Recent Developments in Algorithm Design: Sampling from High-Dimensional Distributions

Prof. Christopher Musco, New York University

COMMUNICATION-FREE COUPLING

SPECULATIVE DECODING

Draft: NYU is a private research university in the city of New York .

Desired Output: NYU is a private research university in New York City.

NYU is a private research **university**

NYU is a private research university **in**

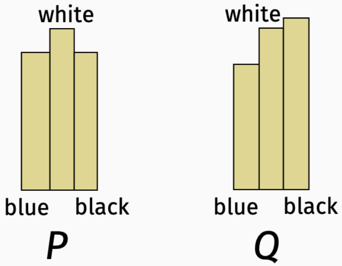
NYU is a private research university in **New**

NYU is a private research university in the **city**

....

SPECULATIVE DECODING

Issue: Even if then next token distribution for the drafter model, $\mathcal{P}$, and the product model, $\mathcal{Q}$ are very similar, it could be unlikely for the draft to be correct.



If $a \sim P$ and $b \sim Q$, $\Pr[a = b] \approx$

COUPLING

Solution: Coordinate the sampling!

Definition (Coupling)

Let $\mathcal{P}$ and $\mathcal{Q}$ be distributions over $\{1, \ldots, n\}$. A coupling between $\mathcal{P}$ and $\mathcal{Q}$ is any distribution over pairs $(a, b) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$ such that a's marginal distribution is $\mathcal{P}$ and b's marginal distribution is $\mathcal{Q}$.

Goal: Efficiently sample from a coupling $\mathcal{C}$ between the small and large model distributions which maximizes

$$\Pr[a = b].$$

Always possible to find a coupling which ensures that $\Pr[a = b] = 1 - D_{TV}(\mathcal{P}, \mathcal{Q})$.

TOTAL VARIANCE DISTANCE

Reminder: For discrete distributions $\mathcal{P}$ and $\mathcal{Q}$ over $\{1, \ldots, n\}$ represented by length n probability vectors $\mathbf{p}, \mathbf{q} \in [0,1]^n$,

$$D_{TV}(\mathcal{P}, \mathcal{Q}) = 1 - \sum_{i=1}^{n} \min(p_i, q_i).$$

SPECULATIVE DECODING COUPLING

The following procedure achieves the optimal bound of $\Pr[a = b] = 1 - D_{TV}(\mathcal{P}, \mathcal{Q})$.

Drafter:

- Sample $a \sim \mathcal{P}$. Sends both a and $\mathbf{p}$ to FullModel.

Full Model:

- Await $(a, \mathbf{p})$ from Drafter.
- With probability $\min(1, q_a/p_a)$ return $b = a$.
- Otherwise, sample b from $\mathcal{Q}' = \{q'_1, \ldots, q'_n\}$, where:

$$q'_i = \frac{\max(0, q_i - p_i)}{\sum_{i=1}^{n} \max(0, q_j - p_j)}$$

Important that the Drafter could sample without knowing the Full Model's Distribution, q! There is only "one-way communication".

Is it posssible to do anything with no communication between the samplers?

Why would we care? The output of the Full Model is always sampled from $\mathcal{Q}$, but the exact value sampled depends on the Drafter distribution $\mathcal{P}$.

- Cannot immediately verify that adding speculative decoding did not change the model distribution.
- If drafter changes, model output is not deterministic from the user's point of view given a fixed random seed.

DRAFTER-INVARIANT SPECULATIVE DECODING

"Coupling without Communication and Drafter-Invariant Speculative Decoding" [**Daliri**, Musco, Suresh, ISIT 2025].

Basically the same idea appeared in:

- Anari, Gao, Rubinstein, STOC 2024
- Liu, Yin, STOC 2022
- Bavarian, Ghazi, Haramaty, Kamath, Rivest, Sudan, 2020.

WEIGHTED MINHASH COUPLING

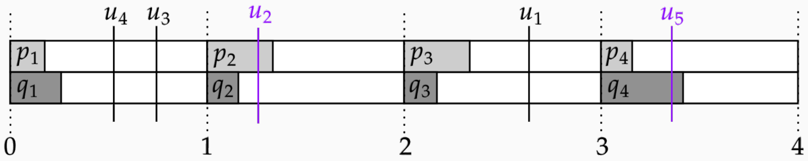
Fix public random variables $u_1, u_2, \ldots \sim \mathrm{Unif}[0, n]$.

Drafter:

- For $k = 1, 2, \ldots,$
    - If $k \in [j-1, j-1+p_j]$ for some j, return $a = j$.

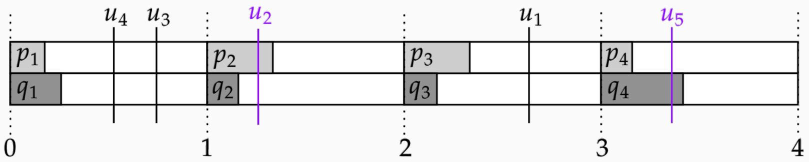
Full Model:

- For $k = 1, 2, \ldots,$
    - If $k \in [j-1, j-1+q_j]$ for some j, return $b = j$.



WEIGHTED MINHASH COUPLING

Claim: $\Pr[a = b] \geq \frac{\sum_{i=1}^{n} \min(p_i, q_i)}{\sum_{i=1}^{n} \max(p_i, q_i)}$



COMMUNICATION FREE COUPLING

Optimal Coupling:

$$\Pr[a = b] = 1 - D_{TV}(\mathcal{P}, \mathcal{Q})$$

. Communication-Free Coupling:

$$\Pr[a = b] \geq \frac{\sum_{i=1}^{n} \min(p_i, q_i)}{\sum_{i=1}^{n} \max(p_i, q_i)} = \frac{1 - D_{TV}(\mathcal{P}, \mathcal{Q})}{1 + D_{TV}(\mathcal{P}, \mathcal{Q})}$$

.

Takeaway: Pay very little for no communication!

Possible to show that this is optimal. No communication-free protocol can achieve for all distributions:

$$\Pr[a = b] > \frac{1 - D_{TV}(\mathcal{P}, \mathcal{Q})}{1 + D_{TV}(\mathcal{P}, \mathcal{Q})}.$$

[Bavarian, Ghazi, Haramaty, Kamath, Rivest, Sudan, 2020].

