Generative Priors in Data Science: From Low-rank to Diffusion Models

Yuejie Chi

Carnegie Mellon University

NASIT 2024

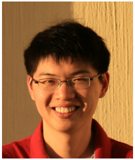
My wonderful collaborators

X. Xu

CMU

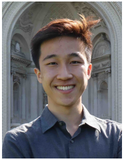
T. Tong

CMU



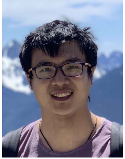
C. Ma

Chicago



H. Dong

CMU



Y. Shen

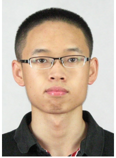
CMU

Y. Wei

UPenn

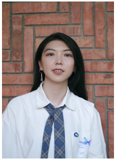
Y. Chen

UPenn



G. Li

CUHK



Y. Huang

UPenn

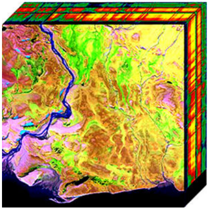
T. Efimov

CMU

Sensing, imaging and computing advances

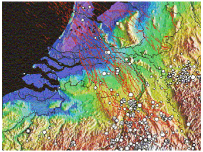
healthcare

Radio astronomy

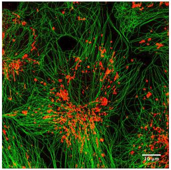


hyperspectral

Internet traffic



seismic imaging



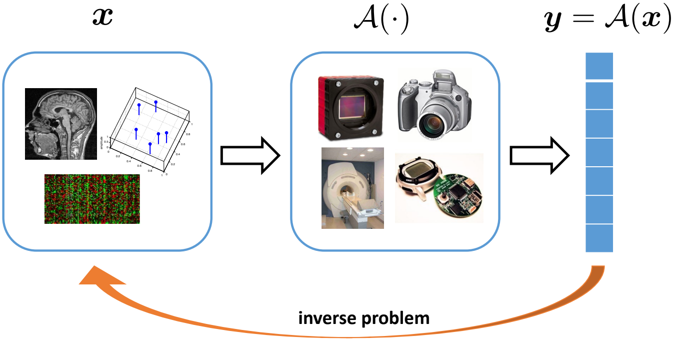
microscopy

Data Science at the Singularity

— David Donoho, HDSR

Inverse problems

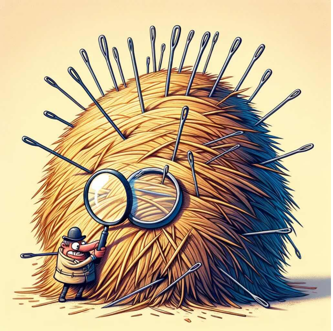
Forward model: we interrogate the signal of interest $\boldsymbol{x}$ through forward model $\mathcal{A}$ and make measurements $\boldsymbol{y}$.



Inverse problem: recover the signal of interest $\boldsymbol{x}$ from $\boldsymbol{y}$.

Challenges: finding needles in a haystack

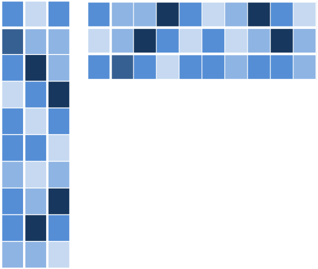
- **Sampling constraints:** sample-starved, low signal-to-noise ratio, nonlinear measurements;
- **Ill-conditioned sources:** weak and fine-grained information;
- **Resiliency:** miscalibration, missing data, corruptions, etc.



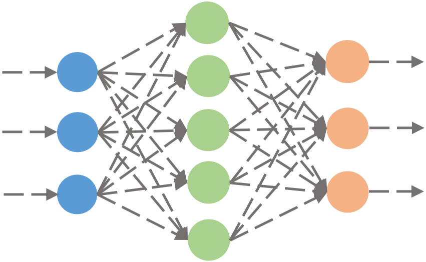
DALLE generated with the prompt "finding needles in the haystack"

Geometry as a prior: from low-rank to generative models

How do we learn effectively leveraging the data priors?



Subspace models:
Sparsity, low-rank, …



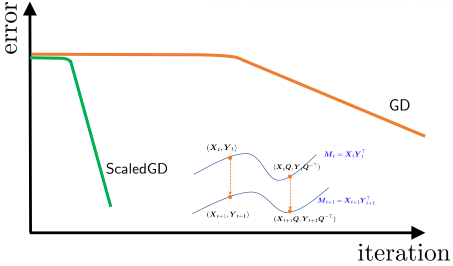
Neural networks:
GAN, VAE, diffusion models…

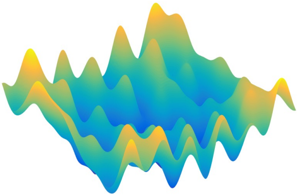
First vignette: low-rank models

An optimization vignette: preconditioning to accelerate nonconvex ill-conditioned low-rank estimation

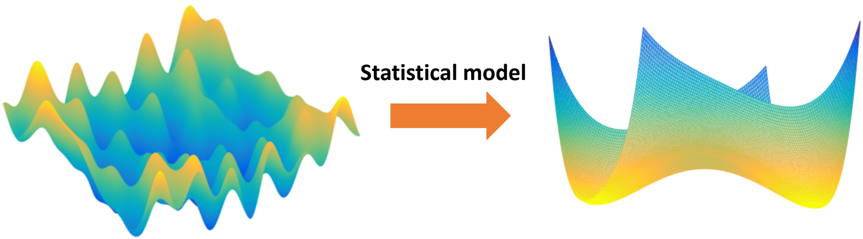
preconditioning

overparameterization





Statistics meets optimization



Statistical model

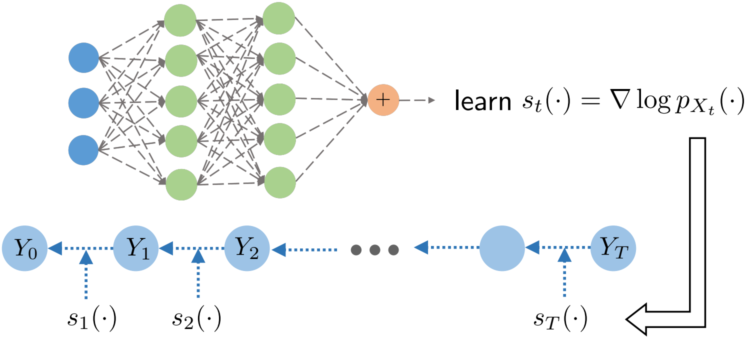
worst case

average case

Simple algorithms can be efficient for nonconvex problems on average!

Second vignette: diffusion models

A sampling vignette: how can we leverage score-based generative models for generation and inverse problems, efficiently and provably?

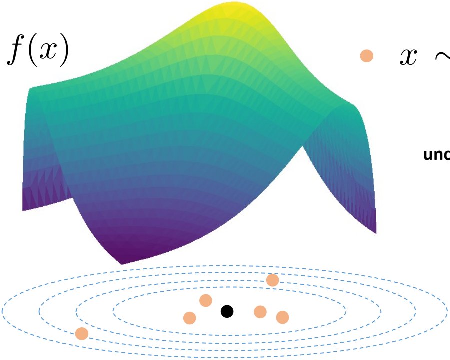


learn $s_t(\cdot) = \nabla \log p_{X_t}(\cdot)$

Sampling meets optimization

- $x^\star = \max_x \; f(x)$

Optimization delivers point estimate



● $x \sim p(x) \propto e^{f(x)}$

Sampling provides uncertainty quantification

Sampling as an alternative to optimization via energy-based modeling.

Part 1:

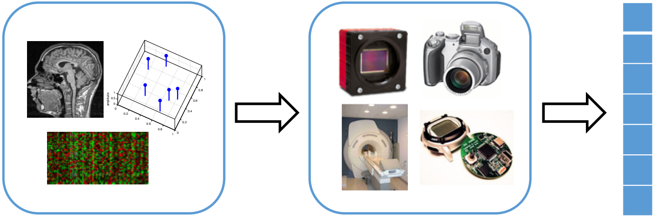
Accelerating gradient descent for ill-conditioned low-rank estimation

A canonical problem: low-rank matrix sensing

$\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$
$\text{rank}(\boldsymbol{M}) = r$

$\mathcal{A}(\cdot)$
linear map

$\boldsymbol{y} \in \mathbb{R}^m$



$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{M}) + \text{noise}$$

Recover M in the sample-starved regime:

$$\underbrace{(n_1 + n_2)r}_{\text{degree of freedom}} \lesssim \underbrace{m}_{\text{sensing budget}} \ll \underbrace{n_1 n_2}_{\text{ambient dimension}}$$

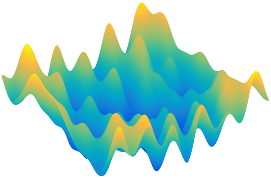
Low-rank matrix factorization

$$\min_{\boldsymbol{Z}\in\mathbb{R}^{n_1\times n_2}} \quad \mathsf{rank}(\boldsymbol{Z}) \qquad\qquad \mathsf{s.t.} \quad \boldsymbol{y}\approx\mathcal{A}(\boldsymbol{Z})$$

$$\Downarrow$$

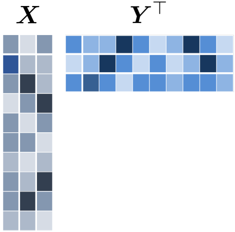
$$\min_{\mathrm{rank}(\boldsymbol{Z})=r} \quad \frac{1}{2}\left\|\boldsymbol{y}-\mathcal{A}(\boldsymbol{Z})\right\|_2^2$$

scalable, but nonconvex!



