MFC CDT Probability and Statistics
Week 8

O. Deniz Akyildiz

Mathematics for our Future Climate: Theory, Data and Simulation (MFC CDT).

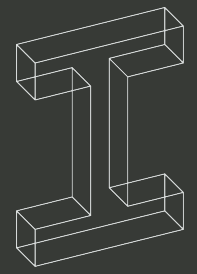
November 25, 2024

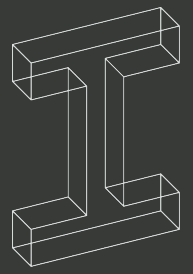
IMPERIAL

https://akyildiz.me/

𝕏: @odakyildiz

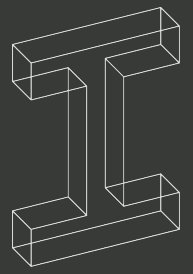
So far, we have covered the basic sampling techniques:





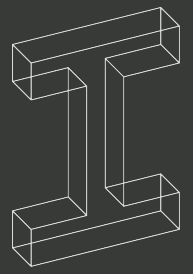
So far, we have covered the basic sampling techniques:

- Uniform random number generation
  - Linear congruential generators



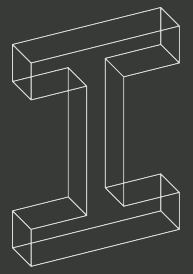
So far, we have covered the basic sampling techniques:

- Uniform random number generation
  - Linear congruential generators
- Inversion (inverse transform) sampling
  - $U \sim \mathcal{U}(0, 1)$
  - $X = F^{-1}(U)$



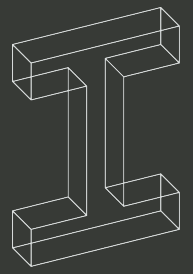
So far, we have covered the basic sampling techniques:

- Uniform random number generation
  - Linear congruential generators
- Inversion (inverse transform) sampling
  - $U \sim \mathcal{U}(0, 1)$
  - $X = F^{-1}(U)$
- Rejection sampling
  - $X' \sim q(x)$
  - Accept X' with probability $\Pi(X')/Mq(X')$



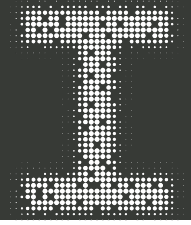
So far, we have covered the basic sampling techniques:

- Uniform random number generation
  - Linear congruential generators
- Inversion (inverse transform) sampling
  - $U \sim \mathcal{U}(0,1)$
  - $X = F^{-1}(U)$
- Rejection sampling
  - $X' \sim q(x)$
  - Accept X' with probability $\Pi(X')/Mq(X')$
- Importance sampling
  - Sample $X_1, \ldots, X_N \sim q(x)$
  - Estimate $(\varphi, \pi) \approx \sum_{i=1}^{N} \varphi(X_i)\mathrm{w}_i$,



So far, we have covered the basic sampling techniques:

- Uniform random number generation
  - Linear congruential generators
- Inversion (inverse transform) sampling
  - $U \sim \mathcal{U}(0,1)$
  - $X = F^{-1}(U)$
- Rejection sampling
  - $X' \sim q(x)$
  - Accept X' with probability $\Pi(X')/Mq(X')$
- Importance sampling
  - Sample $X_1, \ldots, X_N \sim q(x)$
  - Estimate $(\varphi, \pi) \approx \sum_{i=1}^{N} \varphi(X_i)\mathrm{w}_i$,
- Metropolis-Hastings algorithm



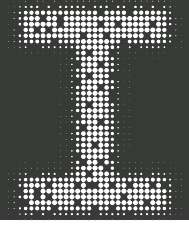
Let us look at now the Bayesian inference problem.

We can solve it in full generality (in theory) using MH.

Recall the general formulation

$$p(x|y_{1:n}) = \frac{p(y_{1:n}|x)p(x)}{p(y_{1:n})} = \frac{\prod_{i=1}^{n} p(y_i|x)p(x)}{p(y_{1:n})},$$

when $y_1, \ldots, y_n$ are conditionally independent given x.



We write

$$p(x|y_{1:n}) \propto \prod_{i=1}^{n} p(y_i|x)p(x),$$

and set

$$\gamma(x) = \prod_{i=1}^{n} p(y_i|x)p(x),$$

as our unnormalised posterior.

Metropolis-Hastings

Bayesian inference with MH

The generic MH for Bayesian inference, given x_{n-1}

- Sample $X' \sim q(x'|x_{n-1})$.
- Accept $x_n = x'$ with probability

$$\alpha(x_{n-1}, x') = \min\left\{1, \frac{\gamma(x')q(x_{n-1}|x')}{\gamma(x_{n-1})q(x'|x_{n-1})}\right\}.$$

- Otherwise, $X_n = x_{n-1}$.

Metropolis-Hastings

Example: Source localisation

Recall our example about localising a source using observations from a sensor network.

We can now formalise this problem. Assume that the source is located at $x \in \mathbb{R}^2$ and the sensor network is located at $s_1, \ldots, s_3 \in \mathbb{R}^2$ (3 sensors).

Assume that these three sensors ”observe” the source according to:

$$p(y_i|x, s_i) = \mathcal{N}(y_i; \|x - s_i\|, R),$$

where y_i is the observation from sensor i.

Metropolis-Hastings

Example: Source localisation

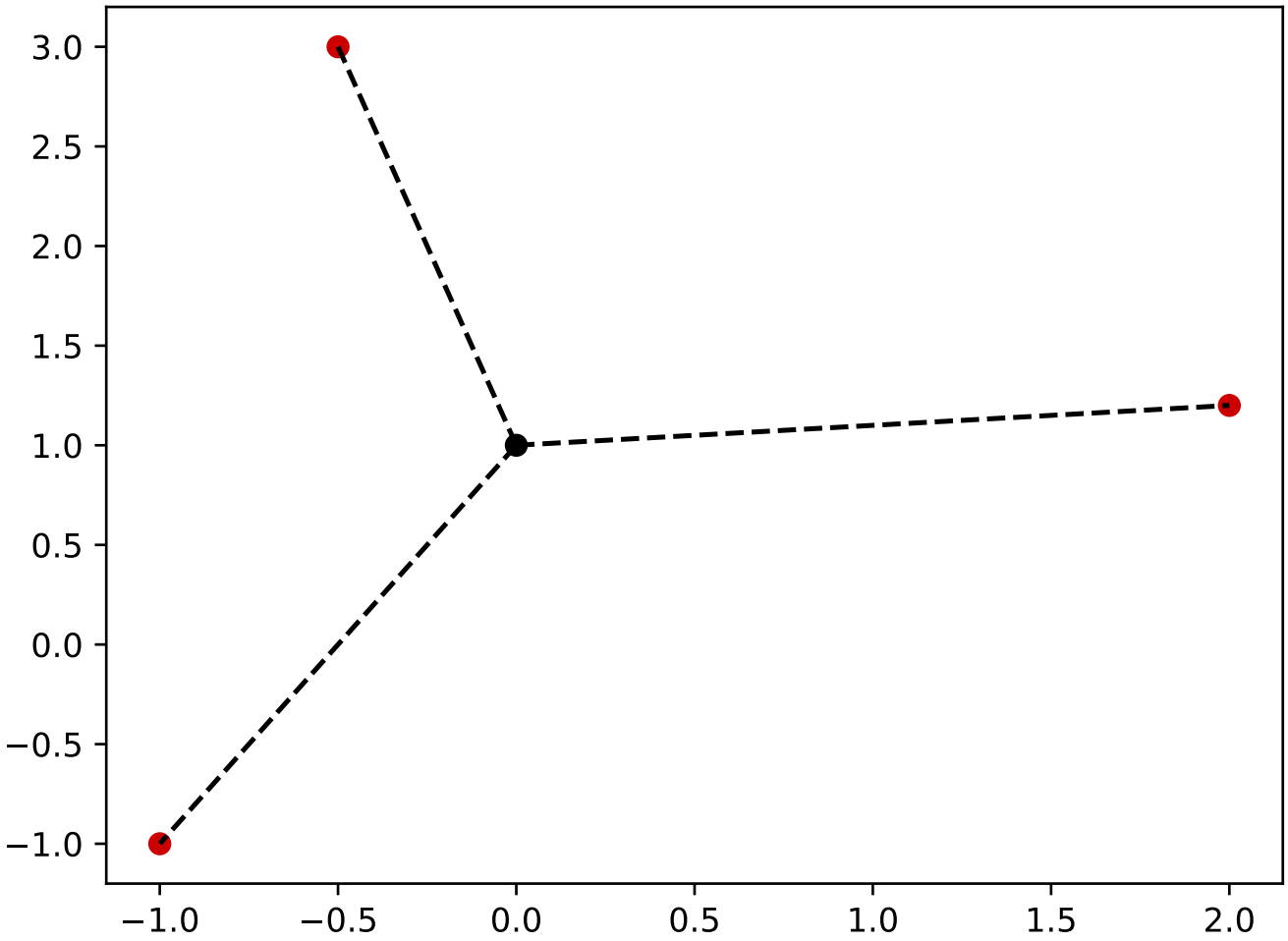


Figure: Source localisation

Metropolis-Hastings

Example: Source localisation

Assume that you are asked to estimate the location of the source given the observations y_1, y_2, y_3. What is the model?

Assume that you are asked to estimate the location of the source given the observations y_1, y_2, y_3. What is the model?

We first need a prior on the source location:

$$p(x) = \mathcal{N}(x; \mu, \Sigma),$$

where μ is the prior mean and Σ is the prior covariance. We already have the likelihoods for each y_i.

The posterior is given by

$$p(x|y_1, y_2, y_3, s_1, s_2, s_3) \propto p(x) \prod_{i=1}^{3} p(y_i|x, s_i).$$

Metropolis-Hastings

Example: Source localisation

We choose a random walk proposal:

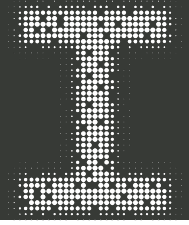
$$q(x'|x) = \mathcal{N}(x'; x, \sigma^2 I).$$

This is symmetric so the acceptance ratio is:

$$\mathrm{r}(x, x') = \frac{p(x')p(y_1|x', s_1)p(y_2|x', s_2)p(y_3|x', s_3)}{p(x)p(y_1|x, s_1)p(y_2|x, s_2)p(y_3|x, s_3)}.$$

Recap on sampling

Metropolis-Hastings



The black-box way of doing it is to use Metropolis-Hastings.

Recap on sampling

Metropolis-Hastings

The black-box way of doing it is to use Metropolis-Hastings.

- Sample $X' \sim q(x'|X_{n-1})$
- Set $X_n = X'$ with probability

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

- Otherwise, set $X_n = X_{n-1}$.

where $\Pi(x) = \exp(-U(x))$.

Recap on sampling

Metropolis-Hastings

The black-box way of doing it is to use Metropolis-Hastings.

- Sample $X' \sim q(x'|X_{n-1})$
- Set $X_n = X'$ with probability

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

- Otherwise, set $X_n = X_{n-1}$.

where $\Pi(x) = \exp(-U(x))$.

This leaves π invariant.

Recap on sampling

Metropolis adjusted Langevin algorithm

The choice of the proposal is crucial. One very popular choice is the Langevin proposal.