GUMBEL SAMPLING

Fix public random variables $u_1, u_2, \ldots \sim \text{Unif}[0,1]$.

Drafter:

- Return $a = \arg\min_{i \in \{1,\ldots,n\}} \frac{-\ln(u_i)}{p_i}$.

Full Model:

- Return $b = \arg\min_{i \in \{1,\ldots,n\}} \frac{-\ln(u_i)}{q_i}$.

This is already how samples are typically obtained! In particular, standard to use the "Gumbel Max Trick":

$$b = \arg\max_{i \in \{1,\ldots,n\}} \left[\ln(q_i) - \ln(\ln(1/u_i))\right].$$

Not too hard to check that $a \sim \mathcal{P}$ and $b \sim \mathcal{Q}$.

GUMBEL SAMPLING

Gumbel sampling gives a pareto improvement over weighted MinHash.

Theorem (Daliri, Musco, Suresh, ISIT 2025)

For any two distributions $\mathcal{P}, \mathcal{Q}$,

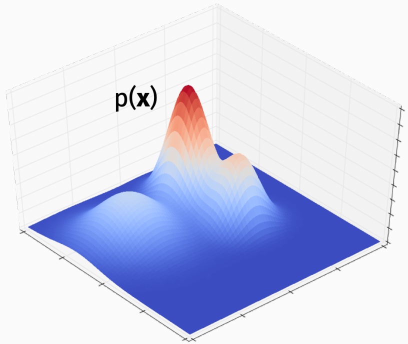
$$\Pr_{(a,b)\sim Gumbel}[a = b] \geq \Pr_{(a,b)\sim MinHash}[a = b],$$

and there exist distributions where inequality is strict.

Question one group is studying for the project: Is Gumbel pareto optimal?

EFFICIENT SAMPLING IN HIGH-DIMENSIONS

Increasingly common goal in machine learning: Sample from a distribution over $\mathbb{R}^d$ with density $p(\mathbf{x})$



Assume $p(\mathbf{x}) \propto \exp(-f(\mathbf{x}))$ for some function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ and that we are given gradient oracle access to $\nabla f(\mathbf{x})$.

EFFICIENT SAMPLING IN HIGH-DIMENSIONS

What I hope to cover:

- Where/why does this problem arise in machine learning?
- What is the (stochastic) gradient Langevin dynamics algorithm and why does is work?
- Where is the area headed / where are opportunities for algorithms research?

GRADIENT REMINDER

Recall that $\nabla f : \mathbb{R}^d \to \mathbb{R}^d$ returns the vector of partial derivatives at a point $\mathbf{x}$:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_d} f(\mathbf{x}) \end{bmatrix}.$$

The gradient determines the instantanious change in f's value with respect to changes in the input variables:

$$\lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{v}) - f(\mathbf{x})}{h} = \langle \nabla f(\mathbf{x}), \mathbf{v} \rangle.$$

LOSS MINIMIZATION

Where do gradients show up in machine learning?

Let $M_{\mathbf{x}} : \mathbb{R}^d \to \mathbb{R}$ be a model parameterized by $\mathbf{x}$. Given a labeled dataset $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_n, b_n)$, goal in supervised learning is to find parameters such that:

$$M_{\mathbf{x}}(\mathbf{a}_i) \approx b_i.$$

Typically accomplished by writing down some loss function $f(\boldsymbol{\theta})$ and minimizing. For example, least squares loss:

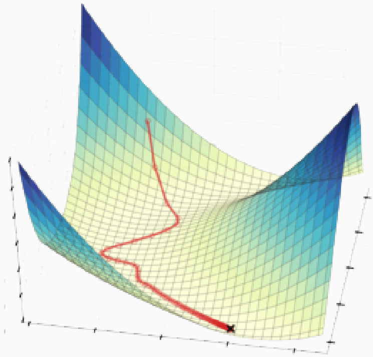
$$f(\mathbf{x}) = \sum_{i=1}^{n} (M_{\mathbf{x}}(\mathbf{a}_i) - b_i)^2.$$

Goal: Find $\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x})$.

GRADIENT DESCENT

Most common algorithm to do so: **gradient descent**.

- Choose starting point $\mathbf{x}_0 \in \mathbb{R}^d$, step size η.
- For $t = 0, \ldots, T$
  - $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta \cdot \nabla f(\mathbf{x}_t)$.



GRADIENT DESCENT

Gradient descent:

- Choose starting point $\mathbf{x}_0 \in \mathbb{R}^d$, step size η.
- For $t = 0, \ldots, T$
  - $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta \cdot \nabla f(\mathbf{x}_t)$.

Justification: We want to make a small change, $\eta \cdot \mathbf{v}$ to $\mathbf{x}_t$ that decreases the value of f.

$$f(\mathbf{x} + \eta \cdot \mathbf{v}) - f(\mathbf{x}) \approx \eta \cdot \langle \nabla f(\mathbf{x}), \mathbf{v} \rangle.$$

Choosing $\mathbf{v} = -\nabla f(\mathbf{x})$ ensures that, if we take $\eta \to 0$,

$$f(\mathbf{x} + \eta \cdot f(\mathbf{v})) - f(\mathbf{x}) < 0$$

WHY GRADIENT DESCENT

- Simple and general. We only need to implement a gradient oracle for computing $\nabla f(\mathbf{x})$. For almost all models with d parameters, can be done in $O(nd)$ time.
- Stochastic approximation of gradient is even faster. Typically $O(d)$ time:

$$\nabla f(\mathbf{x}) = \sum_{i=1}^{n} \nabla \ell(\mathbf{x}, \mathbf{a}_i, b_i).$$

- Guaranteed to converge to a stationary point (e.g., local min) of f for sufficiently small step size.
- Dimension independent convergence rates can be obtained under mild assumptions.

EXAMPLE CONVERGENCE BOUND

f is β-smooth if for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq \beta \|\mathbf{x} - \mathbf{y}\|_2$.