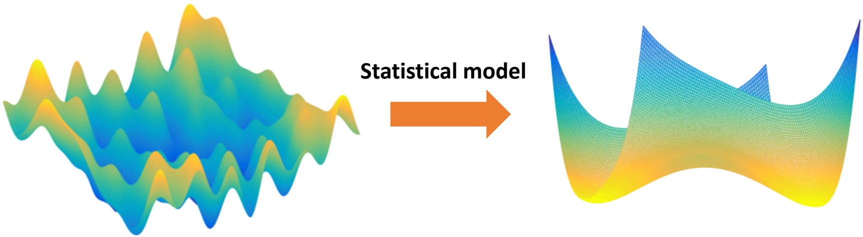
$$\Downarrow$$

$\boldsymbol{Z} =$



$$\min_{\boldsymbol{X}\in\mathbb{R}^{n_1\times r},\boldsymbol{Y}\in\mathbb{R}^{n_2\times r}} \quad f(\boldsymbol{X},\boldsymbol{Y})=\frac{1}{2}\left\|\boldsymbol{y}-\mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top)\right\|_2^2$$

Statistics meets optimization



Statistical model

worst case

average case

Simple algorithms can be efficient for nonconvex problems!

Vanilla gradient descent (GD):

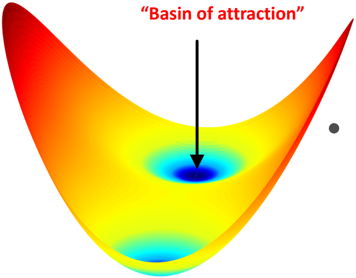
$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta \nabla_{\boldsymbol{X}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t)$$
$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t - \eta \nabla_{\boldsymbol{Y}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t)$$

for $t = 0, 1, \ldots$ from a carefully chosen (e.g., spectral) initialization.

Low-rank matrix sensing: GD with balancing regularization

$$\min_{\boldsymbol{X},\boldsymbol{Y}} f(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top)\right\|_2^2 = \frac{1}{2}\left\|\mathcal{A}(\boldsymbol{M} - \boldsymbol{X}\boldsymbol{Y}^\top)\right\|_2^2$$

$$\min_{\boldsymbol{X},\boldsymbol{Y}} f_{\rm reg}(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top)\right\|_2^2 + \frac{1}{8}\left\|\boldsymbol{X}^\top\boldsymbol{X} - \boldsymbol{Y}^\top\boldsymbol{Y}\right\|_{\rm F}^2$$



- **Spectral initialization:** find an initial point in the "basin of attraction".

$$(\boldsymbol{X}_0, \boldsymbol{Y}_0) \leftarrow \mathsf{SVD}_r(\mathcal{A}^*(\boldsymbol{y}))$$

- **Gradient iterations:**

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta\,\nabla_{\boldsymbol{X}} f_{\rm reg}(\boldsymbol{X}_t, \boldsymbol{Y}_t)$$
$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t - \eta\,\nabla_{\boldsymbol{Y}} f_{\rm reg}(\boldsymbol{X}_t, \boldsymbol{Y}_t)$$

for $t = 0, 1, \ldots$

Recap: GD for asymmetric low-rank matrix sensing

Theorem (Tu et al., ICML 2016)

Suppose $\boldsymbol{M} = \boldsymbol{X}_\star \boldsymbol{Y}_\star^\top$ *is rank-*r *and has a condition number* $\kappa = \sigma_{\max}(\boldsymbol{M})/\sigma_{\min}(\boldsymbol{M})$*. For low-rank matrix sensing with i.i.d. Gaussian design, vanilla GD (with spectral initialization) achieves*

$$\|\boldsymbol{X}_t \boldsymbol{Y}_t^\top - \boldsymbol{M}\|_{\mathrm{F}} \leq \varepsilon \cdot \sigma_{\min}(\boldsymbol{M})$$

- **Computational:** *within* $O\big(\kappa \log \frac{1}{\varepsilon}\big)$ *iterations;*
- **Statistical:** *as long as the sample complexity satisfies*

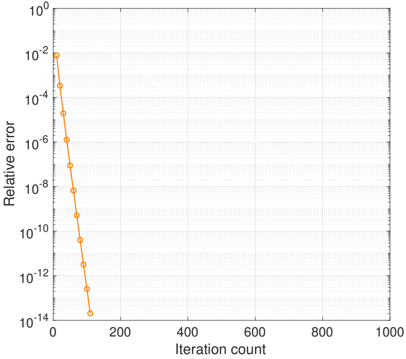
$$m \gtrsim (n_1 + n_2) r^2 \kappa^2 .$$

Similar results hold for many low-rank problems: matrix completion, robust PCA, etc...

(Netrapalli et al. '13, Candès, Li, Soltanolkotabi '14, Sun and Luo '15, Chen and Wainwright '15, Zheng and Lafferty '15, Ma et al. '17, ...)

Global linear convergence of vanilla GD

$$\min_{\boldsymbol{X},\boldsymbol{Y}} \quad f(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\mathcal{P}_{\Omega}(\boldsymbol{X}\boldsymbol{Y}^{\top} - \boldsymbol{M})\right\|_{\mathrm{F}}^{2}$$

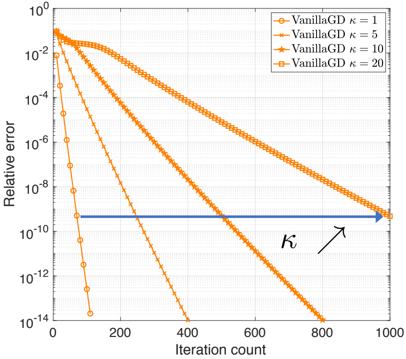


Similar results hold for many low-rank problems...

(Tu et al. '16, Netrapalli et al. '13, Candès, Li, Soltanolkotabi '14, Sun and Luo '15, Chen and Wainwright '15, Zheng and Lafferty '15, Ma et al. '17, ...)

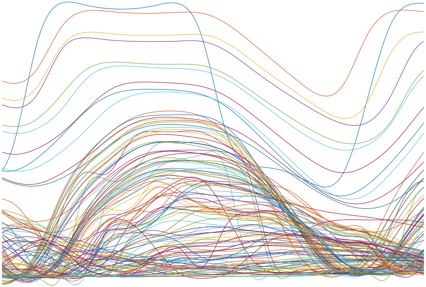
What could go wrong?

$$\min_{\boldsymbol{X},\boldsymbol{Y}} \quad f(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\mathcal{P}_{\Omega}(\boldsymbol{X}\boldsymbol{Y}^{\top} - \boldsymbol{M})\right\|_{\mathrm{F}}^{2}$$

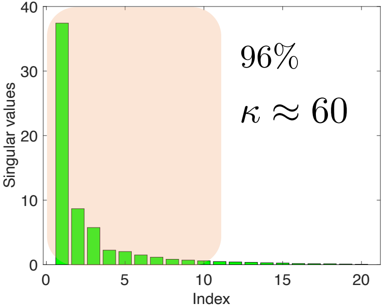


Vanilla GD converges in $O\left(\kappa \log \frac{1}{\varepsilon}\right)$ iterations.

Condition number can be large



chlorine concentration levels
120 junctions, 180 time slots

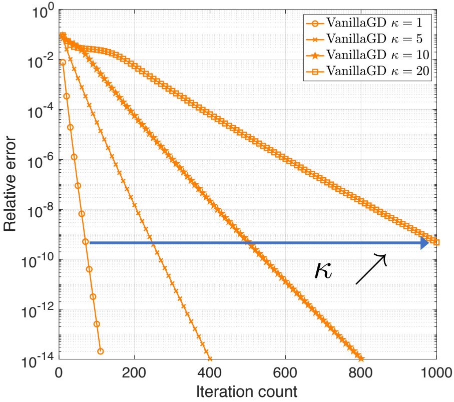


rank-10 approximation

Must mind the condition number!

Data source: `www.epa.gov/water-research/epanet`

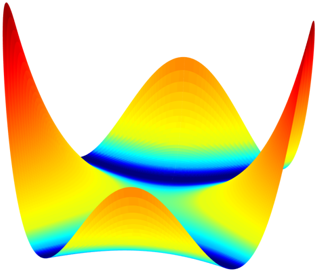
Getting rid of the condition number?