Recap on sampling

Metropolis adjusted Langevin algorithm

The choice of the proposal is crucial. One very popular choice is the Langevin proposal.

$$X' = X_{n-1} - \gamma \nabla U(X_{n-1}) + \sqrt{2\gamma} Z_n$$

where $Z_n \sim \mathcal{N}(0, I)$ and γ is the step size.

Recap on sampling

Metropolis adjusted Langevin algorithm

The choice of the proposal is crucial. One very popular choice is the Langevin proposal.

$$X' = X_{n-1} - \gamma \nabla U(X_{n-1}) + \sqrt{2\gamma} Z_n$$

where $Z_n \sim \mathcal{N}(0, I)$ and γ is the step size. This results in

$$q(x'|x) = \mathcal{N}(x - \gamma \nabla U(x), 2\gamma I).$$

Recap on sampling

Metropolis adjusted Langevin algorithm

The choice of the proposal is crucial. One very popular choice is the Langevin proposal.

$$X' = X_{n-1} - \gamma \nabla U(X_{n-1}) + \sqrt{2\gamma} Z_n$$

where $Z_n \sim \mathcal{N}(0, I)$ and γ is the step size. This results in

$$q(x'|x) = \mathcal{N}(x - \gamma \nabla U(x), 2\gamma I).$$

Why is this a good choice?

Consider the Langevin SDE for a generic drift ∇V:

$$\mathrm{d}X_t = -\nabla V(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion.

Consider the Langevin SDE for a generic drift ∇V:

$$\mathrm{d}X_t = -\nabla V(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion. This SDE has a stationary measure

$$\pi \propto e^{-V(x)}.$$

Therefore, for a classical *sampling* problem for, say $\pi(x)$, we could set $V(x) = -\log \pi(x)$ (negative density).

Consider the Langevin SDE for a generic drift ∇V:

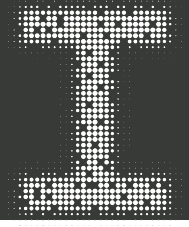
$$\mathrm{d}X_t = -\nabla V(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion. This SDE has a stationary measure

$$\pi \propto e^{-V(x)}.$$

Therefore, for a classical *sampling* problem for, say $\pi(x)$, we could set $V(x) = -\log \pi(x)$ (negative density).

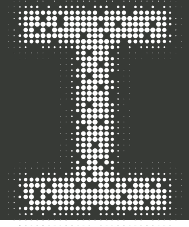
This diffusion converges to its stationary measure exponentially fast if V is μ-strongly-convex.



Consider the Langevin SDE for a generic drift ∇V:

$$\mathrm{d}X_t = -\nabla V(X_t)\mathrm{d}t + \sqrt{\frac{2}{\beta}}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion.

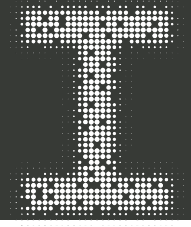


Consider the Langevin SDE for a generic drift ∇V:

$$\mathrm{d}X_t = -\nabla V(X_t)\mathrm{d}t + \sqrt{\frac{2}{\beta}}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion. This SDE has a stationary measure

$$\pi \propto e^{-\beta V(x)}.$$



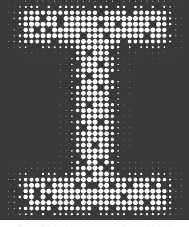
Consider the Langevin SDE for a generic drift ∇V:

$$\mathrm{d}X_t = -\nabla V(X_t)\mathrm{d}t + \sqrt{\frac{2}{\beta}}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion. This SDE has a stationary measure

$$\pi \propto e^{-\beta V(x)}.$$

This stationary measure concentrates on the minima of V as $\beta \to \infty$ (Hwang, 1980).



Consider the Langevin SDE for a generic drift ∇V:

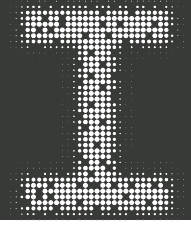
$$\mathrm{d}X_t = -\nabla V(X_t)\mathrm{d}t + \sqrt{\frac{2}{\beta}}\mathrm{d}B_t,$$

where $(B_t)_{t\geq 0}$ is a Brownian motion. This SDE has a stationary measure

$$\pi \propto e^{-\beta V(x)}.$$

This stationary measure concentrates on the minima of V as $\beta \to \infty$ (Hwang, 1980).

Langevin diffusion is a global optimiser.



The Euler discretisation is the *unadjusted Langevin algorithm* (ULA):

$$X_{t+1}^{\gamma} = X_t^{\gamma} - \gamma \nabla V(X_t^{\gamma}) + \sqrt{2\gamma} W_{t+1}$$

where $(W_t)_{t \geq 0}$ are i.i.d standard Normal random variables.

Langevin-based approaches

Crash course on Langevin SDE - III: Numerical discretisation

The Euler discretisation is the *unadjusted Langevin algorithm* (ULA):

$$X_{t+1}^{\gamma} = X_t^{\gamma} - \gamma \nabla V(X_t^{\gamma}) + \sqrt{2\gamma} W_{t+1}$$

where $(W_t)_{t \geq 0}$ are i.i.d standard Normal random variables.

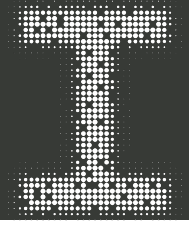
This chain has a *different* stationary measure π^{γ} but a number of guarantees can be derived for its convergence.

Theorem 1 (Durmus and Moulines, 2019)

Let $\mathcal{L}(X_t)$ be the law of the iterates of ULA, then

$$W_2^2(\mathcal{L}(X_t^{\gamma}), \pi) \lesssim \left(1 - \frac{\gamma\kappa}{2}\right)^{t+1} (d/m + \|x - x^{\star}\|^2) + \gamma,$$

under suitable regularity conditions for V, restriction on γ where $\kappa := \kappa(m, L)$.

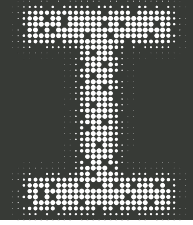


An important note here is that, we can sample from the posterior $p(x|y)$ using ULA as

$$p(x|y) \propto p(x,y),$$

and

$$X_{n+1}^{\gamma} = X_n^{\gamma} + \gamma \nabla \log p(X_n^{\gamma}, y) + \sqrt{2}\gamma W_{n+1}.$$



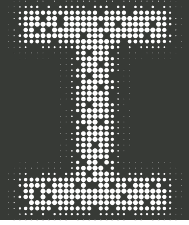
An important note here is that, we can sample from the posterior $p(x|y)$ using ULA as

$$p(x|y) \propto p(x,y),$$

and

$$X_{n+1}^{\gamma} = X_n^{\gamma} + \gamma \nabla \log p(X_n^{\gamma}, y) + \sqrt{2}\gamma W_{n+1}.$$

We can see that this algorithm would approximately sample from $p(x|y)$.

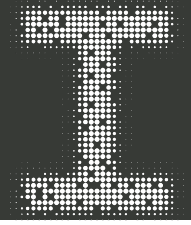


Let us say we have data $y_1, \ldots, y_M$ for M large. We can write the posterior as

$$p(x|y_{1:M}) \propto p(x) \prod_{i=1}^{M} p(y_i|x).$$

therefore, our potential becomes

$$V(x) = -\log p(x) - \sum_{i=1}^{M} \log p(y_i|x).$$



Let us say we have data $y_1, \ldots, y_M$ for M large. We can write the posterior as

$$p(x|y_{1:M}) \propto p(x) \prod_{i=1}^{M} p(y_i|x).$$

therefore, our potential becomes

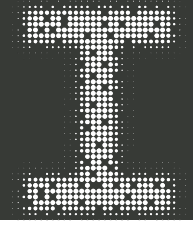
$$V(x) = -\log p(x) - \sum_{i=1}^{M} \log p(y_i|x).$$

Mini-quiz: What is the problem with MALA (or MH in general) in this case?

Langevin-based approaches

ULA for Bayesian inference

A similar problem of course would be for ULA.

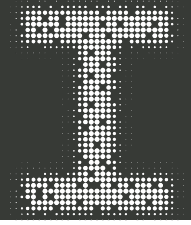


A similar problem of course would be for ULA.

However, we can resolve this, as we can approximate the gradient using subsampling:

$$\begin{aligned}\nabla V(x) &= \nabla \log p(x) + \sum_{i=1}^{M} \nabla \log p(y_i|x), \\ &\approx \nabla \log p(x) + \frac{M}{m} \sum_{j=1}^{m} \nabla \log p(y_{k_j}|x) = \widehat{\nabla V(x)},\end{aligned}$$

where $k_j \sim \text{Unif}\{1, \ldots, M\}$, for $j = 1, \ldots, m$ for $m \ll M$.



A similar problem of course would be for ULA.

However, we can resolve this, as we can approximate the gradient using subsampling:

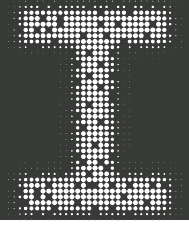
$$\begin{aligned}\nabla V(x) &= \nabla \log p(x) + \sum_{i=1}^{M} \nabla \log p(y_i|x), \\ &\approx \nabla \log p(x) + \frac{M}{m} \sum_{j=1}^{m} \nabla \log p(y_{k_j}|x) = \widehat{\nabla V(x)},\end{aligned}$$

where $k_j \sim \text{Unif}\{1, \ldots, M\}$, for $j = 1, \ldots, m$ for $m \ll M$.

Stochastic gradients.

One can run ULA with stochastic gradients:

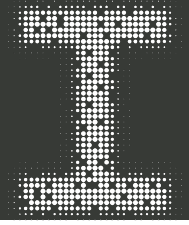
$$X_{n+1}^{\gamma} = X_n^{\gamma} - \gamma \widehat{\nabla V(X_n^{\gamma})} + \sqrt{2}\gamma W_{n+1}.$$



One can run ULA with stochastic gradients:

$$X_{n+1}^{\gamma} = X_n^{\gamma} - \gamma \widehat{\nabla V(X_n^{\gamma})} + \sqrt{2}\gamma W_{n+1}.$$

The resulting method is called *stochastic gradient Langevin dynamics* (SGLD) (Welling and Teh, 2011).



One can run ULA with stochastic gradients:

$$X_{n+1}^{\gamma} = X_n^{\gamma} - \gamma \widehat{\nabla V(X_n^{\gamma})} + \sqrt{2}\gamma W_{n+1}.$$

The resulting method is called *stochastic gradient Langevin dynamics* (SGLD) (Welling and Teh, 2011).

- Widely used for large-scale datasets.

One can run ULA with stochastic gradients:

$$X_{n+1}^{\gamma} = X_n^{\gamma} - \gamma \widehat{\nabla V(X_n^{\gamma})} + \sqrt{2}\gamma W_{n+1}.$$

The resulting method is called *stochastic gradient Langevin dynamics* (SGLD) (Welling and Teh, 2011).

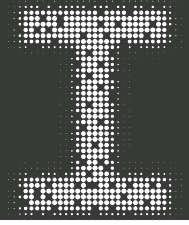
- Widely used for large-scale datasets.
- It has similar guarantees to ULA in Wasserstein-2 distance for strongly convex V.