Theorem (Convergence to Stationary Point)

For any β-smooth, differentiable function f, if we run GD for T steps, we can find a point $\mathbf{x}_T$ such that:

$$\|\nabla f(\mathbf{x}_T)\|_2^2 \leq \frac{2\beta}{T}\left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right)$$

Corollary: If f is convex and $\|\mathbf{x}_0 - \mathbf{x}^*\|_2 = R$, then after $T = O\left(\frac{\beta R^2}{\epsilon}\right)$ steps[1] we have $f(\mathbf{x}_T) - f(\mathbf{x}^*) \leq \epsilon$.

[1] Other methods (e.g., Center-of-Gravity Method) can achieve a better dependence on ϵ, but at the cost of a dependence on d.

ALGORITHMIC WORK AROUND GRADIENT DESCENT

Huge amount of algorithmic research centered around gradient descent and its variants.

- Acceleration/momentum to speed up convergence.
- Generalized steppest descent, mirror descent, etc.
- Stochastic gradient methods, variance reduction.
- Preconditioning, quasi-second order methods, adaptive step size methods.
- Lower bounds (e.g, in first order oracle model).

Where does least squares loss comes from?

Assume fixed dataset $\mathbf{a}_1, \ldots, \mathbf{a}_n$ with targets generated from ground truth model, $M_{\mathbf{x}}$, plus Gaussian noise:

$$\begin{aligned} b_1 &= M_{\mathbf{x}}(\mathbf{a}_1) + \epsilon_1, \\ &\vdots \\ b_n &= M_{\mathbf{x}}(\mathbf{a}_n) + \epsilon_n, \end{aligned}$$

where $\epsilon_1, \ldots, \epsilon_n \sim \mathcal{N}(0, \sigma^2)$.

Would like to choose params. <u>most likely</u> to have generated the targets we observed. Likelihood of data given parameters:

$$L(\mathbf{x}) = p(b_1, \ldots, b_n \mid \mathbf{x}) \propto \prod_{i=1}^{n} \exp\left(-\frac{(b_i - M_{\mathbf{x}}(\mathbf{a}_i))^2}{2\sigma^2}\right).$$

Goal: Compute the maximium likelihood estimator (MLE):

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} L(\mathbf{x}).$$

Equivalent to minimizing the negative log-likelihood:

$$f(\mathbf{x}) = -\log L(\mathbf{x}) = \frac{1}{2\sigma^2}\sum_{i=1}^{n}(b_i - M_{\mathbf{x}}(\mathbf{a}_i))^2 + \text{const}.$$

Most standard ML loss functions are negative log-likelihoods for some other data generation process, including logistic/cross-entropy loss, ℓ_1 loss, etc.

BAYESIAN MACHINE LEARNING

One step further: Assume prior distribution over parameters $\mathbf{x}$. E.g. $x_i \sim \mathcal{N}(0, \gamma^2)$ for all i.

Lets us define a posterior probability of $\mathbf{x}$ given the data:

$$p(\mathbf{x} \mid b_1, \ldots, b_n) = \frac{p(b_1, \ldots, b_n \mid \mathbf{x}) \cdot p(\mathbf{x})}{p(b_1, \ldots, b_n)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}.$$

Goal: Compute the maximum a posteriori (MAP) estimator:

$$\begin{aligned}\mathbf{x}^* &= \arg\max_{\mathbf{x}} p(\mathbf{x} \mid b_1, \ldots, b_n) \\ &= \arg\max_{\mathbf{x}} p(b_1, \ldots, b_n \mid \mathbf{x}) \cdot p(\mathbf{x})\end{aligned}$$

Again, can equivalently minimize the negative log-posterior:

$$f(\mathbf{x}) = -\log(p(b_1, \ldots, b_n \mid \mathbf{x})) - \log(p(\mathbf{x})).$$

Example: Least squares loss with Gaussian prior.

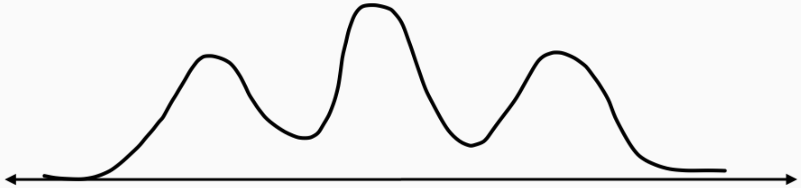
$$p(b_1, \ldots, b_n \mid \mathbf{x}) \propto \prod_{i=1}^{n} \exp\left(-\frac{(b_i - M_{\mathbf{x}}(\mathbf{a}_i))^2}{2\sigma^2}\right)$$

$$p(\mathbf{x}) \propto \prod_{i=1}^{d} \exp\left(-\frac{x_i^2}{2\gamma^2}\right)$$

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{2\sigma^2}\sum_{i=1}^{n}(b_i - M_{\mathbf{x}}(\mathbf{a}_i))^2 + \frac{1}{2\gamma^2}\sum_{i=1}^{d} x_i^2 \\ &= \frac{1}{2\sigma^2}\sum_{i=1}^{n}(b_i - M_{\mathbf{x}}(\mathbf{a}_i))^2 + \frac{1}{2\gamma^2}\|\mathbf{x}\|_2^2. \end{aligned}$$

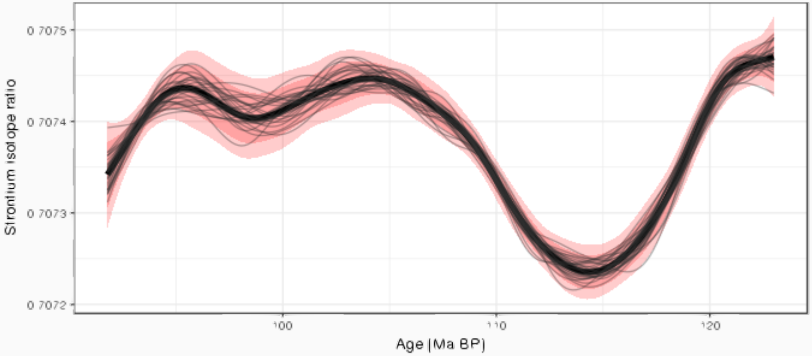
BAYESIAN MACHINE LEARNING

Optimization (usually solved with gradient descent) computes the mode of the posterior distribution $p(\mathbf{x} \mid b_1, \ldots, b_n)$.