Can we accelerate the convergence rate of GD to $O(\log \frac{1}{\varepsilon})$*?*

Our recipe: scaled gradient descent (ScaledGD)

$$f(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top)\right\|_2^2$$



- **Spectral initialization:** find an initial point in the "basin of attraction".
- **Scaled gradient iterations:**

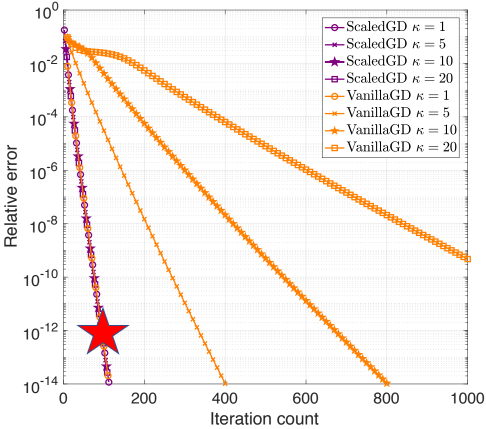
$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta \nabla_{\boldsymbol{X}} f(\boldsymbol{X}_t,\boldsymbol{Y}_t) \underbrace{(\boldsymbol{Y}_t^\top \boldsymbol{Y}_t)^{-1}}_{\texttt{preconditioner}}$$

$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t - \eta \nabla_{\boldsymbol{Y}} f(\boldsymbol{X}_t,\boldsymbol{Y}_t) \underbrace{(\boldsymbol{X}_t^\top \boldsymbol{X}_t)^{-1}}_{\texttt{preconditioner}}$$

for $t = 0, 1, \ldots$

ScaledGD is a *preconditioned* gradient method
without balancing regularization!

ScaledGD for low-rank matrix completion



Huge computational saving: ScaledGD converges in an κ-independent manner with a minimal overhead!

What could go wrong with vanilla GD?

Low-rank matrix factorization:

$$f(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{2} \left\| \boldsymbol{X}\boldsymbol{Y}^\top - \boldsymbol{X}_\star \boldsymbol{Y}_\star^\top \right\|_{\mathrm{F}}^2$$

The rank-1 scalar case: $M_\star = X_\star Y_\star$ and $M_t = X_t Y_t$.

Unbalanced factor. Suppose $X_t = \sqrt{K M_t}$, $Y_t = \sqrt{K^{-1} M_t}$ for some large $K > 1$, GD updates follows

$$\begin{aligned} X_{t+1} &= X_t - \eta(X_t Y_t - M_\star) Y_t = \sqrt{K M_t} - \eta(M_t - M_\star)\sqrt{K^{-1} M_t}, \\ Y_{t+1} &= Y_t - \eta(X_t Y_t - M_\star) X_t = \sqrt{K^{-1} M_t} - \eta(M_t - M_\star)\sqrt{K M_t}. \end{aligned}$$

The learning rate is set as $\eta \propto K^{-1}$ to avoid gradient explosion of Y_t, resulting in slow convergence of X_t.

Vanilla GD suffers from unbalancing.

...unless using a balanced initialization, see (Ma et al., 2021).

A closer peek at the caveats of vanilla GD

Low-rank matrix factorization:

$$f(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\boldsymbol{X}\boldsymbol{Y}^\top - \boldsymbol{X}_\star\boldsymbol{Y}_\star^\top\right\|_{\mathrm{F}}^2$$

The rank-2 case: $\boldsymbol{M}_\star = \begin{bmatrix} \sigma_\star^1 & 0 \\ 0 & \sigma_\star^2 \end{bmatrix}$ and $\boldsymbol{M}_t = \begin{bmatrix} \sigma_t^1 & 0 \\ 0 & \sigma_t^2 \end{bmatrix}$ with $\kappa = \frac{\sigma_\star^1}{\sigma_\star^2}$.

Balanced, but ill-conditioned factors. Let $\boldsymbol{X}_t = \boldsymbol{Y}_t = \begin{bmatrix} \sqrt{\sigma_t^1} & 0 \\ 0 & \sqrt{\sigma_t^2} \end{bmatrix}$,

GD update follows

$$\boldsymbol{X}_{t+1} = \boldsymbol{Y}_{t+1} = \begin{bmatrix} \sqrt{\sigma_t^1}[1-\eta(\sigma_t^1 - \sigma_\star^1)] & 0 \\ 0 & \sqrt{\sigma_t^2}[1-\eta(\sigma_t^2 - \sigma_\star^2)] \end{bmatrix}.$$

The learning rate is set as $\eta \propto (\sigma_\star^1)^{-1}$ to avoid gradient explosion of the top diagonal entry, resulting in slow convergence of the other.

Vanilla GD suffers from ill-conditioning.

ScaledGD as a quasi-Newton method

Low-rank matrix factorization:

$$f(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{2}\left\|\boldsymbol{X}\boldsymbol{Y}^\top - \boldsymbol{X}_\star\boldsymbol{Y}_\star^\top\right\|_{\mathrm{F}}^2$$

ScaledGD is equivalent to:

$$\begin{bmatrix}\mathsf{vec}(\boldsymbol{X})\\ \mathsf{vec}(\boldsymbol{Y})\end{bmatrix} \Longleftarrow \begin{bmatrix}\mathsf{vec}(\boldsymbol{X})\\ \mathsf{vec}(\boldsymbol{Y})\end{bmatrix} - \eta \begin{bmatrix}\nabla^2_{\boldsymbol{X},\boldsymbol{X}} f & \mathbf{0}\\ \mathbf{0} & \nabla^2_{\boldsymbol{Y},\boldsymbol{Y}} f\end{bmatrix}^{-1} \begin{bmatrix}\mathsf{vec}(\nabla_{\boldsymbol{X}} f)\\ \mathsf{vec}(\nabla_{\boldsymbol{Y}} f)\end{bmatrix}.$$

The preconditioners are chosen as the inverse of the block diagonal approximation of the Hessian to low-rank matrix factorization.

ScaledGD is insensitive to ill-conditioning

Recall the rank-2 case with *balanced, but ill-conditioned factors.*

- GD update follows

$$\boldsymbol{X}_{t+1} = \boldsymbol{Y}_{t+1} = \begin{bmatrix} \sqrt{\sigma_t^1}[1 - \eta(\sigma_t^1 - \sigma_\star^1)] & 0 \\ 0 & \sqrt{\sigma_t^2}[1 - \eta(\sigma_t^2 - \sigma_\star^2)] \end{bmatrix}.$$

The learning rate is set as $\eta \propto (\sigma_\star^1)^{-1}$.

- ScaledGD update follows

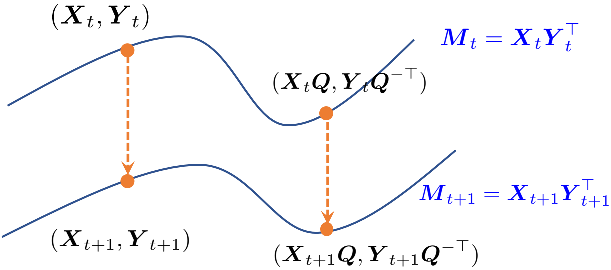
$$\boldsymbol{X}_{t+1} = \boldsymbol{Y}_{t+1} = \begin{bmatrix} \sqrt{\sigma_t^1}[1 - \eta(1 - \frac{\sigma_\star^1}{\sigma_t^1})] & 0 \\ 0 & \sqrt{\sigma_t^2}[1 - \eta(1 - \frac{\sigma_\star^2}{\sigma_t^2})] \end{bmatrix}.$$

The learning rate is set as $\eta \propto 1$.

ScaledGD is insensitive to ill-conditioning.