One can run ULA with stochastic gradients:

$$X_{n+1}^{\gamma} = X_n^{\gamma} - \gamma \widehat{\nabla V(X_n^{\gamma})} + \sqrt{2}\gamma W_{n+1}.$$

The resulting method is called *stochastic gradient Langevin dynamics* (SGLD) (Welling and Teh, 2011).

- Widely used for large-scale datasets.
- It has similar guarantees to ULA in Wasserstein-2 distance for strongly convex V.
- Also used to model and analyse the behaviour of stochastic gradient descent methods (SGD) in deep learning.



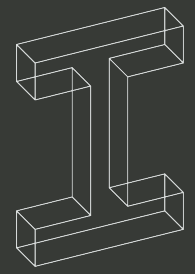
One can run ULA with stochastic gradients:

$$X_{n+1}^{\gamma} = X_n^{\gamma} - \gamma \widehat{\nabla V(X_n^{\gamma})} + \sqrt{2}\gamma W_{n+1}.$$

The resulting method is called *stochastic gradient Langevin dynamics* (SGLD) (Welling and Teh, 2011).

- Widely used for large-scale datasets.
- It has similar guarantees to ULA in Wasserstein-2 distance for strongly convex V.
- Also used to model and analyse the behaviour of stochastic gradient descent methods (SGD) in deep learning.

Web based simulations if time permits.



We have seen approaches so far in **sampling**.

Next: An introduction to generative modelling

Recap on sampling

What is sampling, what is generative modelling?

Sampling: Given a probability measure π, output (approximately) a sample $X \sim \pi$.

Recap on sampling

What is sampling, what is generative modelling?

Sampling: Given a probability measure π, output (approximately) a sample $X \sim \pi$.

- Typically $\pi \propto \exp(-U(x))$ where $U(x)$ is a potential (or energy) function.

Recap on sampling

What is sampling, what is generative modelling?

Sampling: Given a probability measure π, output (approximately) a sample $X \sim \pi$.

- Typically $\pi \propto \exp(-U(x))$ where $U(x)$ is a potential (or energy) function.

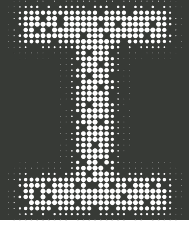
Generative modelling: Given a dataset $\{x_i\}_{i=1}^n$ or given an empirical measure

$$\hat{p}_{\text{data}} = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i},$$

output (approximately) a sample $X \sim p_{\text{data}}$.

Recap on sampling

Markov chain Monte Carlo



Sampling is a very general problem and a very old topic in statistics and computer science. One of the most popular methods is Markov chain Monte Carlo (MCMC).

Recap on sampling

Markov chain Monte Carlo

Sampling is a very general problem and a very old topic in statistics and computer science. One of the most popular methods is Markov chain Monte Carlo (MCMC).

- Scales favourably with dimension

Recap on sampling

Markov chain Monte Carlo

Sampling is a very general problem and a very old topic in statistics and computer science. One of the most popular methods is Markov chain Monte Carlo (MCMC).

- Scales favourably with dimension
- Requires only the ability to evaluate the density up to a normalizing constant

Recap on sampling

Markov chain Monte Carlo

Sampling is a very general problem and a very old topic in statistics and computer science. One of the most popular methods is Markov chain Monte Carlo (MCMC).

- Scales favourably with dimension
- Requires only the ability to evaluate the density up to a normalizing constant

The principle is to construct a Markov chain with stationary distribution π and sample from it.

Energy Based Models

Introduction

Given $\hat{p}_{\text{data}}$, we would like to estimate the density p_{data}.

Energy Based Models

Introduction

Given $\hat{p}_{\text{data}}$, we would like to estimate the density p_{data}.

- This is a classical density estimation problem.

Energy Based Models

Introduction

Given $\hat{p}_{\text{data}}$, we would like to estimate the density p_{data}.

- This is a classical density estimation problem.

Discussion: Can we use kernel density estimation for this problem?

Energy Based Models

Introduction

Given $\hat{p}_{\text{data}}$, we would like to estimate the density p_{data}.

- This is a classical density estimation problem.

Discussion: Can we use kernel density estimation for this problem?

What is a sensible parametric model?

Energy Based Models

Introduction

Consider the Gibbs-type probability measure

$$\pi_\theta \propto \exp(-U_\theta(x)),$$

where $U_\theta(x)$ is a potential function.

Energy Based Models

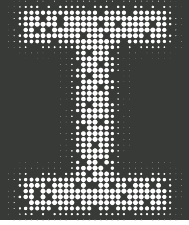
Introduction

Consider the Gibbs-type probability measure

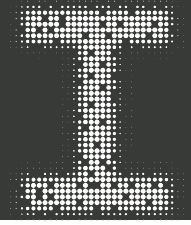
$$\pi_\theta \propto \exp(-U_\theta(x)),$$

where $U_\theta(x)$ is a potential function.

When is this a flexible model for p_{data}?



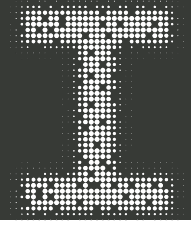
Recall that if $U_\theta(x)$ is a neural network, then $U_\theta(x)$ is a universal approximator.



Recall that if $U_\theta(x)$ is a neural network, then $U_\theta(x)$ is a universal approximator.

For any continuous function $U(x)$, there exists a neural network $U_\theta(x)$ such that

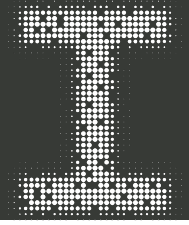
$$\|U(x) - U_\theta(x)\|_\infty < \epsilon.$$



Recall that if $U_\theta(x)$ is a neural network, then $U_\theta(x)$ is a universal approximator.

For any continuous function $U(x)$, there exists a neural network $U_\theta(x)$ such that

$$\|U(x) - U_\theta(x)\|_\infty < \epsilon.$$



Let $U : \mathrm{X} \to \mathbb{R}$ and $U_\theta : \mathrm{X} \to \mathbb{R}$ be a neural network. Let $\pi \propto \exp(-U(x))$ and $\pi_\theta \propto \exp(-U_\theta(x))$.

Let $U : \mathrm{X} \to \mathbb{R}$ and $U_\theta : \mathrm{X} \to \mathbb{R}$ be a neural network. Let $\pi \propto \exp(-U(x))$ and $\pi_\theta \propto \exp(-U_\theta(x))$. Then (Atchadé et al., 2023)

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

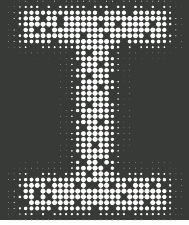
where m is the Lebesgue measure on X (with finite measure).

Let $U : \mathrm{X} \to \mathbb{R}$ and $U_\theta : \mathrm{X} \to \mathbb{R}$ be a neural network. Let $\pi \propto \exp(-U(x))$ and $\pi_\theta \propto \exp(-U_\theta(x))$. Then (Atchadé et al., 2023)

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

where m is the Lebesgue measure on X (with finite measure).

Therefore, an EBM is a universal approximator on bounded spaces.



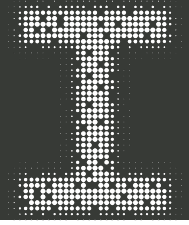
Training EBMs is a hard problem and there is a VAST literature on this topic.

Training EBMs is a hard problem and there is a VAST literature on this topic.

We will review (expanding):

- Maximum Likelihood Estimation
- Score Matching

and variations of these.

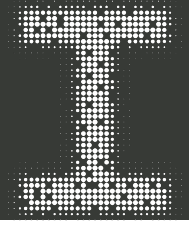


The idea is to maximise the expected likelihood of the data under the model. The *expected* log-likelihood of the data is given by

$$\ell(\theta) := \mathbb{E}_{p_{\text{data}}}[\log \pi_\theta(X)], \tag{1}$$

where p_{data} is the data distribution.

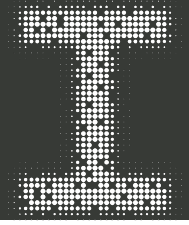
Maximising this is equivalent to minimising the KL divergence between p_{data} and π_θ.



Let us look at the objective a bit more closely.

$$\ell(\theta) = \mathbb{E}_{p_{\text{data}}}[\log \pi_\theta(X)] = \int \log \pi_\theta(x) p_{\text{data}}(x) \mathrm{d}x,$$
$$= -\int U_\theta(x) p_{\text{data}}(x) \mathrm{d}x - \log Z_\theta,$$

where $Z_\theta = \int \exp(-U_\theta(x)) \mathrm{d}x$ is the normalising constant.



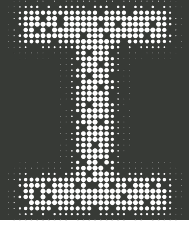
Let us look at the objective a bit more closely.

$$\ell(\theta) = \mathbb{E}_{p_{\text{data}}}[\log \pi_\theta(X)] = \int \log \pi_\theta(x) p_{\text{data}}(x) \mathrm{d}x,$$
$$= -\int U_\theta(x) p_{\text{data}}(x) \mathrm{d}x - \log Z_\theta,$$

where $Z_\theta = \int \exp(-U_\theta(x)) \mathrm{d}x$ is the normalising constant. For gradient based optimisation, we can consider

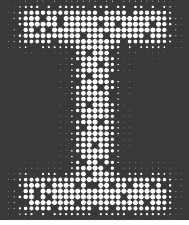
$$\nabla_\theta \ell(\theta) = -\mathbb{E}_{p_{\text{data}}}\left[\nabla_\theta U_\theta(X)\right] - \nabla_\theta \log Z_\theta.$$

The last term is intractable.



Assuming $\int e^{-U_\theta(x)} \mathrm{d}x < \infty$, we have

$$\nabla_\theta \log Z_\theta = -\mathbb{E}_{\pi_\theta}[\nabla_\theta U_\theta(X)].$$



Therefore, we have the full gradient that is of the form

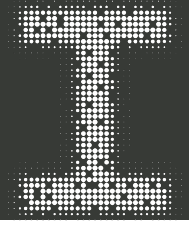
$$\nabla_\theta \ell(\theta) = -\mathbb{E}_{p_{\text{data}}}\left[\nabla_\theta U_\theta(X)\right] + \mathbb{E}_{\pi_\theta}\left[\nabla_\theta U_\theta(X)\right].$$

Therefore, we have the full gradient that is of the form

$$\nabla_\theta \ell(\theta) = -\mathbb{E}_{p_{\text{data}}}\left[\nabla_\theta U_\theta(X)\right] + \mathbb{E}_{\pi_\theta}\left[\nabla_\theta U_\theta(X)\right].$$

We get then its empirical approximation

$$\nabla_\theta \ell_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta U_\theta(x_i) + \mathbb{E}_{\pi_\theta}\left[\nabla_\theta U_\theta(X)\right].$$



Therefore, we have the full gradient that is of the form

$$\nabla_\theta \ell(\theta) = -\mathbb{E}_{p_{\text{data}}}\left[\nabla_\theta U_\theta(X)\right] + \mathbb{E}_{\pi_\theta}\left[\nabla_\theta U_\theta(X)\right].$$

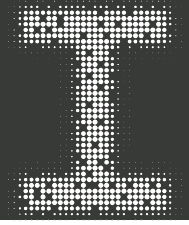
We get then its empirical approximation

$$\nabla_\theta \ell_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta U_\theta(x_i) + \mathbb{E}_{\pi_\theta}\left[\nabla_\theta U_\theta(X)\right].$$

The second term is still problematic!

$$\nabla_\theta \ell_n(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta U_\theta(x_i) + \mathbb{E}_{\pi_\theta}[\nabla_\theta U_\theta(X)].$$

The idea here is to use MCMC to sample from π_θ to approximate the expectation.