Another important goal: Sample parameter vector $\mathbf{x}$ from the posterior distribution. I.e., sample $\mathbf{x} \sim c \cdot e^{-f(\mathbf{x})}$ given a gradient oracle for f

Original Bayesian motivation: Confidence intervals and uncertainty quantification. For new data point $\mathbf{a}_{n+1}$ with unknown label y_{n+1}, can sample from $p(y_i \mid b_1, \ldots, b_n)$ by sampling $\mathbf{x}$ from posterior and computing $M_{\mathbf{x}}(\mathbf{a}_{n+1})$.



BAYESIAN MACHINE LEARNING

For simple models (linear models, GLMS, kernel or Gaussian process regression, etc.) we have model-specific methods to sample from the posterior or compute confidence intervals.

Goal: Extend posterior sampling to any model that we can efficiently compute the gradient of (e.g., neural networks).

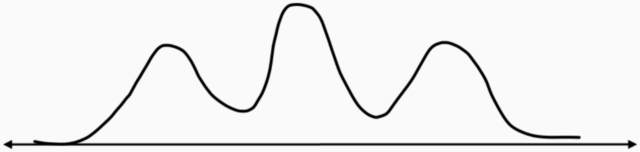
Why work with negative log posterior instead of directly working with posterior?

$$\prod_{i=1}^{n} \exp\left(-\frac{(b_i - M_{\mathbf{x}}(\mathbf{a}_i))^2}{2\sigma^2}\right) \quad \text{vs.} \quad \sum_{i=1}^{n}(b_i - M_{\mathbf{x}}(\mathbf{a}_i))^2$$

UNADJUSTED LANGEVIN ALGORITHM

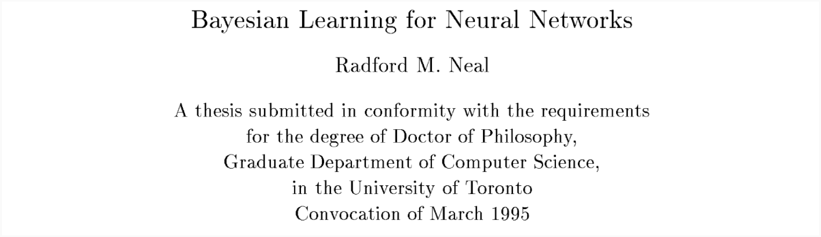
Unadjusted Langevin algorithm to sample from $e^{-f(\mathbf{x})}$:

- Choose starting point $\mathbf{x}_0 \in \mathbb{R}^d$, step size η.
- For $t = 0, \ldots, T$
  - $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta \cdot \nabla f(\mathbf{x}_t) + \sqrt{2\eta} \cdot \mathbf{g}_t$, where $\mathbf{g}_t \sim \mathcal{N}(0, I)$.



Like gradient descent. Far harder to analyze!

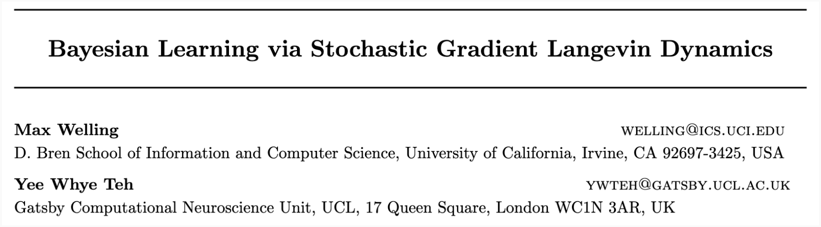
Widely used throughout computational science, statistics, and other fields since at least the 1990s.



Bayesian Learning for Neural Networks

Radford M. Neal

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy,
Graduate Department of Computer Science,
in the University of Toronto
Convocation of March 1995



Bayesian Learning via Stochastic Gradient Langevin Dynamics

Max Welling WELLING@ICS.UCI.EDU
D. Bren School of Information and Computer Science, University of California, Irvine, CA 92697-3425, USA

Yee Whye Teh YWTEH@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, UCL, 17 Queen Square, London WC1N 3AR, UK

UNADJUSTED LANGEVIN ALGORITHM

Unadjusted Langevin algorithm:

- Choose starting point $\mathbf{x}_0 \in \mathbb{R}^d$, step size η.
- For $t = 0, \ldots, T$
    - Sample $\mathbf{g}_t \sim \mathcal{N}(0, I)$.
    - $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta \cdot \nabla f(\mathbf{x}_t) + \sqrt{2\eta} \cdot \mathbf{g}_t$.

Informal claim: For $\eta \to 0$, the distribution of $\mathbf{x}_t$ converges to $c \cdot e^{-f(\mathbf{x})}$ for <u>many</u> natural distributions.

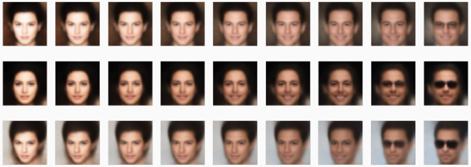
Suffices for f to be is strongly convex, mixture of distributions with this property, anything with Poincaré inequality, etc.

Non-asymptotic convergence rates have only been proven relatively recently, starting with [Durmus, Moulines, 2017].

MORE RECENT MOTIVATION: GENERATIVE AI

We have seen insane progress in (conditional) image generation.

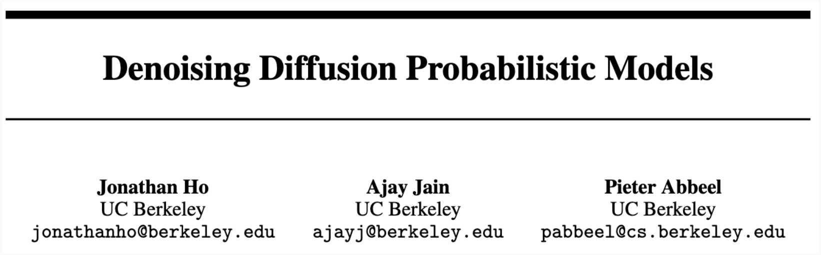
Just a few years ago, we got excited about images like:



MORE RECENT MOTIVATION: GENERATIVE AI

Older methods: Variational Auto-Encoders, Generative Adversarial Networks, normalizing flows, energy-based models, etc.