Key properties of ScaledGD

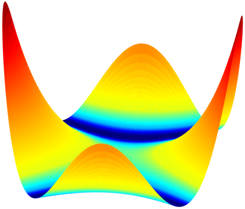
Invariance to invertible transforms: (Tanner and Wei, '16; Mishra '16)



$(\boldsymbol{X}_t, \boldsymbol{Y}_t)$
$(\boldsymbol{X}_t\boldsymbol{Q}, \boldsymbol{Y}_t\boldsymbol{Q}^{-\top})$
$\boldsymbol{M}_t = \boldsymbol{X}_t\boldsymbol{Y}_t^\top$
$(\boldsymbol{X}_{t+1}, \boldsymbol{Y}_{t+1})$
$\boldsymbol{M}_{t+1} = \boldsymbol{X}_{t+1}\boldsymbol{Y}_{t+1}^\top$
$(\boldsymbol{X}_{t+1}\boldsymbol{Q}, \boldsymbol{Y}_{t+1}\boldsymbol{Q}^{-\top})$

New distance metric as Lyapunov function:

$$\mathrm{dist}^2\left(\begin{bmatrix}\boldsymbol{X}\\\boldsymbol{Y}\end{bmatrix}, \begin{bmatrix}\boldsymbol{X}_\star\\\boldsymbol{Y}_\star\end{bmatrix}\right) = \inf_{\boldsymbol{Q}\in\mathrm{GL}(r)} \left\|(\boldsymbol{X}\boldsymbol{Q}-\boldsymbol{X}_\star)\boldsymbol{\Sigma}_\star^{1/2}\right\|_\mathrm{F}^2 + \left\|(\boldsymbol{Y}\boldsymbol{Q}^{-\top}-\boldsymbol{Y}_\star)\boldsymbol{\Sigma}_\star^{1/2}\right\|_\mathrm{F}^2$$



+ a careful trajectory-based analysis

Theoretical guarantees of ScaledGD

Theorem (Tong, Ma and Chi, JMLR 2021)

For low-rank matrix sensing with i.i.d. Gaussian design, ScaledGD with spectral initialization achieves

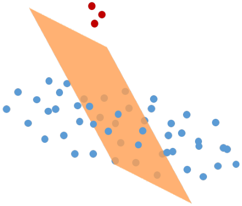
$$\|\boldsymbol{X}_t \boldsymbol{Y}_t^\top - \boldsymbol{M}\|_{\mathrm{F}} \lesssim \varepsilon \cdot \sigma_{\min}(\boldsymbol{M})$$

- **Computational:** *within* $O\big(\log \frac{1}{\varepsilon}\big)$ *iterations;*
- **Statistical:** *the sample complexity satisfies*

$$m \gtrsim (n_1 + n_2) r^2 \kappa^2.$$

Strict improvement over vanilla GD: provable acceleration at the same sample complexity!

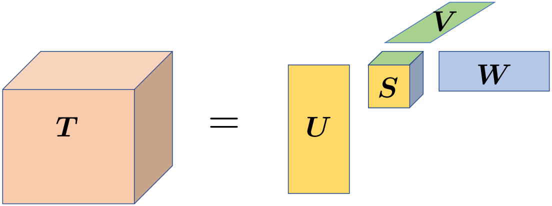
ScaledGD works more broadly



robust PCA

$$\begin{bmatrix} \checkmark & ? & ? & ? & \checkmark \\ ? & ? & \checkmark & \checkmark & ? \\ \checkmark & ? & ? & \checkmark & ? \\ ? & ? & \checkmark & ? & ? \\ \checkmark & ? & ? & ? & ? \\ ? & \checkmark & ? & ? & \checkmark \end{bmatrix}$$

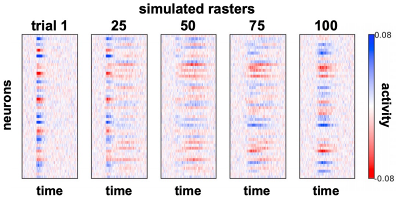
matrix completion



Tucker tensor recovery

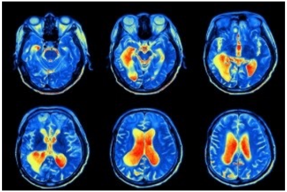
Huge computation savings at comparable sample complexities!

Generalization to tensors

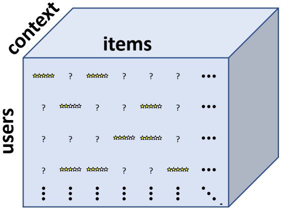


neural recordings

video surveillance



neuroimaging



★★★★★	?	★★★★★	?	?	?	•••
?	★★★★★	?	?	★★★★★	?	•••
?	?	?	★★★★★	★★★★★	?	•••
?	★★★★★	★★★★★	?	?	★★★★★	•••
⋮	⋮	⋮	⋮	⋮	⋮	⋱

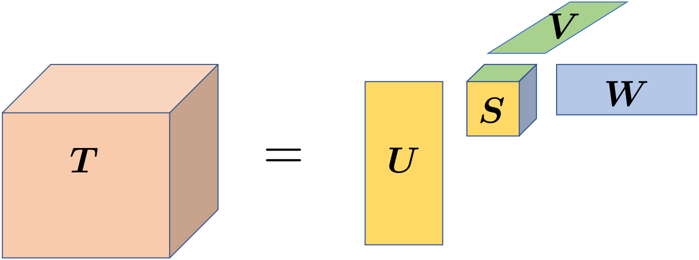
recommendation system

High-order tensors capture multi-way interactions across modalities.

Low-rank tensor under Tucker decomposition

Low-rank Tucker decomposition of a tensor:

$$\boldsymbol{T}(i_1, i_2, i_3) = \sum_{j_1, j_2, j_3} \boldsymbol{S}(j_1, j_2, j_3)\boldsymbol{U}(i_1, j_1)\boldsymbol{V}(i_2, j_2)\boldsymbol{W}(i_3, j_3)$$



$$\boldsymbol{T} = (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}) \cdot \boldsymbol{S},$$

where $\boldsymbol{U} \in \mathbb{R}^{n_1 \times r_1}$, $\boldsymbol{V} \in \mathbb{R}^{n_2 \times r_2}$, $\boldsymbol{W} \in \mathbb{R}^{n_3 \times r_3}$ and $\boldsymbol{S} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$.

Evidence that tensor problems are more challenging

Low-rank tensor recovery

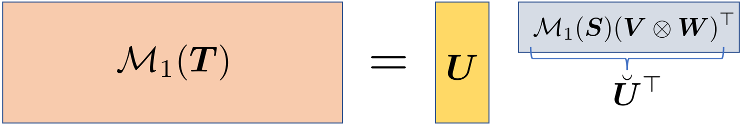
Recover low-rank $\boldsymbol{T}$ from $\boldsymbol{y} = \mathcal{A}(\boldsymbol{T})$.

- **Computation hardness:** the nuclear norm of a tensor is NP-hard to compute (Hillar and Lim, '13);
- **Computational barrier:** polynomial-time algorithm exists when the sample size is above $\Omega(n^{3/2})$ (Barak and Moitra, '16);
- **Little existing results for the Tucker case:** no provably efficient first-order algorithm for low-rank tensor completion (Han, Zhang, Willett, '20).

How to construct scaled gradients for tensors?

$$\min_{\boldsymbol{F}=(\boldsymbol{U},\boldsymbol{V},\boldsymbol{W},\boldsymbol{S})} \; f(\boldsymbol{F}) = \frac{1}{2}\left\|\mathcal{A}((\boldsymbol{U},\boldsymbol{V},\boldsymbol{W})\cdot\boldsymbol{S}) - \boldsymbol{y}\right\|_2^2$$

Step 1: unfolding the tensor along mode-1:



$$\mathcal{M}_1(\boldsymbol{T}) = \boldsymbol{U}\;\underbrace{\mathcal{M}_1(\boldsymbol{S})(\boldsymbol{V}\otimes\boldsymbol{W})^\top}_{\breve{\boldsymbol{U}}^\top}$$

Step 2: Treat this as a matrix problem for updating factor $\boldsymbol{U}$:

$$\boldsymbol{U}_{t+1} = \boldsymbol{U}_t - \eta\nabla_{\boldsymbol{U}} f(\boldsymbol{F}_t)\left(\breve{\boldsymbol{U}}_t^\top\breve{\boldsymbol{U}}_t\right)^{-1}$$

Step 3: update the core tensor $\boldsymbol{S}$:

$$\boldsymbol{S}_{t+1} = \boldsymbol{S}_t - \eta\Big((\boldsymbol{U}_t^\top\boldsymbol{U}_t)^{-1}, (\boldsymbol{V}_t^\top\boldsymbol{V}_t)^{-1}, (\boldsymbol{W}_t^\top\boldsymbol{W}_t)^{-1}\Big)\cdot\nabla_{\boldsymbol{S}} f(\boldsymbol{F}_t)$$

Key property: invariance to parameterization.

ScaledGD for low-rank tensor completion

Theorem (Tong et. al., JMLR 2022)

For low-rank tensor completion under Bernoulli sampling, assume $n = n_1 = n_2 = n_3$*, ScaledGD with spectral initialization and projection achieves*

$$\|(\boldsymbol{U}_t, \boldsymbol{V}_t, \boldsymbol{W}_t) \cdot \boldsymbol{S}_t - \boldsymbol{T}\|_{\mathrm{F}} \lesssim \varepsilon \cdot \sigma_{\min}(\boldsymbol{T})$$

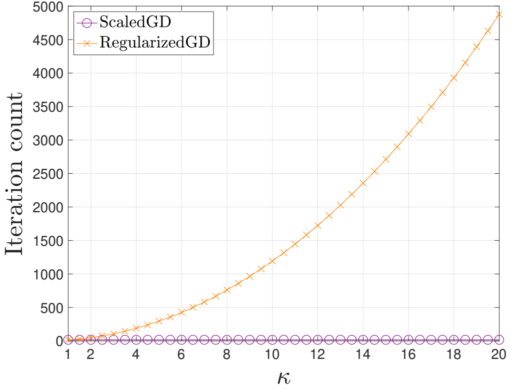
- **Computational:** *within* $O\big(\log \frac{1}{\varepsilon}\big)$ *iterations;*
- **Statistical:** *as long as the sample complexity satisfies*

$$n^3 p \gtrsim \mu^{3/2} r^{5/2} n^{3/2} \kappa^3 \log n.$$

First provable linear convergence at a near-optimal sample complexity for low-Tucker-rank tensor completion!

