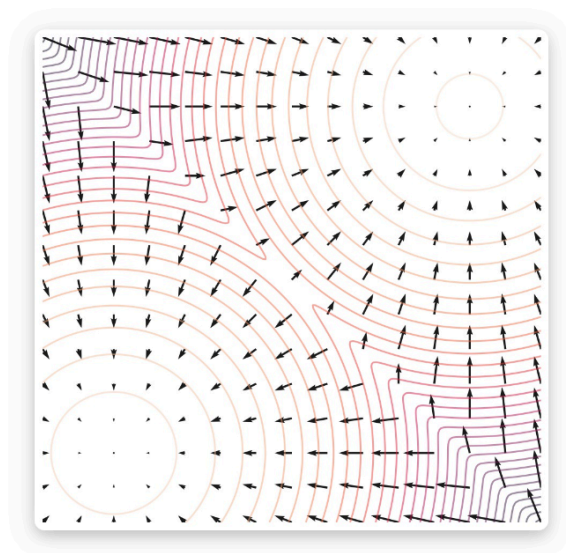
- Idea: directly fitting the score function:

- $\min_{\theta} \mathbb{E}_{p_{data}} \|\nabla_x \log p_{\theta}(x) - \nabla_x \log p_{data}(x)\|^2$

- No need to compute $\nabla_x Z_{\theta}$!

- Problem:

- How to compute $\nabla_x \log p_{data}(x)$?



Score function (the vector field) and density function (contours) of a mixture of two Gaussians.

Score Matching

Score Matching

Sliced Score Matching

$$L(\theta) = \frac{1}{N} \sum_{x \in D} \|s_{\theta}(x)\|^2 - 2 [Tr(Ds_{\theta}(x))]$$

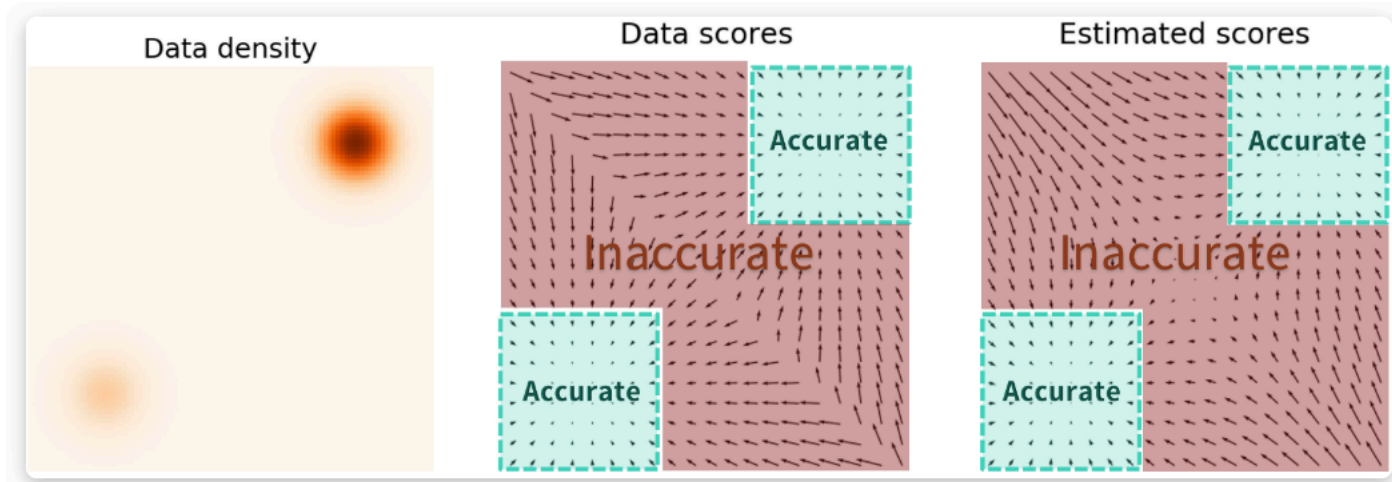
Score Matching: Langevin Dynamics

$$x_{t+1} \leftarrow x_t + \epsilon \nabla_x \log p(x) + \sqrt{2\epsilon} z_t, z_t \sim N(0, I)$$