Consider the following optimization (gradient ascent) procedure

$$\begin{aligned}\theta_{k+1} &= \theta_k + \delta \nabla_\theta \ell_n(\theta_k), \\ &= \theta_k - \frac{\delta}{n} \sum_{i=1}^{n} \nabla U_{\theta_k}(x_i) + \delta \mathbb{E}_{\pi_{\theta_k}}[\nabla U_{\theta_k}(X)]\end{aligned}$$

Consider the following optimization (gradient ascent) procedure

$$\begin{aligned}\theta_{k+1} &= \theta_k + \delta \nabla_\theta \ell_n(\theta_k), \\ &= \theta_k - \frac{\delta}{n} \sum_{i=1}^{n} \nabla U_{\theta_k}(x_i) + \delta \mathbb{E}_{\pi_{\theta_k}}[\nabla U_{\theta_k}(X)]\end{aligned}$$

At iteration k, the expectation needs to be approximated, For this, we run a separate ULA chain at each k:

$$X_k^{(m)} = X_k^{(m-1)} - \gamma \nabla U_{\theta_k}(X_k^{(m-1)}) + \sqrt{2\gamma} W_k^{(m)},$$

where $(W_k^{(m)})_{m \geq 0}$ are i.i.d standard Normal random variables.

Algorithm MLE for EBMs via ULA

Input: The dataset $\{x_i\}_{i=1}^n$, the step size δ, the number of MCMC steps M, the burn-in B, the step size γ, the number of iterations N.
for $k = 1, \ldots, K$ **do**
    Fix $X_k^{(0)}$
    for $m = 1, \ldots, M$ **do**
        $X_k^{(m)} = X_k^{(m-1)} - \gamma \nabla U_{\theta_k}(X_k^{(m-1)}) + \sqrt{2\gamma} W_k^{(m)}$.
    end for
    $\nabla \ell_n(\theta_k) = -\frac{1}{n} \sum_{i=1}^n \nabla U_{\theta_k}(x_i) + \frac{1}{M-B} \sum_{m=B+1}^M \nabla U_{\theta_k}(X_k^{(m)})$.
    $\theta_{k+1} = \theta_k + \delta \nabla \ell_n(\theta_k)$.
end for

Algorithm Pseudocode for EBM training via CD-1

Input: The dataset $\{x_i\}_{i=1}^n$, the step size δ, the number of MCMC steps M, the burn-in B, the step size γ, the number of iterations N.
for $k = 1, \ldots, K$ **do**
    Randomly choose i.
    $X_k^{(0)} = x_i$.
    $X_k^{(1)} = X_k^{(0)} - \gamma \nabla U_{\theta_k}(X_k^{(0)}) + \sqrt{2\gamma} W_k^{(1)}$.
    $\nabla \ell_n(\theta_k) = -\frac{1}{n} \sum_{i=1}^n \nabla U_{\theta_k}(x_i) + \nabla U_{\theta_k}(X_k^{(1)})$.
    $\theta_{k+1} = \theta_k + \delta \nabla \ell_n(\theta_k)$.
end for

Often with the stochastic gradients:

$$n^{-1} \sum_{i=1}^{n} \nabla U_{\theta_k}(x_i) \approx J^{-1} \sum_{j=1}^{J} \nabla U_{\theta_k}(x_{i_j}).$$

Algorithm Pseudocode for EBM training via PCD

Input: The dataset $\{x_i\}_{i=1}^n$, the step size δ, the number of MCMC steps M, the burn-in B, the step size γ, the number of iterations N.
for $k = 1, \ldots, K$ **do**
    for $i = 1, \ldots, N$ **do**
        $X_k^{(i)} = X_k^{(i)} - \gamma \nabla U_{\theta_k}(X_k^{(i)}) + \sqrt{2\gamma} W_k^{(i)}$.
    end for
    $\nabla \ell_n(\theta_k) = -n^{-1} \sum_{i=1}^n \nabla U_{\theta_k}(x_i) + N^{-1} \sum_{i=1}^N \nabla U_{\theta_k}(X_k^{(i)})$.
    $\theta_{k+1} = \theta_k + \delta \nabla \ell_n(\theta_k)$.
end for

Algorithm "Short-Run MCMC" (Nijkamp et al., 2019)

Input: The dataset $\{x_i\}_{i=1}^n$, the step size δ, the number of MCMC steps M, the burn-in B, the step size γ, the number of iterations N.
for $k = 1, \ldots, K$ **do**
    Fix $X_k^{(0)} \sim p_0$
    for $m = 1, \ldots, M$ **do**
        $X_k^{(m)} = X_k^{(m-1)} - \gamma \nabla U_{\theta_k}(X_k^{(m-1)}) + \sqrt{2\gamma} W_k^{(m)}$.
    end for
    $\nabla \ell_n(\theta_k) = -\frac{1}{n} \sum_{i=1}^n \nabla U_{\theta_k}(x_i) + \frac{1}{M-B} \sum_{m=B+1}^M \nabla U_{\theta_k}(X_k^{(m)})$.
    $\theta_{k+1} = \theta_k + \delta \nabla \ell_n(\theta_k)$.
end for

Note this method also noises the data with a single step every iteration.

We covered so far maximum likelihood estimation for EBMs.

We covered so far maximum likelihood estimation for EBMs.

- MLE training procedures are unstable and hard to tune.

We covered so far maximum likelihood estimation for EBMs.

- MLE training procedures are unstable and hard to tune.

Next: Energy Based Models and Score Matching.

Recall the Euler discretisation is the *unadjusted Langevin algorithm* (ULA):

$$X_{t+1}^{\gamma} = X_t^{\gamma} + \gamma \nabla \log \Pi(X_t^{\gamma}) + \sqrt{2\gamma} W_{t+1}$$

where $(W_t)_{t \geq 0}$ are i.i.d standard Normal random variables.

Recall the Euler discretisation is the *unadjusted Langevin algorithm* (ULA):

$$X_{t+1}^{\gamma} = X_t^{\gamma} + \gamma \nabla \log \Pi(X_t^{\gamma}) + \sqrt{2\gamma} W_{t+1}$$

where $(W_t)_{t \geq 0}$ are i.i.d standard Normal random variables.

Notice that we only need the gradient of the (unnormalised) log-density for sampling.

Recall the Euler discretisation is the *unadjusted Langevin algorithm* (ULA):

$$X_{t+1}^{\gamma} = X_t^{\gamma} + \gamma \nabla \log \Pi(X_t^{\gamma}) + \sqrt{2\gamma} W_{t+1}$$

where $(W_t)_{t \geq 0}$ are i.i.d standard Normal random variables.

Notice that we only need the gradient of the (unnormalised) log-density for sampling.

The idea of score matching is to directly estimate the gradient of the log-density.

Score matching methods are based on Fisher divergence, which is defined as

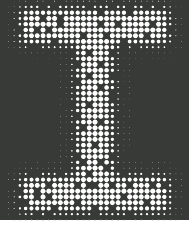
$$\mathrm{F}(\pi_1||\pi_2) = \frac{1}{2}\mathbb{E}_{\pi_1}[\|\nabla \log \pi_1(X) - \nabla \log \pi_2(X)\|^2]. \tag{2}$$

In the case of generative modelling, we are interested in computing $\theta^\star$ where

$$\theta^\star \in \operatorname*{argmin}_\theta \mathrm{F}(p_{\text{data}}||\pi_\theta),$$

where

$$\mathrm{F}(p_{\text{data}}||\pi_\theta) = \frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|\nabla \log p_{\text{data}}(X) - \nabla \log \pi_\theta(X)\|^2]. \tag{3}$$



Proposition 1 (Hyvärinen, 2005)

The loss in (3) *can be written as*

$$\mathrm{F}(p_{data}||\pi_\theta) = \mathbb{E}_{p_{data}}\left[\mathrm{Tr}\nabla^2 \log \pi_\theta(X) + \frac{1}{2}\|\nabla \log \pi_\theta(X)\|^2\right]. \quad (4)$$

Energy Based Models

Score Matching

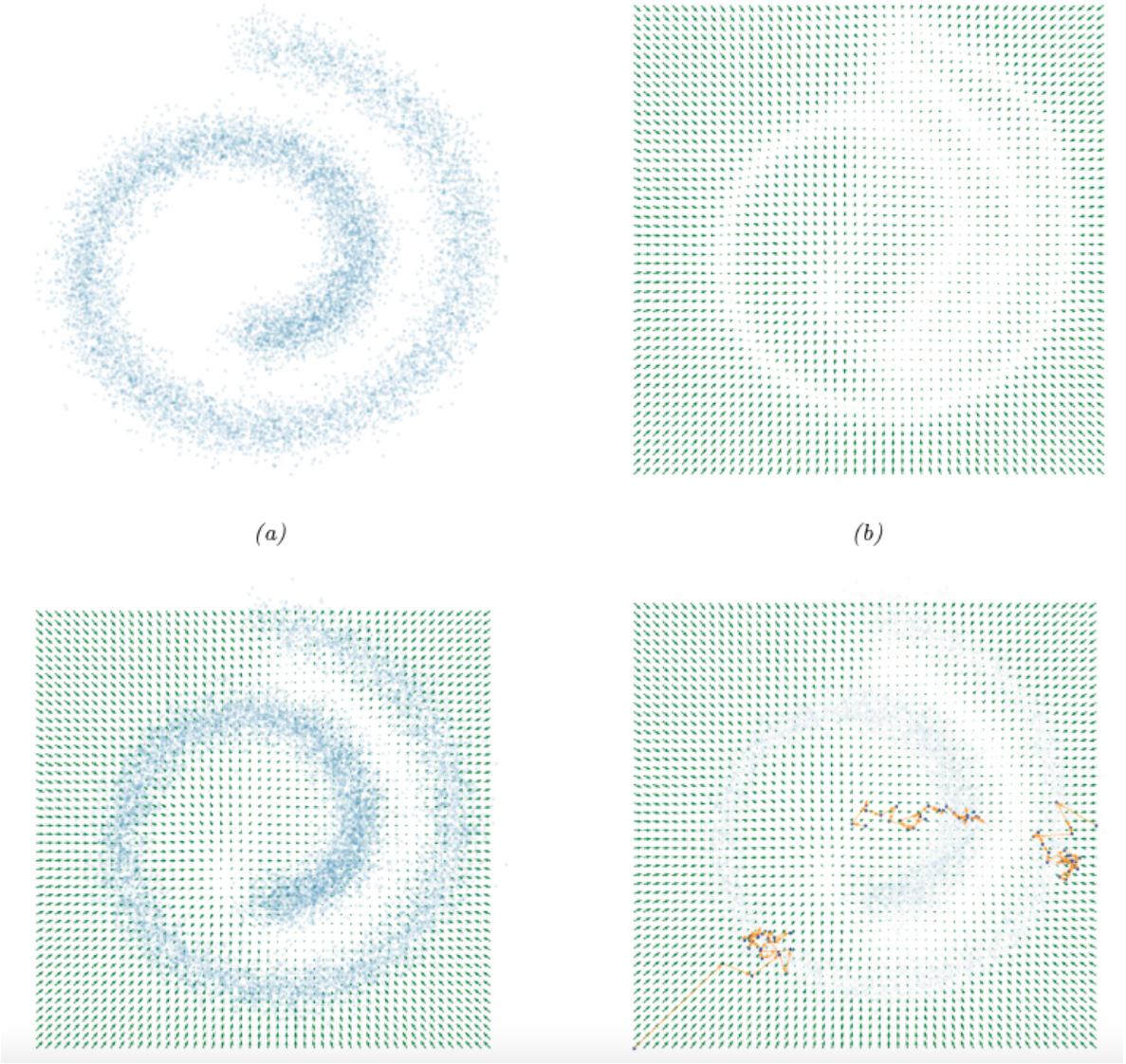


Figure: From Murphy (2023)

Unfortunately, this idea doesn't quite work in practice.