Leading modern methods: Denoising Diffusion Models, Score-based Generative Models



Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

ENERGY-BASED MODELS

View image generation as a sampling problem: $p(\mathbf{x})$ is the distribution over "natural images". Want to sample from p, or p conditioned on some prompt.

Energy-based models: Train model $M_{\boldsymbol{\theta}}$ that takes in an image $\mathbf{x}$ and returns a negative log probability. Given a training set of images $\mathbf{x}_1, \ldots, \mathbf{x}_n$, goal is to minimize:

$$\sum_{i=1}^{n} M_{\boldsymbol{\theta}}(\mathbf{x}_i) = \prod_{i=1}^{n} e^{-M_{\boldsymbol{\theta}}(\mathbf{x}_i)}.$$

Can sample new images using Langevin dynamics, where $f(\mathbf{x}) = M_{\boldsymbol{\theta}}(\mathbf{x}_i)$.

Lots of issues with normalization... how do you ensure $M_{\boldsymbol{\theta}}$ models a normalized probability density?

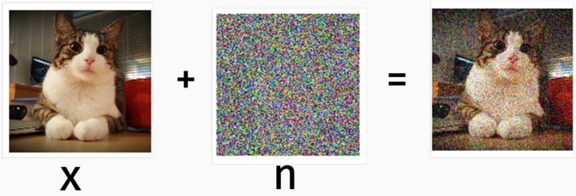
SCORE-BASED MODELS

Train a model that directly predicts $\nabla(-\log(p(\mathbf{x})))$. This is called the score function, but it is no different from the gradient ∇f we needed to implement Langevin dynamics.

How to train the model without input/output pairs?

$$(\mathbf{x}_i, \nabla(-\log(p(\mathbf{x}_\mathrm{i}))))$$

Methods to do so are called **score-matching methods**. Lots of cool algorithmic ideas. One approach based on adding noise:



Intuitivtely, $\nabla f(\mathbf{x}+\mathbf{n}) = -\mathbf{n}$.

UNADJUSTED LANGEVIN ALGORITHM

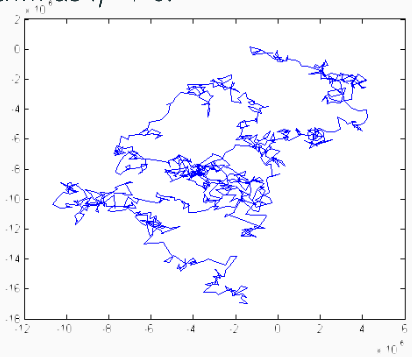
Unadjusted Langevin algorithm:

- Choose starting point $\mathbf{x}_0 \in \mathbb{R}^d$, step size η.
- For $t = 0, \ldots, T$
  - Sample $\mathbf{g}_t \sim \mathcal{N}(0, I)$.
  - $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta \cdot \nabla f(\mathbf{x}_t) + \sqrt{2\eta} \cdot \mathbf{g}_t$.

Informal claim: For $\eta \to 0$, the distribution of $\mathbf{x}_t$ converges to $c \cdot e^{-f(\mathbf{x})}$ for <u>many</u> natural distributions.

LANGEVIN DYNAMICS

Continuous-time Langevin Dynamics: Typical analysis begins by considering the continuous-time limit of the unadjusted Langevin algorithm as $\eta \to 0$.



To do so, we need to define a Brownian motion, which is the continuous limit of a Gaussian random walk.

BROWNIAN MOTION

(One-dimensional) Brownian motion: A Brownian motion B_t is a continuous function of time $t \geq 0$ with the properties:

- $B_0 = 0$
- For any $t_1 < t_2 < \ldots < t_n$, $B_{t_2 - t_1}, B_{t_3 - t_2}, \ldots, B_{t_n - t_{n-1}}$ are independent r.v.s.
- For any $t_1 < t_2$, $B_{t_2} - B_{t_1} \sim \mathcal{N}(0, t_2 - t_1)$.

dB_t denotes the instantanious change of the Brownian motion at time t. Think of dB_t as $\sqrt{dt} \cdot g$, where $g \sim \mathcal{N}(0, 1)$.

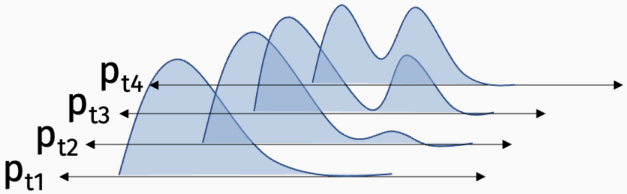
Langevin Stochastic Differential Equation:

$$dX_t = \underbrace{-f'(X_t)dt}_{\text{drift term}} + \underbrace{\sqrt{2} \cdot dB_t}_{\text{diffusion term}} .$$

STEADY-STATE OF LANGEVIN EQUATION

$$dX_t = -f'(X_t)dt + \sqrt{2}dB_t$$

Claim: The distribution $c \cdot e^{-f(X)}$ is an invariant distribution of the Langevin SDE. If $X_0 \sim c \cdot e^{-f(X)}$, then $X_t \sim c \cdot e^{-f(X)}$ $\forall t > 0$.



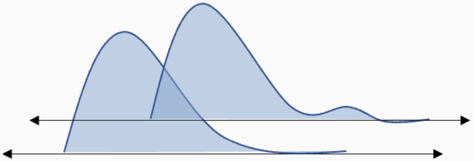
Let p_t be the distribution of X_t. The goal is to show that $\frac{d}{dt}p_t(X) = 0$ for all $X \in \mathbb{R}$ when $p_t(X) = c \cdot e^{-f(X)}$.

STEADY-STATE OF LANGEVIN EQUATION

Fokker-Planck Equation: If $dX_t = z(X_t)dt + \sqrt{2}dB_t$, then

$$\begin{aligned}\frac{d}{dt}p_t(X) &= -\frac{d}{dX}[z(X)p_t(X)] + p_t''(x)\\ &= -z'(X)p_t(X) - z(X)p_t'(X) + p_t''(x)\end{aligned}$$

(p_t' and p_t'' denote the derivatives of p_t <u>with respect to X.</u>)



STEADY-STATE OF LANGEVIN EQUATION

Fokker-Planck Equation: If $dX_t = z(X_t)dt + \sqrt{2}dB_t$, then

$$\frac{d}{dt}p_t(X) = -z'(X)p_t(X) - z(X)p_t'(X) + p_t''(x)$$

Claim: If $z(X) = -f'(X)$ for some f, then $\frac{d}{dt}p_t(X) = 0$ for all $X \in \mathbb{R}$ when $p_t(X) = c \cdot e^{-f(X)}$.