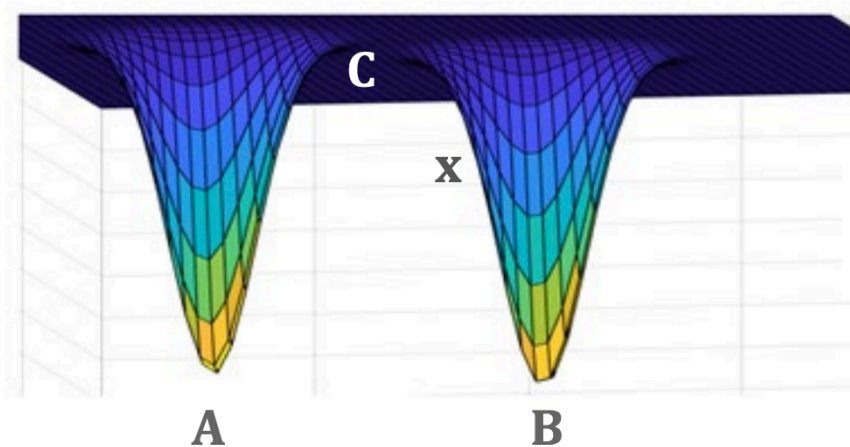
Stationary (equilibrium distribution): $p(x)$

Practical Issues

- Score function estimation is inaccurate in low density regions (few data available).



- Sampling is Slow



Annealing: Denoising Score Matching

- Fit several “smoothed” versions of p_{data} :
 - Choose temperatures: $\sigma_1, \sigma_2, \dots, \sigma_T$
 - $p_{\sigma_i, data}(x) = p_{data}(x) * N(0, \sigma_i) = \int_{\delta} p_{data}(x - \delta) N(x; \delta, \sigma_i) d\delta$
- Implementation:
 - Take a sample x , draw a sample $z \sim N(0, \sigma_i)$, output $x' = x + z$.

$$\sigma_1 > \sigma_2 > \dots > \sigma_{L-1} > \sigma_L$$

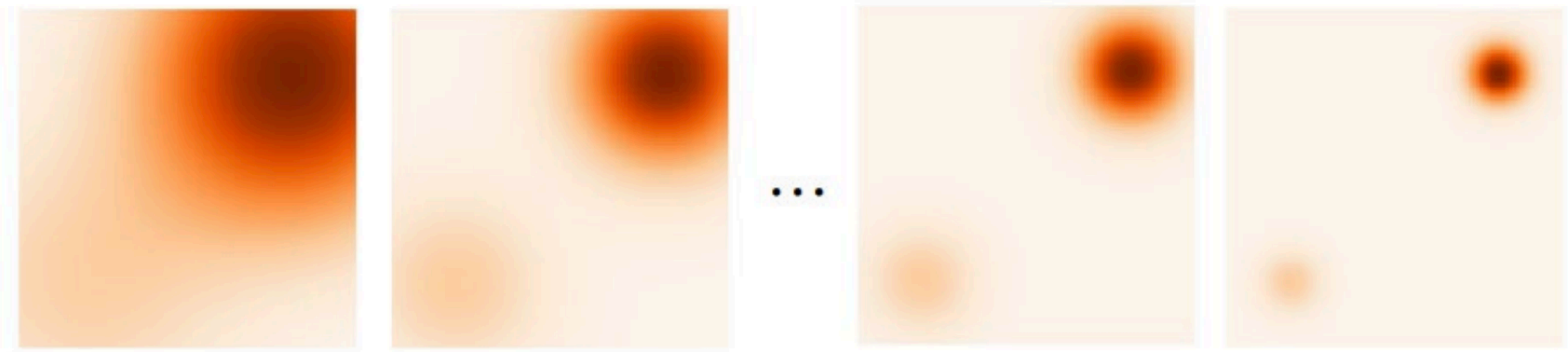


Figure by Stefano Ermon.