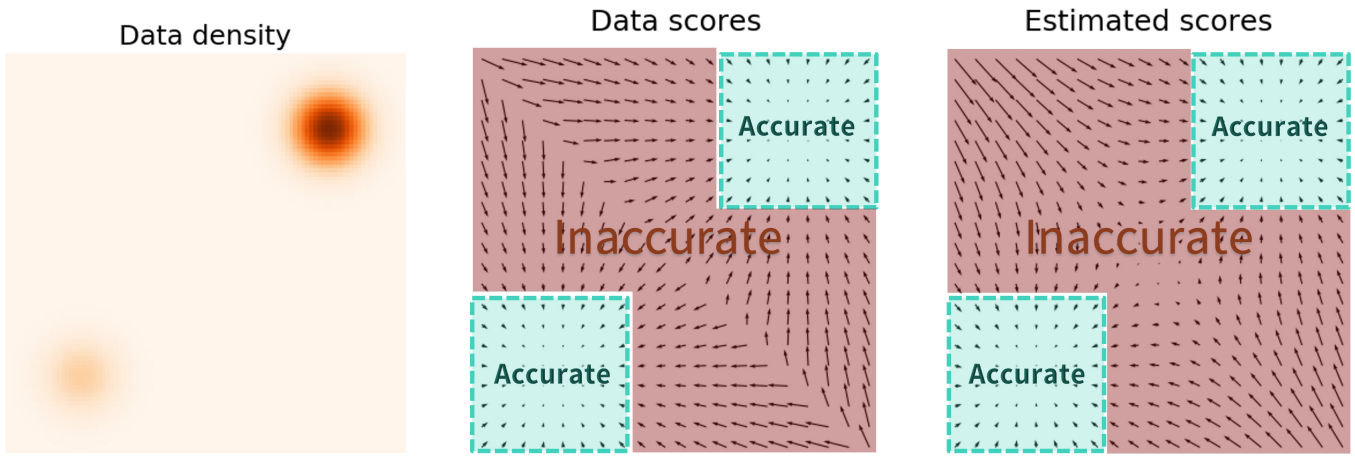


Image credit: `https://yang-song.net/blog/2021/score/`

Score matching can be inefficient and is expensive due to the Hessian term. Instead, we can leverage the following idea.

Let us define the noisy version of the data distribution as

$$p^{\sigma}_{\text{data}}(x) = \int p_{\text{data}}(x')K(x|x')\mathrm{d}x',$$

where $K(x|x')$ is a kernel.

Proposition 2 (Vincent, 2011)

Given the Fisher divergence between the noisy data distribution and the model distribution,

$$\mathrm{F}(p^{\sigma}_{data}||\pi_\theta) = \mathbb{E}_{p^{\sigma}_{data}}\left[\frac{1}{2}\|\nabla \log p^{\sigma}_{data}(X) - \nabla \log \pi_\theta(X)\|^2\right], \quad (5)$$

we have

$$\mathrm{F}(p^{\sigma}_{data}||\pi_\theta) = \mathbb{E}_{p(x,x')}\left[\frac{1}{2}\|\nabla_x \log K(X|X') - \nabla \log \pi_\theta(X)\|^2\right]. \quad (6)$$

where $p(x, x') = p_{data}(x')K(x|x')$.

Proposition 2 (Vincent, 2011)

Given the Fisher divergence between the noisy data distribution and the model distribution,

$$\mathrm{F}(p^{\sigma}_{data}||\pi_\theta) = \mathbb{E}_{p^{\sigma}_{data}}\left[\frac{1}{2}\|\nabla \log p^{\sigma}_{data}(X) - \nabla \log \pi_\theta(X)\|^2\right], \quad (5)$$

we have

$$\mathrm{F}(p^{\sigma}_{data}||\pi_\theta) = \mathbb{E}_{p(x,x')}\left[\frac{1}{2}\|\nabla_x \log K(X|X') - \nabla \log \pi_\theta(X)\|^2\right]. \quad (6)$$

where $p(x, x') = p_{data}(x')K(x|x')$.

For example, one can simply choose

$$K(x|x') = \mathcal{N}(x; x', \sigma^2 I_d),$$

where σ is a hyperparameter. In this situation, the estimate will take the form (Song and Kingma, 2021)

$$\mathrm{F}(p^{\sigma}_{\text{data}}||\pi_\theta) = \mathbb{E}_{p_{\text{data}}(x')}\mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2 I_d)}\left[\frac{1}{2}\left\|\frac{z}{\sigma} + \nabla_x \log \pi_\theta(x' + \sigma z)\right\|^2\right].$$

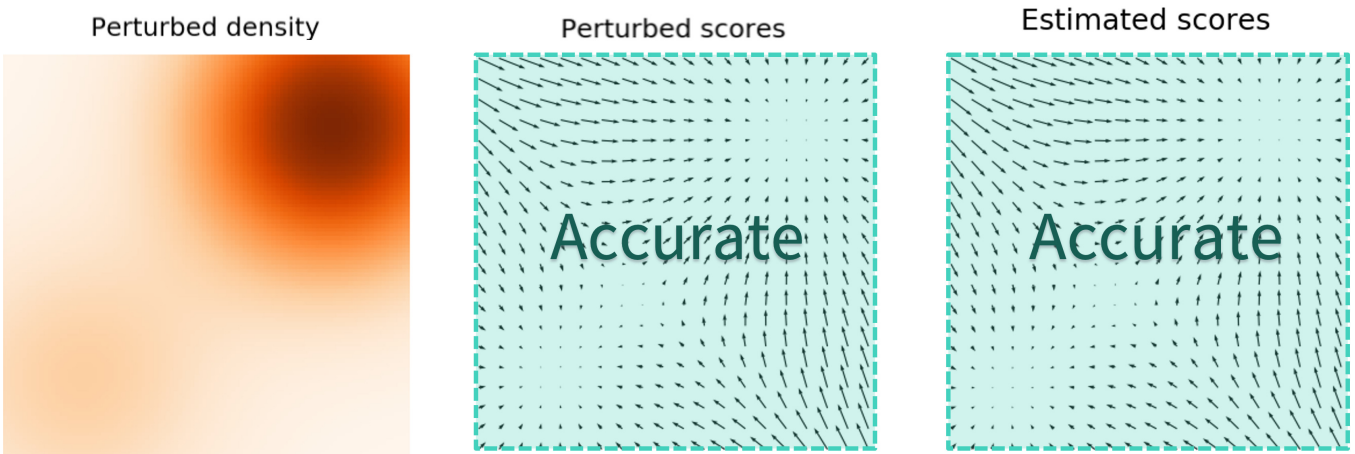
One can estimate this score, by plugging the empirical data distribution in using

$$\hat{p}_{\text{data}} = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}.$$

and using samples from the Gaussian distribution, which results in the unbiased estimate of the loss

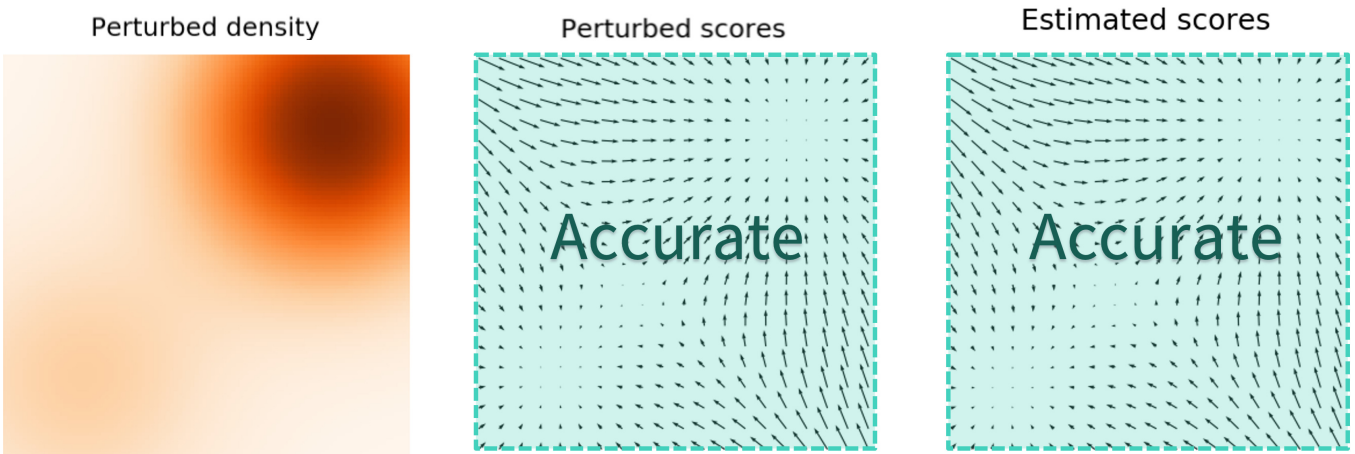
$$\hat{\mathrm{F}}(p^{\sigma}_{\text{data}} || \pi_\theta) = \frac{1}{2n} \sum_{i=1}^{n} \left\| \frac{z^{(i)}}{\sigma} + \nabla_x \log \pi_\theta(x_i + \sigma z^{(i)}) \right\|^2.$$

Perturbed scores are better behaved:



But there is a bias as we approximate the noisy data distribution.

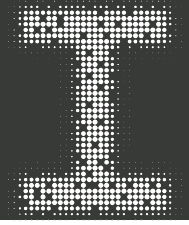
Perturbed scores are better behaved:



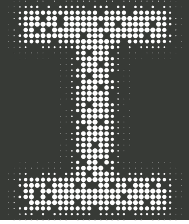
But there is a bias as we approximate the noisy data distribution.

This observation led to the idea of *adding* progressively more noise to data and learn associated scores (a bit later).

Image credit: `https://yang-song.net/blog/2021/score/`

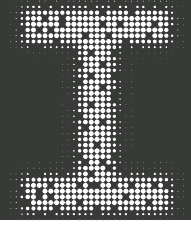


Denoising score matching appeared as the cheapest option to approximate scores, but has an inherent bias as it estimates the noisy scores.



Denoising score matching appeared as the cheapest option to approximate scores, but has an inherent bias as it estimates the noisy scores.

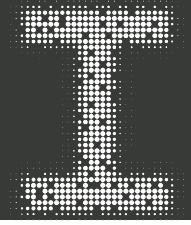
For $\sigma \approx 0$, the bias is small, but training is unstable and mixing can be arbitrarily slow, especially for p_{data} with multiple modes or which concentrates on low-dimensional manifolds.



Denoising score matching appeared as the cheapest option to approximate scores, but has an inherent bias as it estimates the noisy scores.