Numerical evidence

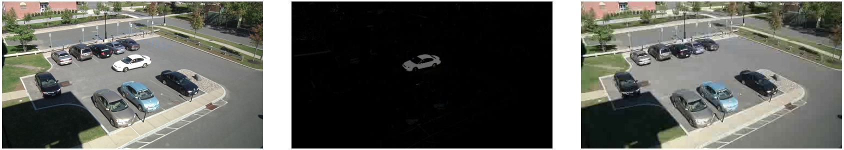
$$\min_{\boldsymbol{F}=(\boldsymbol{U},\boldsymbol{V},\boldsymbol{W},\boldsymbol{S})} \; f(\boldsymbol{F}) = \frac{1}{2}\left\|\mathcal{P}_{\Omega}((\boldsymbol{U},\boldsymbol{V},\boldsymbol{W})\cdot\boldsymbol{S}) - \boldsymbol{T})\right\|_{\mathrm{F}}^{2}$$



The benefit of ScaledGD is even more evident for tensors!

Tensor robust principal component analysis

Data $=$ Sparse $+$ Low-rank



Theorem (Dong et al., 2022)

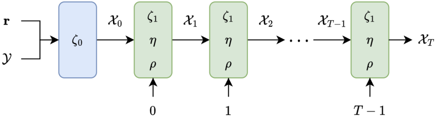
For a low-rank plus sparse tensor, ScaledGD with spectral initialization and iteration-varying thresholding converges at a constant rate, as long as the corruption level per fiber satisfies

$$\alpha \lesssim \frac{1}{\mu^2 r^3 \kappa}.$$

Can use selective mode updates to accelerate computation!

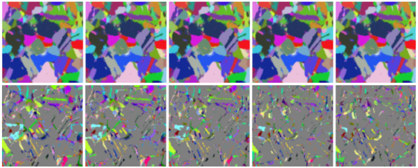
Unrolling for saliency detection in materials data

Unrolling ScaledGD + self-supervised learning for tensor RPCA



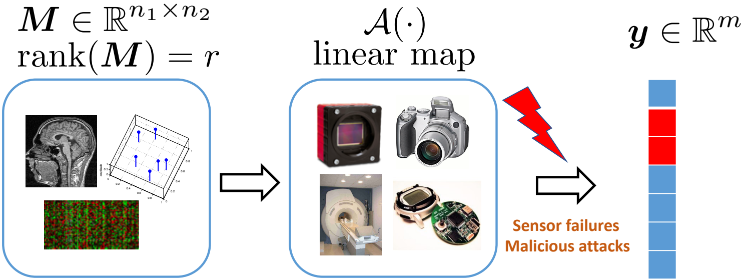
some materials data

low-rank + sparse decomposition



"Deep Unfolded Tensor Robust PCA with Self-supervised Learning", Dong, Shah, Donegan, and Chi, ICASSP 2023.

Robustness to outliers and corruptions?



$\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$
$\text{rank}(\boldsymbol{M}) = r$

$\mathcal{A}(\cdot)$
linear map

$\boldsymbol{y} \in \mathbb{R}^m$

Sensor failures
Malicious attacks

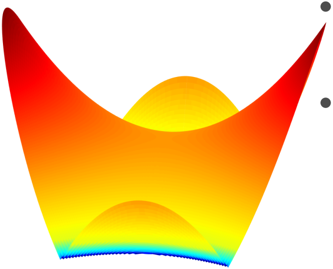
$$\boldsymbol{y} \;=\; \mathcal{A}(\boldsymbol{M}) + \underbrace{\boldsymbol{s}}_{\text{outliers}}, \quad \mathcal{A}(\boldsymbol{M}) = \{\langle \boldsymbol{A}_i, \boldsymbol{M} \rangle\}_{i=1}^m$$

Arbitrary but sparse outliers: $\|\boldsymbol{s}\|_0 \leq \alpha \cdot m$, where $0 \leq \alpha < 1$ is fraction of outliers.

Dealing with outliers: subgradient methods

Least absolute deviation (LAD):

$$\min_{\boldsymbol{X},\boldsymbol{Y}} \quad f(\boldsymbol{X},\boldsymbol{Y}) = \left\| \boldsymbol{y} - \mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top) \right\|_1$$



- **Median-truncated spectral initialization:** (Li et.al.'19).
- **Subgradient iterations:** (Charisopoulos et.al.'19; Li et al'18)

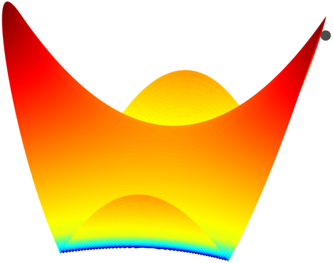
$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta_t \, \partial_{\boldsymbol{X}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t)$$
$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t - \eta_t \, \partial_{\boldsymbol{Y}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t)$$

Suffer from similar slow down due to ill-conditioning.

Dealing with outliers: scaled subgradient methods

Least absolute deviation (LAD):

$$\min_{\boldsymbol{X},\boldsymbol{Y}} \quad f(\boldsymbol{X},\boldsymbol{Y}) = \left\| \boldsymbol{y} - \mathcal{A}(\boldsymbol{X}\boldsymbol{Y}^\top) \right\|_1$$



- **Median-truncated spectral initialization:** (Li et.al.'19).
- **Scaled subgradient iterations:**

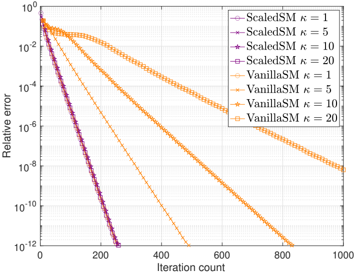
$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta_t \, \partial_{\boldsymbol{X}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t) \underbrace{(\boldsymbol{Y}_t^\top \boldsymbol{Y}_t)^{-1}}_{\texttt{preconditioner}}$$

$$\boldsymbol{Y}_{t+1} = \boldsymbol{Y}_t - \eta_t \, \partial_{\boldsymbol{Y}} f(\boldsymbol{X}_t, \boldsymbol{Y}_t) \underbrace{(\boldsymbol{X}_t^\top \boldsymbol{X}_t)^{-1}}_{\texttt{preconditioner}}$$

where η_t is set as Polyak's or geometric decaying stepsize.

Performance guarantees

	matrix sensing	**quadratic sensing**
Subgradient Method (Charisopoulos et al, '19)	$\frac{\kappa}{(1-2\alpha)^2}\log\frac{1}{\epsilon}$	$\frac{r\kappa}{(1-2\alpha)^2}\log\frac{1}{\epsilon}$
ScaledSM (Tong, Ma, Chi, TSP '21)	$\frac{1}{(1-2\alpha)^2}\log\frac{1}{\epsilon}$	$\frac{r}{(1-2\alpha)^2}\log\frac{1}{\epsilon}$



Robustness to both ill-conditioning and adversarial corruptions!

What if we do not know the exact rank?

So far we have assumed the exact rank is given.... what if we do not know the exact rank?

Misspecification by overparameterization:

$$\boldsymbol{M} = \boldsymbol{X}\boldsymbol{X}^\top, \qquad \boldsymbol{X} \in \mathbb{R}^{n\times r'}, \qquad r' > r$$

ScaledGD:(λ):

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta \nabla_{\boldsymbol{X}} f(\boldsymbol{X}_t) \underbrace{(\boldsymbol{X}_t^\top \boldsymbol{X}_t)^{-1}}_{\texttt{preconditioner}} \qquad \underbrace{(\boldsymbol{X}_t^\top \boldsymbol{X}_t + \lambda \boldsymbol{I})^{-1}}_{\texttt{preconditioner}}$$

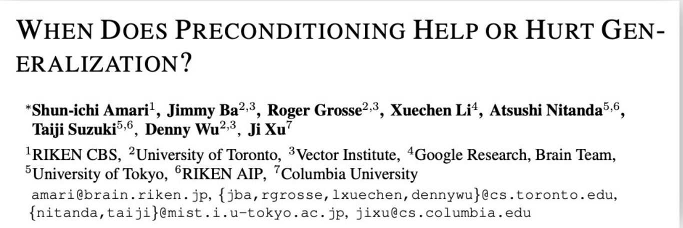
analysis break down and might be unstable...
add regularization to stabilize the preconditioner

Does preconditioning hurt generalization?

- Infinitely many global minima, not all generalize
- Can we still guarantee generalization?

optimization

generalization



WHEN DOES PRECONDITIONING HELP OR HURT GENERALIZATION?