Annealing: Denoising Score Matching

$$\arg \min_{\theta} \sum_i \lambda(\sigma_i) \mathbb{E}_{x \sim p_{\sigma_i, data}} \|s_{\theta}(x, i) - \nabla_x \log p_{\sigma_i, data}(x)\|^2$$

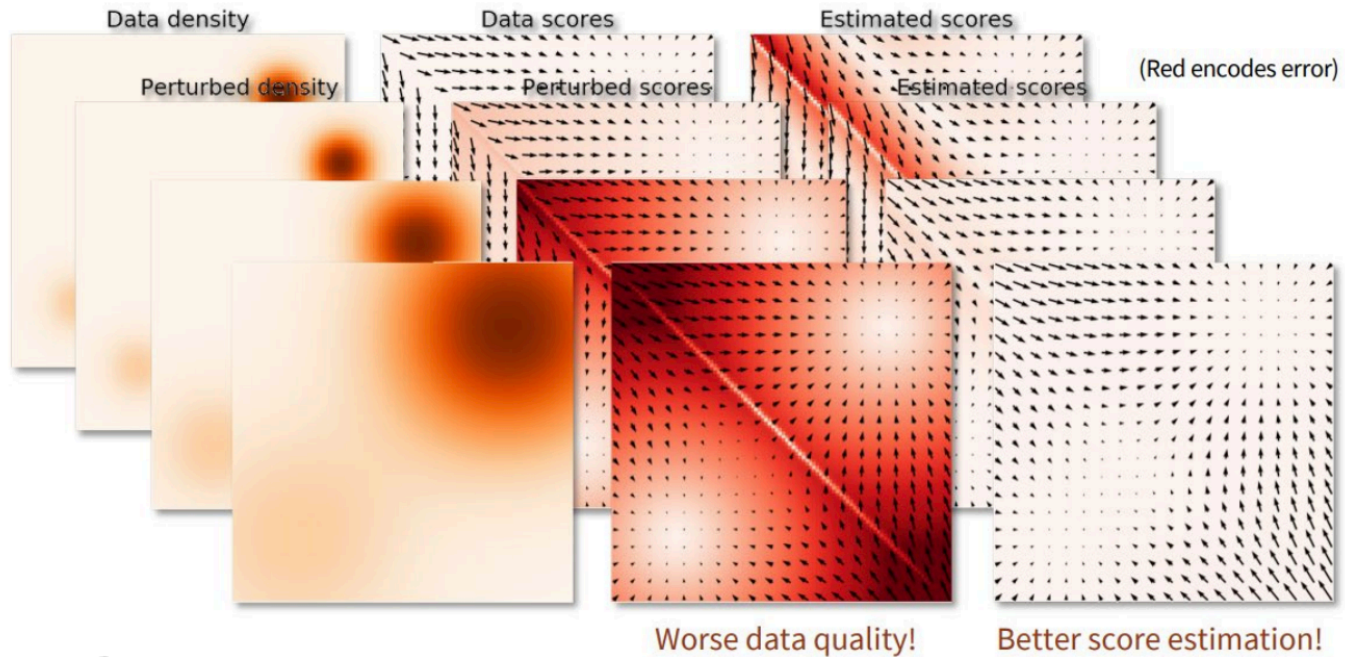


Figure by Stefano Ermon.

Annealed Langevin Dynamics

Algorithm 1 Annealed Langevin dynamics.

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

1: Initialize $\tilde{\mathbf{x}}_0$

2: **for** $i \leftarrow 1$ to L **do**

3: $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ $\triangleright \alpha_i$ is the step size.