For $\sigma \approx 0$, the bias is small, but training is unstable and mixing can be arbitrarily slow, especially for p_{data} with multiple modes or which concentrates on low-dimensional manifolds.

Can we "bridge" easy to sample distributions and the data distribution, via the use of noise scales?



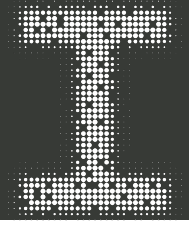
This is the idea in Song and Ermon (2019) that unlocked the path to diffusion models.

This is the idea in Song and Ermon (2019) that unlocked the path to diffusion models.

Let $\{\sigma_i\}_{i=1}^{L}$ denote the sequence of L noise levels that satisfies $\frac{\sigma_1}{\sigma_2} = \ldots = \frac{\sigma_{L-1}}{\sigma_L} > 1$. For each noise level, we define

$$p_{\text{data}}^{\sigma_i}(x) = \int p_{\text{data}}(x') K_{\sigma_i}(x|x') \mathrm{d}x',$$

The pedestrian approach is to train a model for each noise level, but this is computationally expensive.



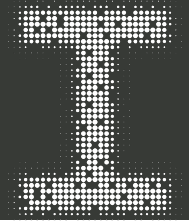
This is the idea in Song and Ermon (2019) that unlocked the path to diffusion models.

Let $\{\sigma_i\}_{i=1}^{L}$ denote the sequence of L noise levels that satisfies $\frac{\sigma_1}{\sigma_2} = \ldots = \frac{\sigma_{L-1}}{\sigma_L} > 1$. For each noise level, we define

$$p_{\text{data}}^{\sigma_i}(x) = \int p_{\text{data}}(x') K_{\sigma_i}(x|x') \mathrm{d}x',$$

The pedestrian approach is to train a model for each noise level, but this is computationally expensive.

We will instead choose a single network for all noise levels: Noise conditional score networks (NCSN).



Let us denote the DSM loss for a single noise level as

$$\ell(\theta, \sigma) = \mathrm{F}(p^{\sigma}_{\text{data}} || \pi_\theta) = \mathbb{E}_{p^{\sigma}_{\text{data}}} \left[\frac{1}{2} \| \nabla \log p^{\sigma}_{\text{data}}(X) - \mathrm{s}_\theta(X) \|^2 \right].$$

Let us denote the DSM loss for a single noise level as

$$\ell(\theta, \sigma) = \mathrm{F}(p^{\sigma}_{\text{data}} || \pi_\theta) = \mathbb{E}_{p^{\sigma}_{\text{data}}} \left[\frac{1}{2} \| \nabla \log p^{\sigma}_{\text{data}}(X) - \mathrm{s}_\theta(X) \|^2 \right].$$

Let us clarify this for the specific kernel K. Recall that we can write

$$\ell(\theta, \sigma) = \mathbb{E}_{p(x,x')} \left[\frac{1}{2} \| \nabla_x \log K_\sigma(X|X') - \mathrm{s}_\theta(X, \sigma) \|^2 \right].$$

where $p(x, x') = p_{\text{data}}(x') K_\sigma(x|x')$.

The final loss is then

$$\ell(\theta, \sigma) = \mathbb{E}_{p(x,x')} \left[\frac{1}{2} \left\| \frac{X - X'}{\sigma^2} + \mathrm{s}_\theta(X, \sigma) \right\|^2 \right].$$

It is typical to build the final loss as

$$\ell(\theta) = \frac{1}{L} \sum_{i=1}^{L} \lambda(\sigma_i) \ell(\theta, \sigma_i),$$

where $\lambda(\sigma_i)$ is a weighting function. Song and Ermon (2019) propose $\lambda(\sigma) = \sigma^2$. Therefore, the training phase of the algorithm is to find

$$\theta_\star \in \operatorname*{argmin}_\theta \ell(\theta).$$

Algorithm Pseudocode for sampling from EBMs using annealed Langevin dynamics

Input: The trained NCSN $s_{\theta_\star}(x, \sigma)$, the number of noise levels L, the step size ε, the number of iterations T
$X_0 =$ random initialisation.
for $i = 1, \ldots, L$ **do**
    $\gamma_i = \varepsilon \frac{\sigma_i^2}{\sigma_L^2}$.
    for $t = 1, \ldots, T$ **do**
        $X_{t+1} = X_t + \gamma_i s_{\theta_\star}(X_t, \sigma_i) + \sqrt{2\gamma_i} W_{t+1}$.
    end for
    $X_0 = X_T$.
end for

Energy Based Models

Denoising Score Matching with Multiple Levels: Towards diffusion models

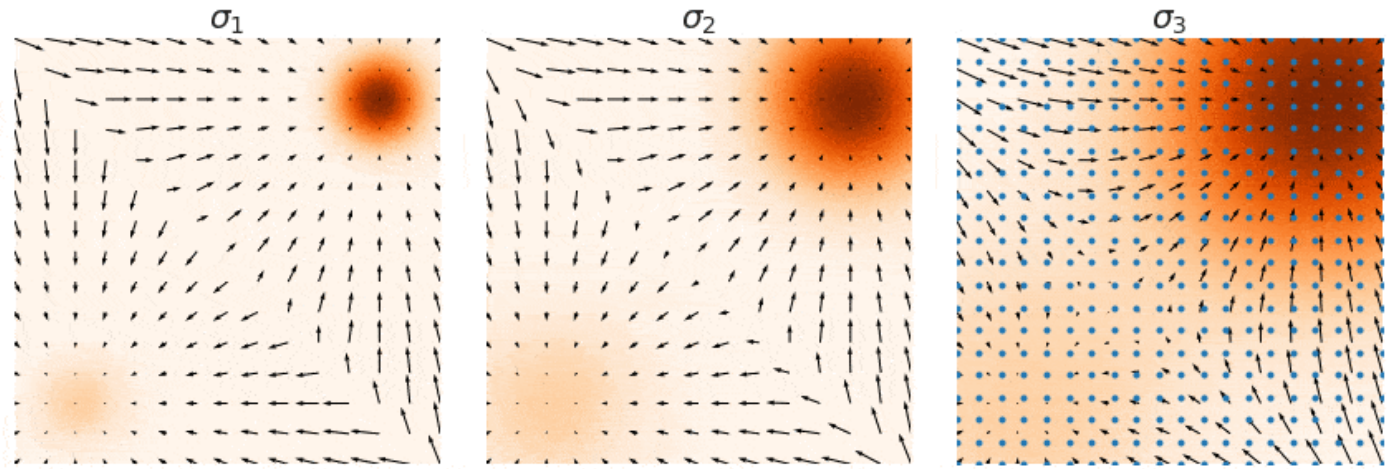


Figure from: `https://yang-song.net/blog/2021/score/`

Energy Based Models

Denoising Score Matching with Multiple Levels: Towards diffusion models

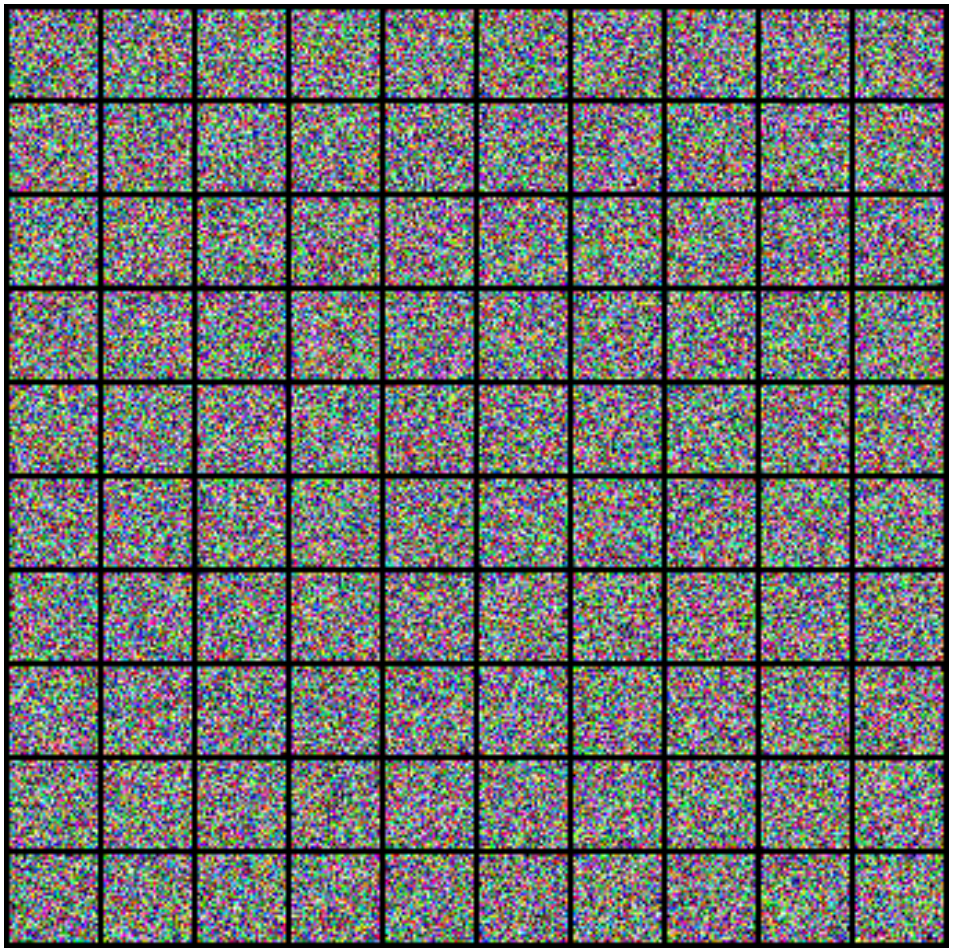


Figure from: `https://yang-song.net/blog/2021/score/`

Taking this idea to its limit, Song et al., 2020 proposed to use forward-reverse SDEs of the form:

$$\mathrm{d}\mathbf{X}_t = -\frac{1}{2}\beta(t)\mathbf{X}_t\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\mathbf{W}_t, \quad \mathbf{X}_0 \sim p_0 = p_{\text{data}}. \tag{7}$$

and

$$\mathrm{d}\mathbf{X}_t = \frac{1}{2}\beta(t)\mathbf{X}_t\mathrm{d}t + \beta(t)\nabla \log p_t(\mathbf{X}_t)\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\bar{\mathbf{W}}_t, \tag{8}$$

where $\mathbf{X}_T \sim p_T$. Here $\nabla \log p_t$ is **intractable** and approximated via score matching and $p_t(x) = \int p_{t|0}(x|x_0)p_{\text{data}}(x_0)\mathrm{d}x_0$.