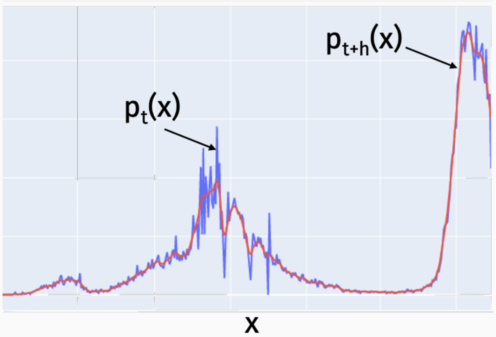
Proof:

FOKKER-PLANCK INTUITION

Diffusion-only Fokker-Planck Equation: If $dX_t = \sqrt{2}dB_t$, then

$$\frac{d}{dt}p_t(X) = p_t''(x)$$

Adding Gaussian noise at each time step smooths the distribution. Can be thought of as a moving average.

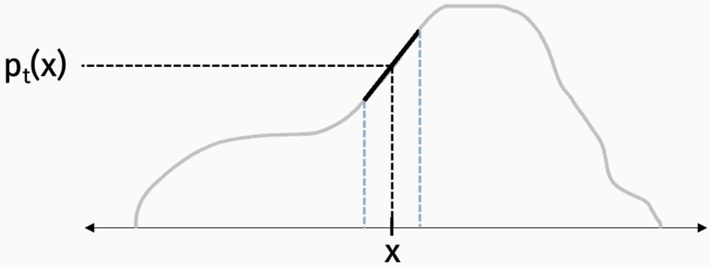


FOKKER-PLANCK INTUITION

Diffusion-only Fokker-Planck Equation: If $dX_t = \sqrt{2}dB_t$, then

$$\frac{d}{dt}p_t(X) = p_t''(x)$$

How does moving avergage change $p_t(X)$?

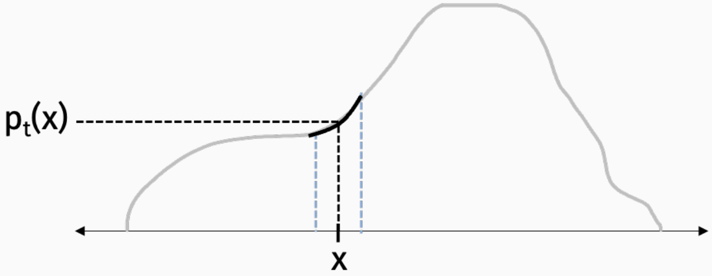


FOKKER-PLANCK INTUITION

Diffusion-only Fokker-Planck Equation: If $dX_t = \sqrt{2}dB_t$, then

$$\frac{d}{dt}p_t(X) = p_t''(x)$$

How does moving avergage change $p_t(X)$?

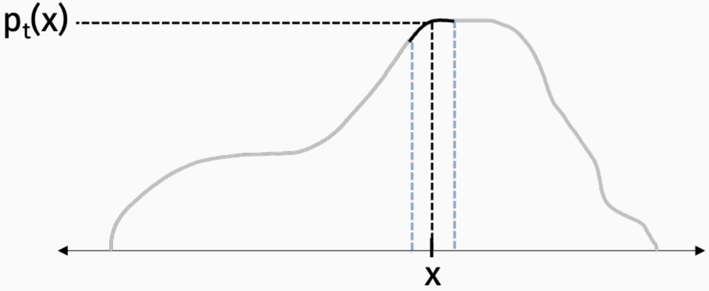


FOKKER-PLANCK INTUITION

Diffusion-only Fokker-Planck Equation: If $dX_t = \sqrt{2}dB_t$, then

$$\frac{d}{dt}p_t(X) = p_t''(x)$$

How does moving avergage change $p_t(X)$?



FOKKER–PLANCK INTUITION

Diffusion-only Fokker-Planck Equation: If $dX_t = \sqrt{2} dB_t$, then

$$\frac{d}{dt} p_t(X) = p_t''(x)$$

$$\frac{d}{dt} p_t(X) = \lim_{h \to 0} \frac{p_{t+h}(X) - p_t(X)}{h}$$

Very informal argument:

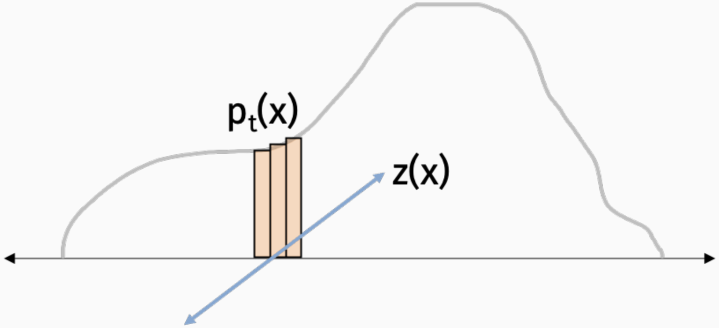
$$\begin{aligned} p_{t+h}(x) &\approx \frac{2}{\sqrt{h}} \int_{x-\sqrt{h}}^{x+\sqrt{h}} p_t(y) dy \\ &\approx \frac{2}{\sqrt{h}} \int_{x-\sqrt{h}}^{x+\sqrt{h}} p_t(x) + p_t'(x)(y-x) + \frac{1}{2} p_t''(x)(y-x)^2 + o(h)\, dy \end{aligned}$$

$$\begin{aligned} p_{t+h}(x) - p_t(x) &\approx \frac{2}{\sqrt{h}} \int_{x-\sqrt{h}}^{x+\sqrt{h}} \frac{1}{2} p_t''(x)(y-x)^2\, dy \\ &= \frac{1}{\sqrt{h}} p_t''(x) \frac{1}{3} (y-x)^3 \Big|_{x-\sqrt{h}}^{x+\sqrt{h}} \propto p_t''(x) h. \end{aligned}$$