4: **for** $t \leftarrow 1$ to T **do**

5: Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$

6: $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i} \mathbf{z}_t$

7: **end for**

8: $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$

9: **end for**

return $\tilde{\mathbf{x}}_T$

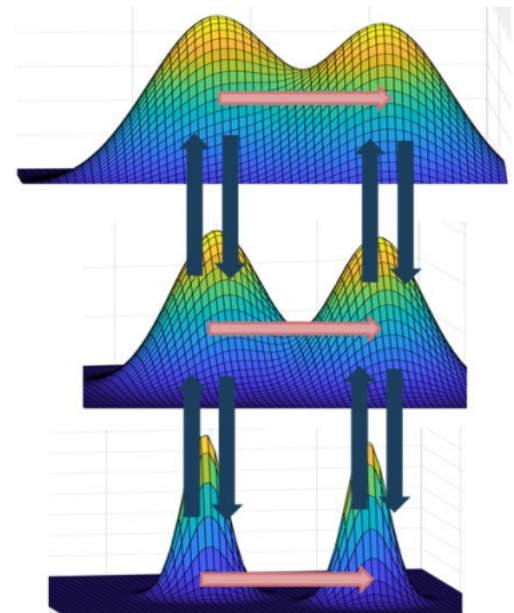


Figure from Song-Ermon '19

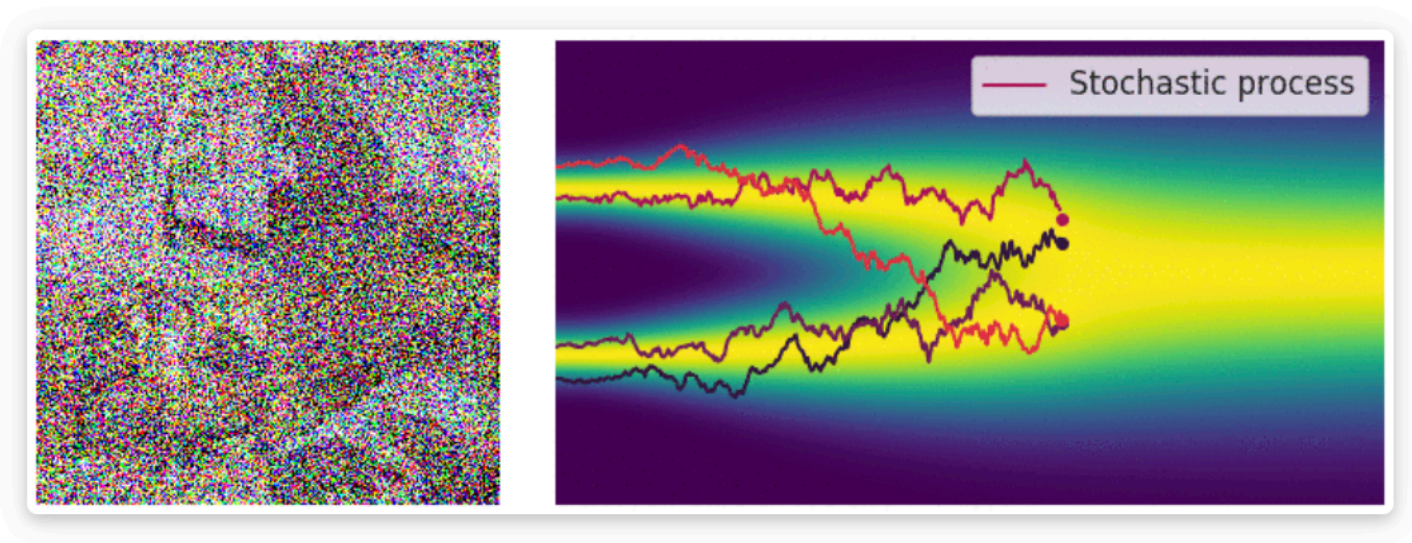
Diffusion Models



An image generated by Stable Diffusion based on the text prompt "a photograph of an astronaut riding a horse"

Perturbing Data with an SDE

- Let the number of noise scales approaches infinity!