*Shun-ichi Amari[1], Jimmy Ba[2,3], Roger Grosse[2,3], Xuechen Li[4], Atsushi Nitanda[5,6], Taiji Suzuki[5,6], Denny Wu[2,3], Ji Xu[7]

[1]RIKEN CBS, [2]University of Toronto, [3]Vector Institute, [4]Google Research, Brain Team, [5]University of Tokyo, [6]RIKEN AIP, [7]Columbia University

amari@brain.riken.jp, {jba,rgrosse,lxuechen,dennywu}@cs.toronto.edu, {nitanda,taiji}@mist.i.u-tokyo.ac.jp, jixu@cs.columbia.edu

Theoretical guarantees

Theorem (Xu, Shen, Ma, Chi, ICML 2023)

For low-rank matrix sensing with i.i.d. Gaussian design, ScaledGD(λ) *with* $\lambda \asymp \sigma_{\min}(\boldsymbol{M})$, $\eta \asymp 1$, *and small random initialization* $\boldsymbol{X}_0 \sim \alpha\mathcal{N}(0, 1/n)$ *with sufficiently small* α *achieves*

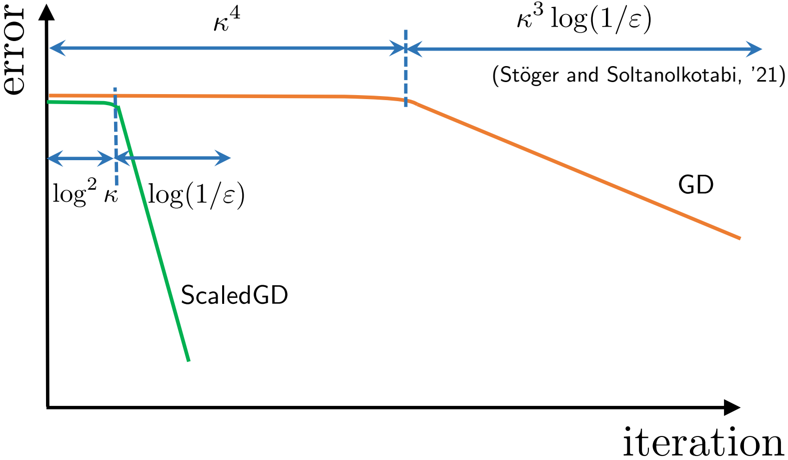
$$\|\boldsymbol{X}_t\boldsymbol{X}_t^\top - \boldsymbol{M}\|_{\mathrm{F}} \lesssim \varepsilon \cdot \sigma_{\min}(\boldsymbol{M})$$

- **Computational:** *within* $O\Big(\log\kappa\log(\kappa n) + \log\frac{1}{\varepsilon}\Big)$ *iterations;*
- **Statistical:** *the sample complexity satisfies*

$$m \gtrsim nr^2\textit{poly}(\kappa).$$

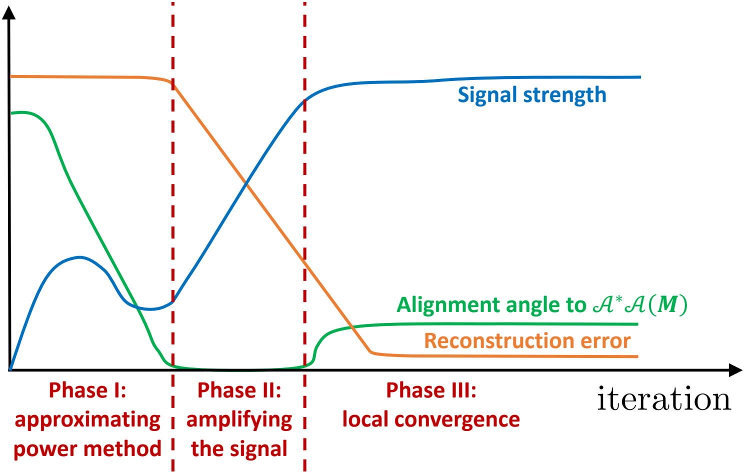
- Our analysis also enables exact convergence under random initialization with correct rank specification.

Comparison with overparameterized GD



ScaledGD picks up the signal component much faster than GD even from small random initialization!

A peek at the analysis: three-phased learning



Phase III: the reconstruction error decays exponentially with a constant rate

Near minimax-optimality

Noisy and approximately low-rank case:

$$y_i = \langle \boldsymbol{A}_i, \boldsymbol{M} \rangle + \xi_i, \quad \text{where} \quad \xi_i \sim \mathcal{N}(0, \sigma^2)$$

Theorem (Xu, Shen, Chi, Ma, '23)

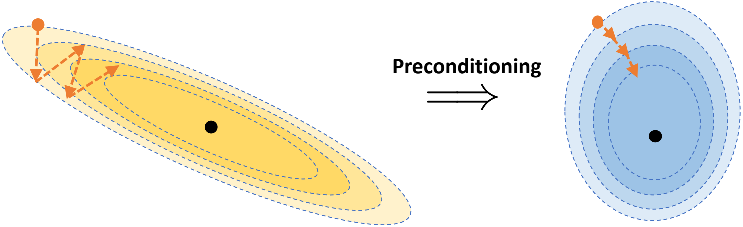
For low-rank matrix sensing with i.i.d. Gaussian design, and sufficiently small noise level, overparameterized ScaledGD(λ) with the same configuration as before achieves

$$\|\boldsymbol{X}_t \boldsymbol{X}_t^\top - \boldsymbol{M}\|_{\mathrm{F}} \lesssim \underbrace{\kappa^2 \sigma \sqrt{nr}}_{\text{noise}} + \underbrace{\kappa^2 \|\boldsymbol{M} - \boldsymbol{M}_r\|_{\mathrm{F}}}_{\text{approx. lowrank}},$$

where $\boldsymbol{M}_r$ is the best rank-r approximation of $\boldsymbol{M}$.

- first near minimax-optimal result up to κ^2;

Summary: preconditioning helps!

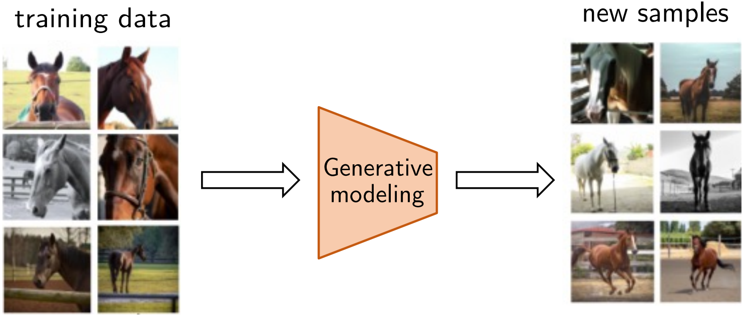


Preconditioning can dramatically increase the computational efficiency of vanilla gradient methods without hurting statistical efficiency

Part 2:

Towards demystifying score-based diffusion models for generation and inverse problems

Generative models



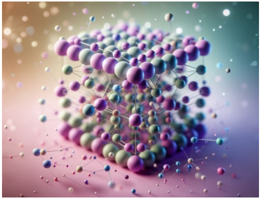
training data

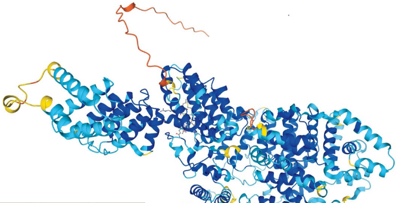
Generative modeling

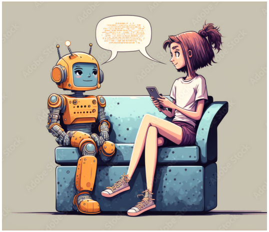
new samples

- Given training data $\underbrace{X^{\text{train},i} \sim p_{\text{data}}}_{\text{from a general distribution}}$ $(1 \leq i \leq N)$ in $\mathbb{R}^d$
- Generate new samples $Y \sim p_{\text{data}}$

From generative models to generative AI







Generative AI is transforming nearly every field of our society.

Approaching generative modeling via density estimation?

Suppose we to learn the distribution directly (parameterized by θ):

$$p_\theta(x) = \frac{e^{-f_\theta(x)}}{Z_\theta}$$

where Z_θ is a normalizing constant depending on θ.

- Use maximum likelihood to estimate θ,

  $$\max_\theta \sum_{i=1}^{N} \log p_\theta(x_i)$$

  and then sample from the learned $p_\theta(x)$.
- Intractable! Why?