FOKKER-PLANCK INTUITION

Drift-only Fokker-Planck Equation: If $dX_t = z(X_t)dt$, then

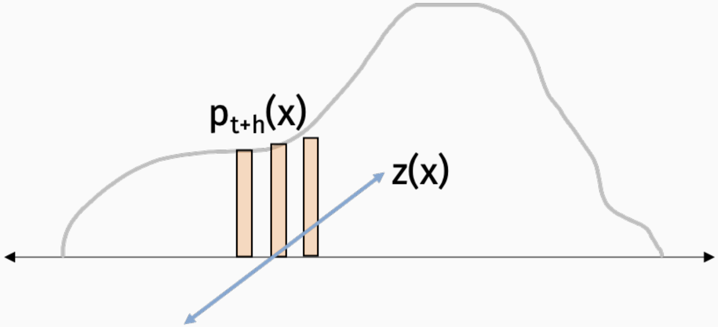
$$\frac{d}{dt}p_t(X) = -\frac{d}{dX}[z(X)p_t(X)] = -z'(X)p_t(X) - z(X)p_t'(X)$$



FOKKER-PLANCK INTUITION

Drift-only Fokker-Planck Equation: If $dX_t = z(X_t)dt$, then

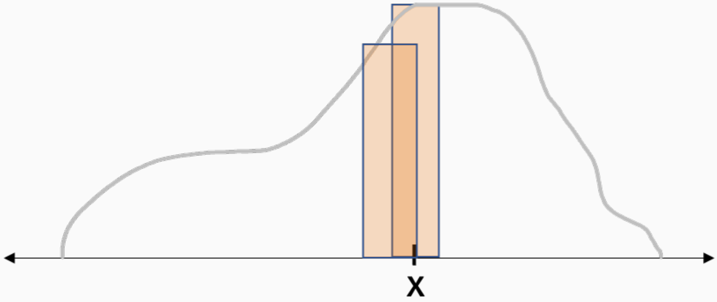
$$\frac{d}{dt}p_t(X) = -\frac{d}{dX}[z(X)p_t(X)] = -z'(X)p_t(X) - z(X)p_t'(X)$$



FOKKER-PLANCK INTUITION

Drift-only Fokker-Planck Equation: If $dX_t = z(X_t)dt$, then

$$\frac{d}{dt}p_t(X) = -\frac{d}{dX}[z(X)p_t(X)] = -z'(X)p_t(X) - z(X)p_t'(X)$$



STEADY-STATE OF LANGEVIN EQUATION

$$dX_t = -f'(X_t)dt + \sqrt{2}dB_t$$

Claim: The distribution $c \cdot e^{-f(X)}$ is an invariant distribution of the Langevin SDE. If $X_0 \sim c \cdot e^{-f(X)}$, then $X_t \sim c \cdot e^{-f(X)}$ $\forall t > 0$.

To get meaningful algorithmic results, need to show:

1. (Fast) convergence to this invariant distribution.
2. Discretization argument to show that the discrete-time ULA also converges close ot the invariant distribution.

EXAMPLE THEORETICAL RESULT

Flavor of result people are interested in proving:

Theorem (See e.g., Chewi 2024)

Suppose f is an α-smooth, β-strongly convex function with condition number $\kappa = \alpha/\beta$. Then after:

$$T = \tilde{O}(\kappa d/\epsilon^2) \text{ iterations},$$

the unadjusted Langevin algorithm returns a sample from a distribution $\mathcal{P}$ satisfying:

$$W_2(\mathcal{P}, c \cdot e^{-f(\mathbf{x})}) \leq \epsilon,$$

where W_2 is the Wasserstein-2 distance.

SIMPLE CASE: GAUSSIAN DENSITY

Suppose we want to sample from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Let $\mathbf{H} = \boldsymbol{\Sigma}^{-1}$.

$$p(\mathbf{x}) \propto \exp(-f(\mathbf{x})) \quad \text{where} \quad f(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{H}(\mathbf{x} - \boldsymbol{\mu})$$

$$\nabla f(\mathbf{x}) = \mathbf{H}(\mathbf{x} - \boldsymbol{\mu}).$$

Unadjusted Langevin algorithm:

- $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \eta \cdot \nabla f(\mathbf{x}_{t-1}) + \sqrt{2\eta} \cdot \mathbf{g}_{t-1}$.

If we initialized $\mathbf{x}_0$ as a Gaussain, then every iterate is Gaussian distributed. I.e. $\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. Want to show:

$$\boldsymbol{\mu}_t \rightarrow \boldsymbol{\mu} \qquad\qquad \boldsymbol{\Sigma}_t \rightarrow \boldsymbol{\Sigma}.$$

SIMPLE CASE: GAUSSIAN DENSITY

Suppose we want to sample from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Let $\mathbf{H} = \boldsymbol{\Sigma}^{-1}$.

$$p(\mathbf{x}) \propto \exp(-f(\mathbf{x})) \quad \text{where} \quad f(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{H}(\mathbf{x} - \boldsymbol{\mu})$$

$$\nabla f(\mathbf{x}) = \mathbf{H}(\mathbf{x} - \boldsymbol{\mu}).$$

Unadjusted Langevin algorithm:

- $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \eta \cdot \nabla f(\mathbf{x}_{t-1}) + \sqrt{2\eta} \cdot \mathbf{g}_{t-1}$.