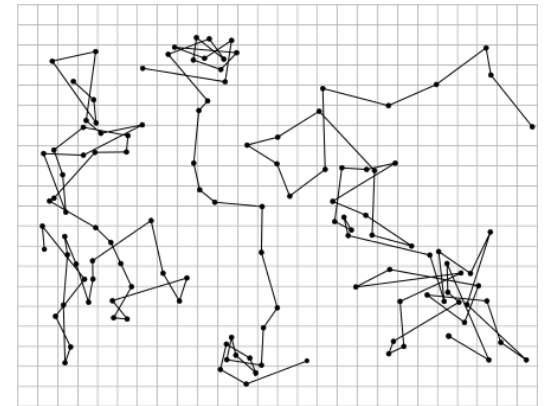


Perturbing data to noise with a continuous-time stochastic process.

Stochastic Differential Equations

$$dx = f(x, t)dt + g(t)dw$$

- $x(0)$: real image, $x(T)$: Gaussian noise.
- $f(x,t)$: drift terms. $g(t)$: diffusion coefficient.
- dw : Brownian motion
 - $w(t + u) - w(t) \sim N(0, u)$
- $f(x,t)$ and $g(t)$ are parts of the model.



- Variance Exploding SDE: $dx = \sqrt{\frac{d[\sigma^2(t)]}{dt}}dw.$
- Variance Preserving SDE: $dx = -\frac{1}{2}\beta(t)xdt + \sqrt{\beta(t)}dw.$
- $\sigma(t), \beta(t)$ are hyper-parameters.

Reversing the SDE

- Reversing the SDE: finding some stochastic process that goes from noise to data.
 - Use to generate data!
- Theorem (Anderson '82): there exists a reversing SDE, and it has a nice form:

$$dx = [f(x, t) - g^2(t) \nabla_x \log p_t(x)]dt + g(t)dw$$

- Strategy: learn the score function, then solve this reverse SDE.

Reversing the SDE

- Learning the score function: use score matching!

$$\arg \min_{\theta} \sum_i \lambda(\sigma_i) \mathbb{E}_{x \sim p_{\sigma_i, data}} \|s_{\theta}(x, i) - \nabla_x \log p_{\sigma_i, data}(x)\|^2$$

$$\Rightarrow \arg \min_{\theta} \mathbb{E}_{t \sim \text{unif}[0, T]} \mathbb{E}_{p_t(x)} \left[\lambda(t) \|s_{\theta}(x, t) - \nabla_x \log p_t(x)\|^2 \right]$$

- Use existing techniques: sliced score matching
- No need to tune temperature schedule
 - Still need to choose a forward SDE, $\lambda(\sigma_i)$, etc
 - Typically choose $\lambda(t) \propto 1/\mathbb{E} \left[\|\nabla_{x(t)} \log p(x(t) | x(0))\|^2 \right]$

Sampling by Solving the Reverse SDE

$$dx = [f(x, t) - g^2(t) \nabla_x \log p_t(x)]dt + g(t)dw$$

- Euler-Maruyama discretization:
 - $\Delta x \leftarrow [f(x, t) - g^2(t)s_\theta(x, t)]\Delta t + g(t)\sqrt{\Delta t}z_t$
 - $x \leftarrow x + \Delta x$
 - $t \leftarrow t + \Delta t$
- Other solvers:
 - Runge-Kutta
 - Predictor-corrector (Song et al. '21)

Evaluating Probability by Converting to ODE

- De-randomizing SDE

$$dx = [f(x, t) - g^2(t) \nabla_x \log p_t(x)]dt + g(t)dw$$

$$dx = [f(x, t) - g^2(t) \nabla_x \log p_t(x)]dt, x(T) \sim p_T$$

- Given an initial distribution and an ODE, we can evaluate probability at any time
 - Say given $x(T) \sim p_T$ and $dx = f(x, t)dt$

$$\log p_0(x(0)) = \log p_T(X(T)) + \int_0^T \text{Tr}(Df_\theta(x, t))dt$$

- Solve via ODE.