Score function is all you need

The **(Stein's) score function** of a distribution $p(x)$ is defined as

$$s(x) = \nabla_x \log p(x).$$

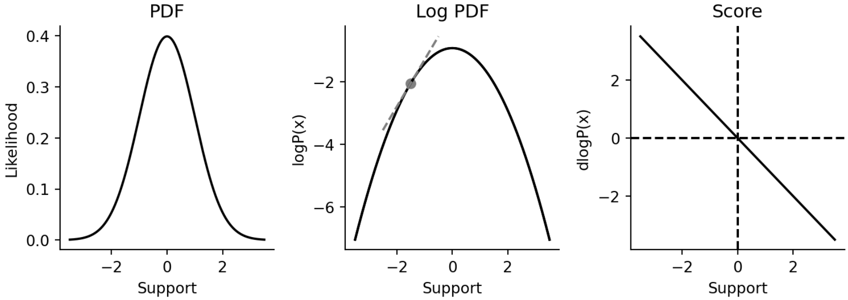
Charles Stein

Note that

$$\begin{aligned} s(x) &= \nabla_x \log \frac{e^{-f_\theta(x)}}{Z_\theta} \\ &= -\nabla_x f_\theta(x) - \nabla_x \log Z_\theta = -\nabla_x f_\theta(x) \end{aligned}$$

getting rid of the annoying Z_θ!

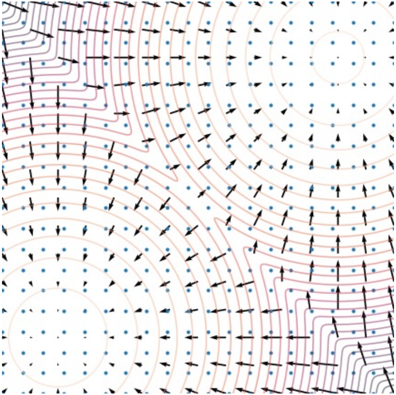
Score function of Gaussian distribution



(Figure credit: internet)

The score function points towards regions of higher probability.

Score function of Gaussian mixtures



(Figure credit: internet)

The score function points towards regions of higher probability.

Score function is all you need: Langevin dynamics

Unadjusted Langevin algorithm (ULA): from some x_0, perform iterative sampling

$$x_{t+1} = x_t + \eta s(x_t) + \sqrt{2\eta} z_t,$$

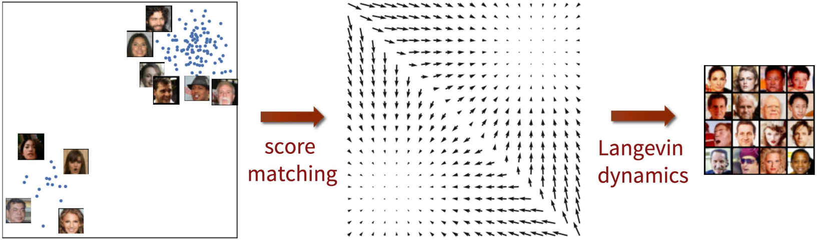
where $z_t \sim \mathcal{N}(0, I)$ and η is some learning rate.

- In continuous-time, ULA recovers the Langevin dynamic:

$$dX_\tau = -\nabla f(X_\tau) d\tau + \sqrt{2} dB_\tau$$

- When $\eta \to 0$, x_t converges to a sample from $p(x)$ under some regularity conditions.
- Only needs the score function to sample.

Score-based generative model via Langevin dynamics



score matching

Langevin dynamics

Data samples

$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p(\mathbf{x})$

Scores

$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$

New samples

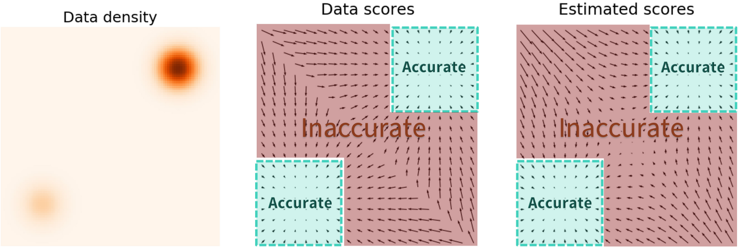
(Figure credit: Y. Song)

Dismay performance in practice. Why?

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

Manifold hypothesis

- Real-world data live on low-dimensional manifold.
- Reliable score estimation is available only in high-density regions.
- However, our initial sample is highly likely in low density regions (where score estimates are poor).

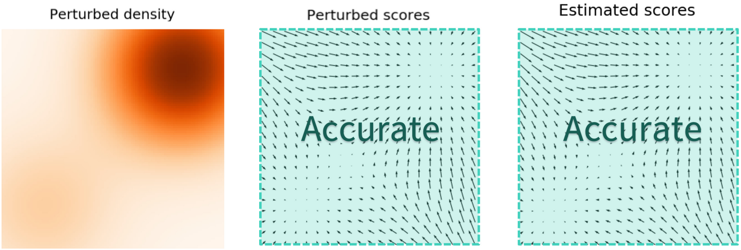


(Figure credit: Y. Song)

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

Adding noise to data

- To improve data coverage (and score estimation), we can add noise to it.
- However, this makes the data distribution different from what we want.



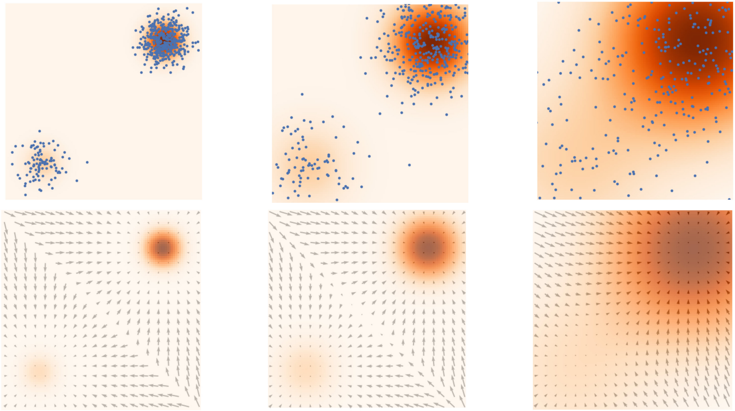
(Figure credit: Y. Song)

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

Key idea: noise annealing

Annealing: introducing data perturbation at multiple noise levels.

$$\sigma_1 < \sigma_2 < \sigma_3$$



(Figure credit: Y. Song)

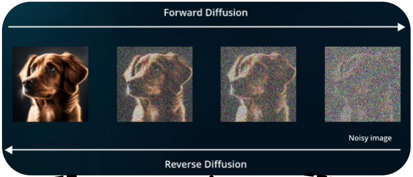
`https://yang-song.net/blog/2021/score/`

State-of-the-art diffusion models

Inspired by nonequilibrium thermodynamics

— *Sohl-Dickstein, Weiss, Maheswaranathan, Ganguli '15*

Diffusion models

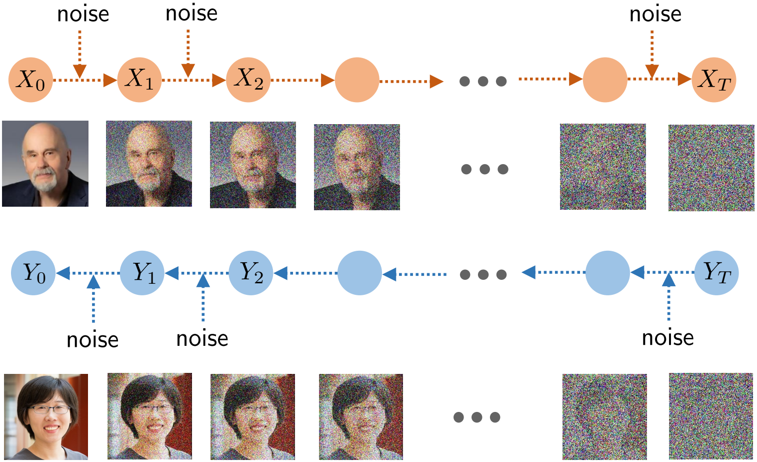


Stable Diffusion

DALLE

Sora

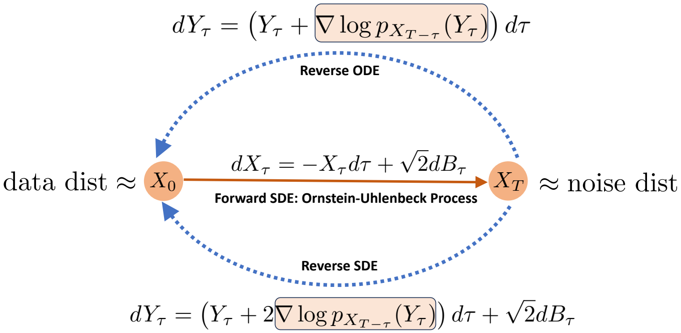
A high-level description of diffusion models



- **forward process (training):** (progressively) diffuse data into noise
- **reverse process (sampling):** convert pure noise into desired data

How to learn a reverse process s.t. $Y_t \overset{\mathrm{d}}{\approx} X_t$ $(1 \le t \le T)$?