Diffusion Models

Score-based models

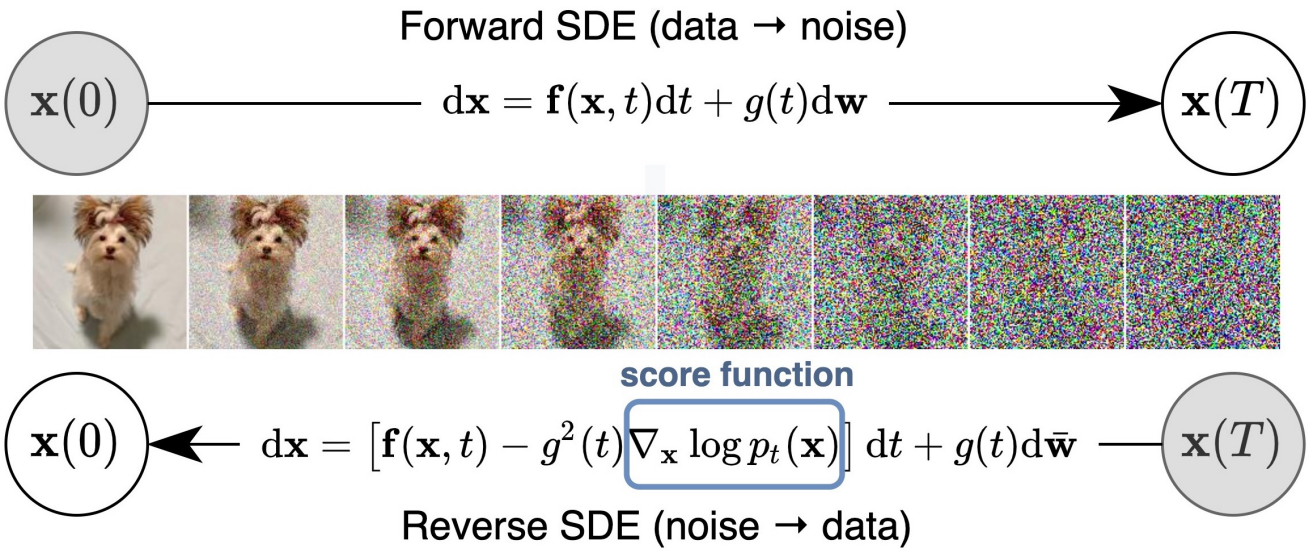


Image credit: `https://yang-song.net/blog/2021/score/`

With a similar score matching idea, the score function can be learned via the minimization

$$\theta^\star \in \arg\min \mathbb{E}_{t\in\text{Unif}[0,1]}\mathbb{E}_{x_0\sim p_{\text{data}}}\mathbb{E}_{x\sim p_{t|0}(x|x_0)}[\|\nabla \log p_{t|0}(x|x_0) - s_\theta(x,t)\|^2],$$

Finally with the approximation

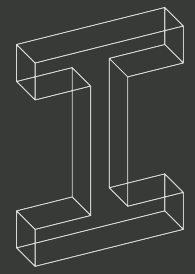
$$\mathrm{s}_{\theta^\star}(x,t) \approx \nabla \log p_t(x),$$

we can implement SGMs by discretizing the reverse process.

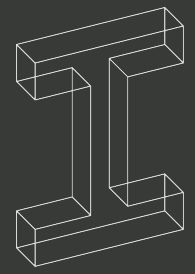
To be concrete, after training, we would use

$$\mathrm{d}\mathbf{X}_t = \frac{1}{2}\beta(t)\mathbf{X}_t\mathrm{d}t + \beta(t)s_{\theta^\star}(x,t)\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\bar{\mathbf{W}}_t, \tag{9}$$

and its discretizations for generation.

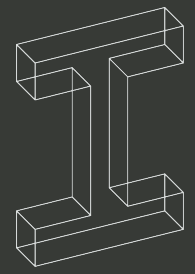


Diffusion models took the machine learning community by storm by their impressive sample quality.



Diffusion models took the machine learning community by storm by their impressive sample quality.

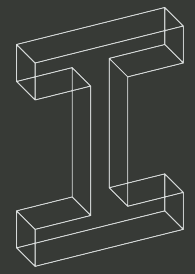
- Outperforms earlier generative models (such as GANs, VAEs) in terms of sample quality.



Diffusion models took the machine learning community by storm by their impressive sample quality.

- Outperforms earlier generative models (such as GANs, VAEs) in terms of sample quality.

A big interest in the computational statistics community is to use these models for inverse problems.



Diffusion models took the machine learning community by storm by their impressive sample quality.

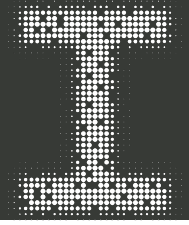
- Outperforms earlier generative models (such as GANs, VAEs) in terms of sample quality.

A big interest in the computational statistics community is to use these models for inverse problems.

Next: Diffusion models for inverse problems.

Diffusion Models

for inverse problems



The setup:

- Assume we have a *trained* model $s_{\theta^\star}(x, t)$ for sampling from p_{data}.

Diffusion Models

for inverse problems

The setup:

- Assume we have a *trained* model $s_{\theta^\star}(x, t)$ for sampling from p_{data}.
- Given a likelihood $p(y|x)$, we would like to sample from the posterior $p(x|y) \propto p(y|x)p_{\text{data}}(x)$.

Diffusion Models

for inverse problems

The setup:

- Assume we have a *trained* model $s_{\theta^\star}(x, t)$ for sampling from p_{data}.
- Given a likelihood $p(y|x)$, we would like to sample from the posterior $p(x|y) \propto p(y|x)p_{\text{data}}(x)$.

If we wanted to use an SGM for the posterior, we would need at time t

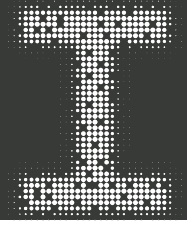
$$\begin{aligned}\nabla \log p_t(x_t|y) &= \nabla \log p_t(x_t) + \nabla \log p_{y|t}(y|x_t), \\ &\approx s_{\theta^\star}(x_t, t) + \nabla \log p_{y|t}(y|x_t).\end{aligned}$$

where

$$p_{y|t}(y|x_t) = \int p_{y|0}(y|x_0)p(x_0|x_t)\mathrm{d}x_0.$$

Diffusion Models

for inverse problems



The setup:

- Assume we have a *trained* model $s_{\theta^\star}(x, t)$ for sampling from p_{data}.
- Given a likelihood $p(y|x)$, we would like to sample from the posterior $p(x|y) \propto p(y|x)p_{\text{data}}(x)$.

If we wanted to use an SGM for the posterior, we would need at time t

$$\begin{aligned}\nabla \log p_t(x_t|y) &= \nabla \log p_t(x_t) + \nabla \log p_{y|t}(y|x_t), \\ &\approx s_{\theta^\star}(x_t, t) + \nabla \log p_{y|t}(y|x_t).\end{aligned}$$

where

$$p_{y|t}(y|x_t) = \int p_{y|0}(y|x_0)p(x_0|x_t)\mathrm{d}x_0.$$

In other words, if we approximate $\nabla \log p_{y|t}(y|x)$, then we can use *pre-trained* models as priors.

Let us simplify even more, and assume that we would like to recover $p(x_0|y)$ where

$$y = Hx_0 + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_y^2 I_y).$$

i.e., we have a linear inverse problem with Gaussian noise:

$$p(y|x_0) = \mathcal{N}(y; Hx_0, \sigma_y^2 I_y).$$

Recall that we are after approximating

$$p_{y|t}(y|x_t) = \int p(y|x_0)p(x_0|x_t)\mathrm{d}x_0.$$

Given that $p(y|x_0)$ is Gaussian, we **could** compute the integral analytically if $p(x_0|x_t)$ were Gaussian.

One solution is to use Tweedie's formula.

Proposition 3 (Tweedie's formula)

Let $\mathbf{m}_{0|t}$ and $\mathbf{C}_{0|t}$ be the mean and the covariance of $p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)$, respectively. Then given the marginal density $p_t(\mathbf{x}_t)$, the mean is given as

$$\mathbf{m}_{0|t} = \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t] = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t + v_t \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)).$$

Then the covariance $\mathbf{C}_{0|t}$ is given by

$$\begin{aligned}\mathbf{C}_{0|t} &= \mathbb{E}\left[(\mathbf{x}_0 - \mathbf{m}_{0|t})(\mathbf{x}_0 - \mathbf{m}_{0|t})^\top \mid \mathbf{x}_t\right] \\ &= \frac{v_t}{\alpha_t}(\mathbf{I}_{d_x} + v_t \nabla^2 \log p_t(\mathbf{x}_t)) = \frac{v_t}{\sqrt{\alpha_t}} \nabla_{\mathbf{x}_t} \mathbf{m}_{0|t}.\end{aligned}$$

Proposition 4 (Moment projection)

Let $p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)$ be a distribution with mean $\mathbf{m}_{0|t}$ and covariance $\mathbf{C}_{0|t}$. Let $\hat{p}_{0|t}(\mathbf{x}_0|\mathbf{x}_t)$ be the the closest Gaussian in KL divergence to $p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)$, i.e.,

$$\hat{p}_{0|t}(\mathbf{x}_0|\mathbf{x}_t) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)||q),$$

where $\mathcal{Q}$ is the family of multivariate Gaussian distributions. Then

$$\hat{p}_{0|t}(\mathbf{x}_0|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_0; \mathbf{m}_{0|t}, \mathbf{C}_{0|t}).$$

This is a well-known moment matching result, see, e.g., Bishop, 2006.

Merging Tweedie and moment projection propositions leads to *Tweedie moment projection*:

$$p_{0|t}(\mathbf{x}_0|\mathbf{x}_t) \approx \mathcal{N}\left(\mathbf{x}_0; \mathbf{m}_{0|t}, \frac{v_t}{\sqrt{\alpha_t}} \nabla_{\mathbf{x}_t} \mathbf{m}_{0|t}\right), \tag{10}$$

Finally using Tweedie moment projection, we can construct the approximation of the gradient of the "smoothed" likelihood:

$$\begin{aligned} p_{\mathbf{y}|t}(\mathbf{y}|\mathbf{x}_t) &= \int p_{\mathbf{y}|0}(\mathbf{y}|\mathbf{x}_0) p_{0|t}(\mathbf{x}_0|\mathbf{x}_t) \mathrm{d}\mathbf{x}_t \\ &\approx \mathcal{N}\left(\mathbf{y}; \mathbf{H}\mathbf{m}_{0|t}, \mathbf{H}\frac{\nu_t}{\sqrt{\alpha_t}}\nabla_{\mathbf{x}_t}\mathbf{m}_{0|t}\mathbf{H}^\top + \sigma_y^2 \mathbf{I}_{d_y}\right) \end{aligned}$$

which leads to the approximation

$$\begin{aligned} f^{\mathbf{y}}(\mathbf{x}_t) &:= \nabla_{\mathbf{x}_t}\mathbf{m}_{0|t}\mathbf{H}^\top (\mathbf{H}\frac{\nu_t}{\sqrt{\alpha_t}}\nabla_{\mathbf{x}_t}\mathbf{m}_{0|t}\mathbf{H}^\top + \sigma_y^2 \mathbf{I}_{d_y})^{-1}(\mathbf{y} - \mathbf{H}\mathbf{m}_{0|t}) \\ &\approx \nabla_{\mathbf{x}_t} \log p_{\mathbf{y}|t}(\mathbf{y}|\mathbf{x}_t), \end{aligned}$$

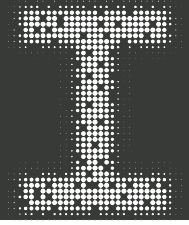
Finally using Tweedie moment projection, we can construct the approximation of the gradient of the "smoothed" likelihood:

$$
\begin{aligned}
p_{\mathbf{y}|t}(\mathbf{y}|\mathbf{x}_t) &= \int p_{\mathbf{y}|0}(\mathbf{y}|\mathbf{x}_0) p_{0|t}(\mathbf{x}_0|\mathbf{x}_t) \mathrm{d}\mathbf{x}_t \\
&\approx \mathcal{N}\left(\mathbf{y}; \mathbf{H}\mathbf{m}_{0|t}, \mathbf{H}\frac{\nu_t}{\sqrt{\alpha_t}}\nabla_{\mathbf{x}_t}\mathbf{m}_{0|t}\mathbf{H}^\top + \sigma_y^2 \mathbf{I}_{d_y}\right)
\end{aligned}
$$

which leads to the approximation

$$
\begin{aligned}
f^{\mathbf{y}}(\mathbf{x}_t) &:= \nabla_{\mathbf{x}_t}\mathbf{m}_{0|t}\mathbf{H}^\top(\mathbf{H}\frac{\nu_t}{\sqrt{\alpha_t}}\nabla_{\mathbf{x}_t}\mathbf{m}_{0|t}\mathbf{H}^\top + \sigma_y^2\mathbf{I}_{d_y})^{-1}(\mathbf{y} - \mathbf{H}\mathbf{m}_{0|t}) \\
&\approx \nabla_{\mathbf{x}_t} \log p_{\mathbf{y}|t}(\mathbf{y}|\mathbf{x}_t),
\end{aligned}
$$

Many approximations... This is because things are hard otherwise.