Plugging in definition of gradient:

$$\begin{aligned}
\mathbf{x}_t &= \mathbf{x}_{t-1} - \eta \mathbf{H}(\mathbf{x}_{t-1} - \boldsymbol{\mu}) + \sqrt{2\eta} \cdot \mathbf{g}_{t-1} \\
(\mathbf{x}_t - \boldsymbol{\mu}) &= (\mathbf{x}_{t-1} - \boldsymbol{\mu}) - \eta \mathbf{H}(\mathbf{x}_{t-1} - \boldsymbol{\mu}) + \sqrt{2\eta} \cdot \mathbf{g}_{t-1} \\
&= (\mathbf{I} - \eta \mathbf{H})(\mathbf{x}_{t-1} - \boldsymbol{\mu}) + \sqrt{2\eta} \cdot \mathbf{g}_{t-1}
\end{aligned}$$

CONVERGENCE OF MEAN

Unrolling the iteration:

$$\begin{aligned}(\mathbf{x}_t - \boldsymbol{\mu}) &= (\mathbf{I} - \eta\mathbf{H})(\mathbf{x}_{t-1} - \boldsymbol{\mu}) + \sqrt{2\eta}\cdot \mathbf{g}_{t-1} \\ &= (\mathbf{I} - \eta\mathbf{H})((\mathbf{I} - \eta\mathbf{H})(\mathbf{x}_{t-2} - \boldsymbol{\mu}) + \sqrt{2\eta}\cdot \mathbf{g}_{t-2}) + \sqrt{2\eta}\cdot \mathbf{g}_{t-1} \\ &\vdots \\ &= (\mathbf{I} - \eta\mathbf{H})^t(\mathbf{x}_0 - \boldsymbol{\mu}) + \sqrt{2\eta}\sum_{i=0}^{t-1}(\mathbf{I} - \eta\mathbf{H})^{t-1-i}\mathbf{g}_i\end{aligned}$$

First observation: If we choose $\eta = 1/\lambda_{\max}(\mathbf{H})$, then $\|\mathbb{E}[\mathbf{x}_t] - \boldsymbol{\mu}\|_2 \leq \epsilon\|\mathbb{E}[\mathbf{x}_0] - \boldsymbol{\mu}\|_2$ after $t = O(\kappa \log(1/\epsilon))$ iterations.

In other words, we quickly converge to a Gaussian distribution with the correct mean!

CONVERGENCE OF MEAN

$$(\mathbf{x}_t - \boldsymbol{\mu}) = (\mathbf{I} - \eta \mathbf{H})^t (\mathbf{x}_0 - \boldsymbol{\mu}) + \sqrt{2\eta} \sum_{i=0}^{t-1} (\mathbf{I} - \eta \mathbf{H})^{t-1-i} \mathbf{g}_i$$

First observation: If we choose $\eta = 1/\lambda_{\max}(\mathbf{H})$, then $\|\mathbb{E}[\mathbf{x}_t] - \boldsymbol{\mu}\|_2 \leq \epsilon \|\mathbb{E}[\mathbf{x}_0] - \boldsymbol{\mu}\|_2$ after $t = O(\kappa \log(1/\epsilon))$ iterations.

CONVERGENCE OF COVARIANCE

What about the covariance matrix?

$$\begin{aligned}
\boldsymbol{\Sigma}_t &= \mathbb{E}\left[(\mathsf{x}_t - \boldsymbol{\mu}_t)(\mathsf{x}_t - \boldsymbol{\mu}_t)^T\right] \\
&= \left((\mathsf{I} - \eta\mathsf{H})^t(\mathsf{x}_0 - \boldsymbol{\mu}) + \sqrt{2\eta}\sum_{i=0}^{t-1}(\mathsf{I} - \eta\mathsf{H})^{t-1-i}\mathsf{g}_i - (\boldsymbol{\mu}_t - \boldsymbol{\mu}_t)\right)\left(\cdots\right)^T \\
&= \left((\mathsf{I} - \eta\mathsf{H})^t(\mathsf{x}_0 - \boldsymbol{\mu}_0) + \sqrt{2\eta}\sum_{i=0}^{t-1}(\mathsf{I} - \eta\mathsf{H})^{t-1-i}\mathsf{g}_i\right)\left(\cdots\right)^T
\end{aligned}$$

Basically all cross-terms cancel. If we assume $\mathsf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, I)$, we get:

$$\begin{aligned}
\boldsymbol{\Sigma}_t &= (\mathsf{I} - \eta\mathsf{H})^{2t} + 2\eta\sum_{i=0}^{t-1}(\mathsf{I} - \eta\mathsf{H})^{2t-2-2i} \\
&= (\mathsf{I} - \eta\mathsf{H})^{2t} + 2\eta\sum_{i=0}^{t-1}(\mathsf{I} - \eta\mathsf{H})^{2i}
\end{aligned}$$

CONVERGENCE OF COVARIANCE

$$\boldsymbol{\Sigma}_t = (\mathsf{I} - \eta \mathsf{H})^{2t} + 2\eta \sum_{i=0}^{t-1} (\mathsf{I} - \eta \mathsf{H})^{2i}$$

We have that $\sum_{i=0}^{\infty} \mathsf{A}^i = (\mathsf{I} - \mathsf{A})^{-1}$ and thus:

$$\sum_{i=0}^{t-1} \mathsf{A}^i = (\mathsf{I} - \mathsf{A})^{-1} - \mathsf{A}^t (\mathsf{I} - \mathsf{A})^{-1}.$$

Apply to $\mathsf{A} = (\mathsf{I} - \eta \mathsf{H})^2 = \mathsf{I} - 2\eta \mathsf{H} + \eta^2 \mathsf{H}^2$

$$\begin{aligned}\boldsymbol{\Sigma}_t &= 2\eta \left(2\eta \mathsf{H} - \eta^2 \mathsf{H}\right)^{-1} - (\mathsf{I} - \eta \mathsf{H})^{2t} \left(2\eta \mathsf{H} - \eta^2 \mathsf{H}\right)^{-1} + (\mathsf{I} - \eta \mathsf{H})^{2t} \\ &= (\mathsf{H} + .5\eta \mathsf{H}^2)^{-1} - (\mathsf{I} - \eta \mathsf{H})^{2t} \left(2\eta \mathsf{H} - \eta^2 \mathsf{H}\right)^{-1} + (\mathsf{I} - \eta \mathsf{H})^{2t} \\ &\approx \mathsf{H}^{-1}\end{aligned}$$

for small enough η.

ALGORITHMIC QUESTIONS

Can we accelerate convergence using the existing toolkit of optimization tricks?

- Acceleration/momentum, preconditioning, variance reduction, etc.

Can we take advantage of additional oracles, e.g. that can draw samples $x \sim e^{-f(x)}$?

- See e.g. [Koehler, Vuong, 2023]

Lower bounds on gradient oracle complexity?

- See e.g. [Chewi, de Dios Pont, Li, Lu, Narayanan, 2024]