Note that this framework subsumes the previous ones:

- Denoising posterior sampling (Chung et al., 2022)

$$\mathbf{m}_{0|t}^{\text{DPS-D}} = \mathbf{m}_{0|t} \qquad \text{and} \qquad \mathbf{C}_{0|t}^{\text{DPS-D}} = 0.$$

- Pseudo-inverse Guided Diffusion (Song et al., 2023)

$$\mathbf{m}_{0|t}^{\Pi\text{G}} = \mathbf{m}_{0|t} \qquad \text{and} \qquad \mathbf{C}_{0|t}^{\Pi\text{G}} = r_t^2 \mathbf{I}_{d_x}$$

Diffusion Models

Noisy super-resolution

$\mathbf{H} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ project to low dimensional subspace (low resolution)

$\mathbf{y} = \mathbf{H}\mathbf{x} + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_{\mathbf{y}}^2 \mathbf{I})$ corrupt with Gaussian noise

Measurement, $\mathbf{y}$ is a corrupted image from the FFHQ validation set.

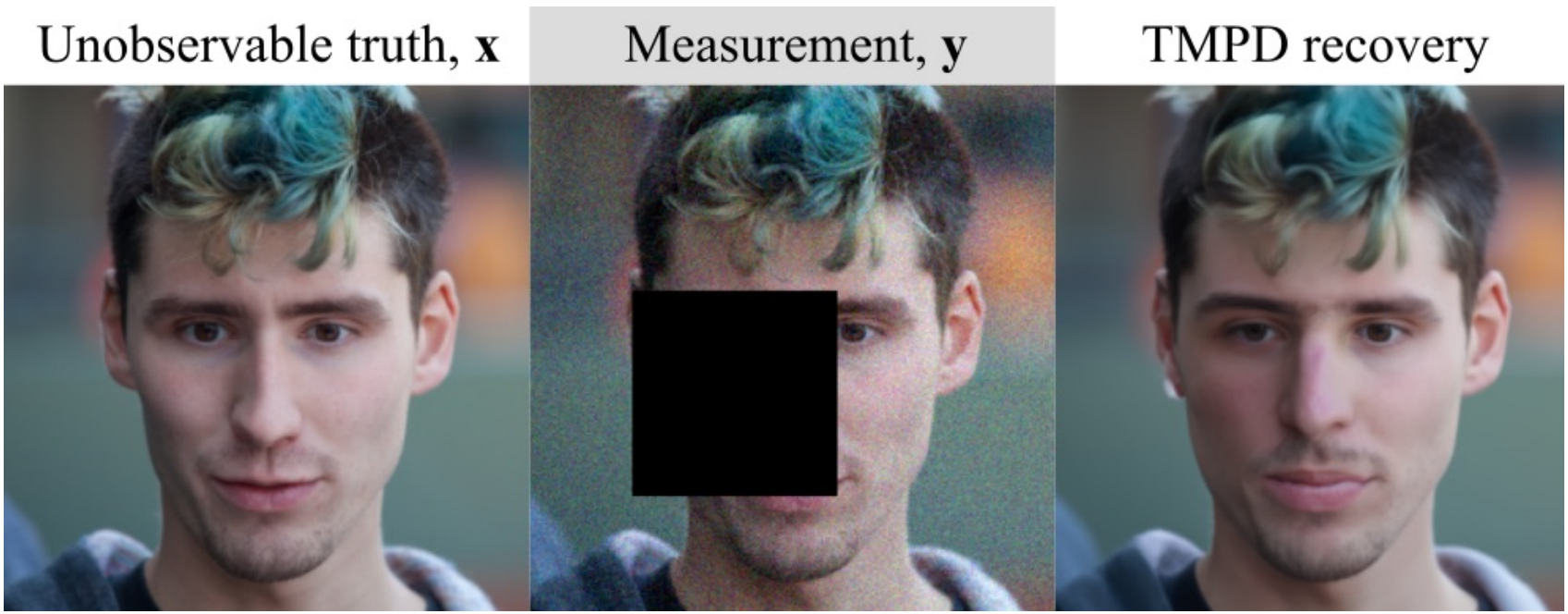
Diffusion Models

Noisy Inpainting

$\mathbf{H} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ project to low dimensional subspace (e.g., 'box' mask)

$\mathbf{y} = \mathbf{H}\mathbf{x} + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_{\mathbf{y}}^2 \mathbf{I})$ corrupt with Gaussian noise

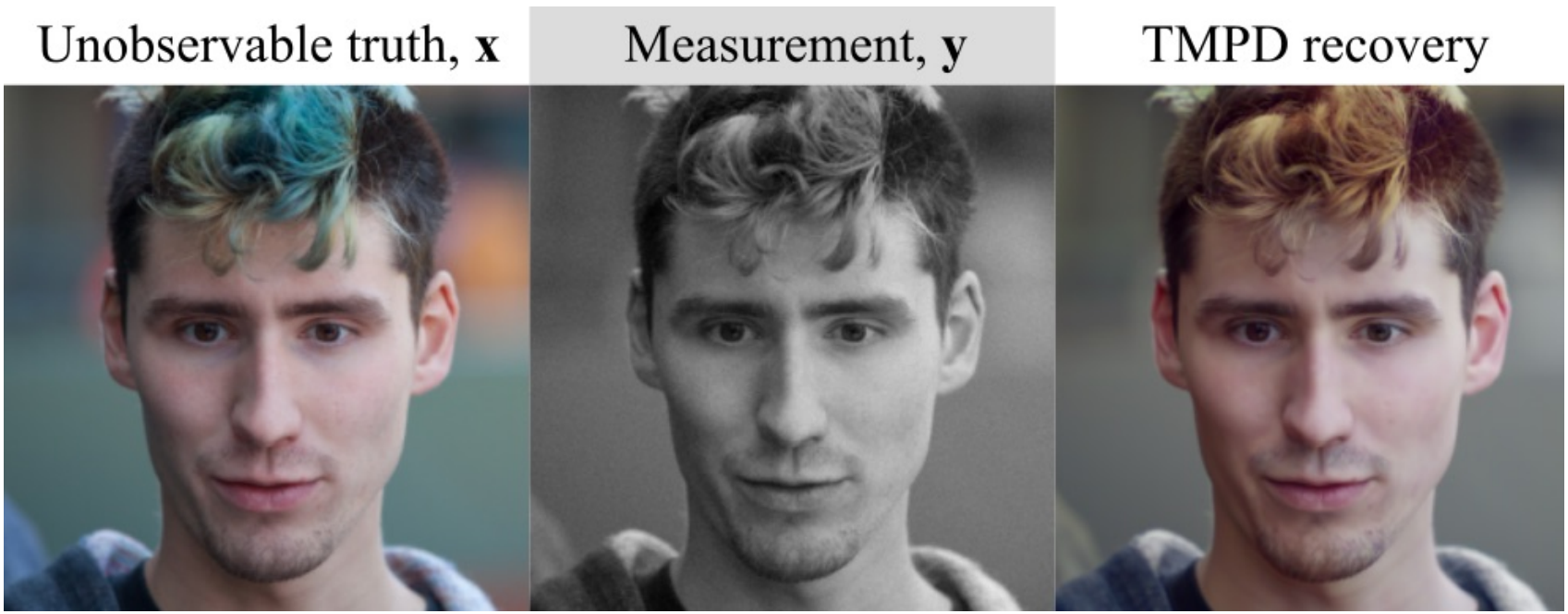
Measurement, $\mathbf{y}$ is a corrupted image from the FFHQ validation set.



$\mathbf{H} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x/3}$ project to low dim. subspace (from RGB to gray-scale)

$\mathbf{y} = \mathbf{H}\mathbf{x} + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_{\mathbf{y}}^2 \mathbf{I})$ corrupt with Gaussian noise

Measurement, $\mathbf{y}$ is a corrupted image from the FFHQ validation set.



References I

- Atchadé, Yves, Keer Jiang, and Yi Sun (2023). *On Generative Energy-Based Models*.
- Bishop, Christopher M (2006). *Pattern Recognition and Machine Learning*. Springer.
- Boys, Benjamin, Mark Girolami, Jakiw Pidstrigach, Sebastian Reich, Alan Mosca, and O Deniz Akyildiz (2024). "Tweedie moment projected diffusions for inverse problems". In: *Transactions of Machine Learning Research (TMLR)*.
- Chung, Hyungjin, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye (2022). "Diffusion Posterior Sampling for General Noisy Inverse Problems". In: *The Eleventh International Conference on Learning Representations*.

References II

- Durmus, Alain and Eric Moulines (2019). "High-dimensional Bayesian inference via the unadjusted Langevin algorithm". In: *Bernoulli* 25.4A, pp. 2854–2882.
- Hwang, Chii-Ruey (1980). "Laplace's method revisited: weak convergence of probability measures". In: *The Annals of Probability*, pp. 1177–1182.
- Hyvärinen, Aapo (2005). "Estimation of non-normalized statistical models by score matching". In: *Journal of Machine Learning Research* 6.4.
- Murphy, Kevin P (2023). *Probabilistic machine learning: Advanced topics*. MIT press.

References III

- Nijkamp, Erik, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu (2019). "Learning non-convergent non-persistent short-run mcmc toward energy-based model". In: *Advances in Neural Information Processing Systems* 32.
- Song, Jiaming, Arash Vahdat, Morteza Mardani, and Jan Kautz (2023). "Pseudoinverse-guided diffusion models for inverse problems". In: *International Conference on Learning Representations*.
- Song, Yang and Stefano Ermon (2019). "Generative modeling by estimating gradients of the data distribution". In: *Advances in neural information processing systems* 32.
- Song, Yang and Diederik P Kingma (2021). "How to train your energy-based models". In: *arXiv preprint arXiv:2101.03288*.

References IV

- Song, Yang, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole (2020). "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations*.
- Vincent, Pascal (2011). "A connection between score matching and denoising autoencoders". In: *Neural computation* 23.7, pp. 1661–1674.
- Welling, Max and Yee W Teh (2011). "Bayesian learning via stochastic gradient Langevin dynamics". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer, pp. 681–688.