It is feasible as long as one knows the score function (Anderson'82; Haussmann and Pardoux'86; Song et el.'20)...



$$dY_\tau = \left(Y_\tau + \boxed{\nabla \log p_{X_{T-\tau}}(Y_\tau)}\right) d\tau$$

Reverse ODE

$$dX_\tau = -X_\tau d\tau + \sqrt{2} dB_\tau$$

$$\text{data dist} \approx X_0 \longrightarrow X_T \approx \text{noise dist}$$

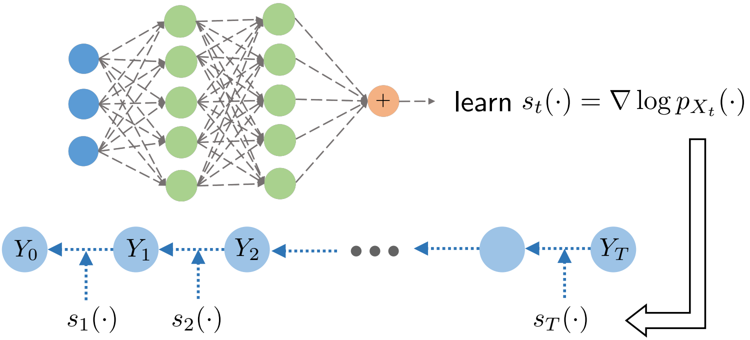
Forward SDE: Ornstein-Uhlenbeck Process

Reverse SDE

$$dY_\tau = \left(Y_\tau + 2\boxed{\nabla \log p_{X_{T-\tau}}(Y_\tau)}\right) d\tau + \sqrt{2} dB_\tau$$

Score is all you need

- **score functions** of marginals of forward process: $\underbrace{\nabla \log p_{X_t}(X)}_{\text{w.r.t. } X}$



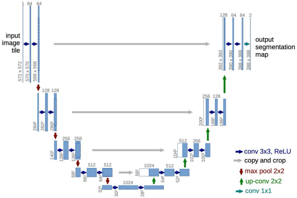
1. **score learning/matching:** learn estimates $s_t(\cdot)$ for $\nabla \log p_{X_t}(\cdot)$
2. **data generation:** sampling w/ the aid of score estimates $\{s_t(\cdot)\}$

Score matching via denoising

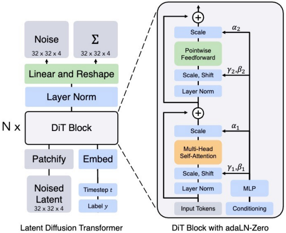
$$X_0 \sim p_{\text{data}}, \quad X_t = \sqrt{\bar{\alpha}_t} X_0 + \sqrt{1-\bar{\alpha}_t}\,\mathcal{N}(0, I_d)$$

Tweedie's formula (Hyvarinen, 2005; Vincent, 2011):

$$s_t^\star(x) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \underbrace{\mathbb{E}_{x_0 \sim p_{\text{data}},\, \epsilon_t \sim \mathcal{N}(0, I_d)} \left[\epsilon_t \,\middle|\, \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t = x \right]}_{\text{MMSE denoising}} .$$



U-Net
[Ronneberger, Fischer, Brox, 2015]



Diffusion Transformers
[Peebles and Xie, 2022]

From score networks to downstream tasks

✄ decouple

score learning ← ✗ → $\underbrace{\text{downstream tasks}}_{\textbf{our focus}}$

Sampling (unconditional generation):

When and how fast do stochastic/deterministic samplers converge?

Acceleration:

Can we accelerate the convergence of stochastic and deterministic diffusion samplers provably?

Inverse problems (conditional generation):

Can we design provably robust posterior samplers using unconditional diffusion priors?

Non-asymptotic convergence for diffusion-based generative models

Two mainstream approaches

— Ho, Jain, Abbeel '20

$$X_0 \sim p_{\mathsf{data}}, \quad X_t = \sqrt{1-\beta_t}X_{t-1} + \sqrt{\beta_t}\mathcal{N}(0, I_d), \quad 1 \le t \le T$$

1. A stochastic sampler: $\underbrace{\textbf{denoising diffusion probabilistic models}}_{\text{DDPM}}$

$$Y_T \sim \mathcal{N}(0, I_d)$$
$$Y_{t-1} = \underbrace{\frac{1}{\sqrt{1-\beta_t}}\Big(Y_t + \beta_t s_t(Y_t)\Big)}_{\text{deterministic component}} + \underbrace{\sqrt{\beta_t}\mathcal{N}(0, I_d)}_{\text{random component}}, \quad t = T, \cdots, 1$$

Probability flow ODE

— *Song, Sohl-Dickstein, Kingma, Kumar, Ermon, Poole '20*

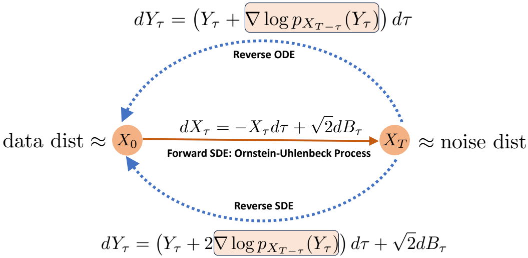
$$X_0 \sim p_{\mathsf{data}}, \quad X_t = \sqrt{1-\beta_t}X_{t-1} + \sqrt{\beta_t}\mathcal{N}(0, I_d), \quad 1 \leq t \leq T$$

2. A <u>deterministic</u> sampler based on **probability flow ODE**

$$\begin{aligned} Y_T &\sim \mathcal{N}(0, I_d) \\ Y_{t-1} &= \underbrace{\frac{1}{\sqrt{1-\beta_t}}\left(Y_t + \frac{\beta_t}{2} s_t(Y_t)\right)}_{\text{purely deterministic}}, \qquad t = T, \cdots, 1 \end{aligned}$$

Towards understanding the non-asymptotic convergence

Question: can we understand non-asymptotic convergence of diffusion models in discrete time?



$$dY_\tau = \left(Y_\tau + \boxed{\nabla \log p_{X_{T-\tau}}(Y_\tau)}\right) d\tau$$

Reverse ODE

$$dX_\tau = -X_\tau d\tau + \sqrt{2} dB_\tau$$

data dist $\approx X_0$ → $X_T \approx$ noise dist

Forward SDE: Ornstein-Uhlenbeck Process

Reverse SDE

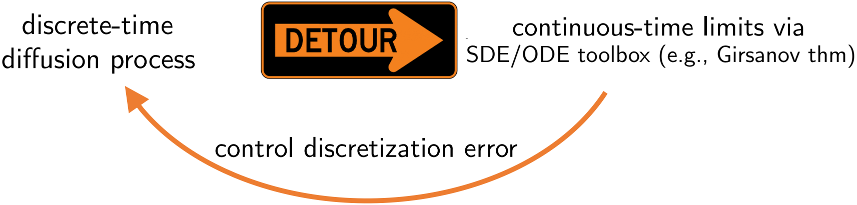
$$dY_\tau = \left(Y_\tau + 2\boxed{\nabla \log p_{X_{T-\tau}}(Y_\tau)}\right) d\tau + \sqrt{2} dB_\tau$$

Sources of errors:

- initialization error (dealing with the gap between X_T and Y_T)
- discretization error.
- score estimation error

Prior approaches

— Li, Lu, Tan '22
— Chen, Lee, Lu '22
— Chen, Chewi, Li, Li, Salim, Zhang '22
— Chen, Daras, Dimakis '23
— Chen, Chewi, Lee, Li, Lu, Salim '23



discrete-time diffusion process

continuous-time limits via SDE/ODE toolbox (e.g., Girsanov thm)

control discretization error

- Built upon toolboxes from SDE/ODE
- Existing analyses were **inadequate for deterministic samplers**

This work: a non-asymptotic framework that analyzes discrete-time processes directly + accommodates deterministic samplers

Assumption on target data distribution

- **Minimal data assumptions:**

$$\mathbb{P}(\|X_0\|_2 \leq T^{c_R}) = 1$$

for arbitrarily large constant $c_R > 0$

- **learning rates:** for some large constants $c_0, c_1 > 0$,

$$\beta_1 = \frac{1}{T^{c_0}}$$
$$\beta_t = \frac{c_1 \log T}{T} \min\left\{\beta_1\left(1 + \frac{c_1 \log T}{T}\right)^t, \, 1\right\}$$

Non-asymptotic complexity of generation

Theorem (Li, Wei, Chen, Chi, ICLR 2024)

Suppose we have **perfect scores:** $s_t(\cdot) = \nabla \log p_{X_t}(\cdot)$ *for all* t*.*

- *For the deterministic sampler (DDIM-type/prob. flow ODE),*

$$\mathsf{TV}\big(p_{X_1}, p_{Y_1}\big) \lesssim \frac{d^2}{T} \qquad \textit{up to log factor.}$$

- *For the stochastic sampler (DDPM-type),*

$$\mathsf{TV}\big(p_{X_1}, p_{Y_1}\big) \lesssim \frac{d^2}{\sqrt{T}} \qquad \textit{up to log factor.}$$

- *first* polynomial-time bounds for *plain* probability flow ODE
- The deterministic samplers are faster than the stochastic ones.

Assumption on score estimation error

- ℓ_2 error: score function estimate obeys

$$\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{X\sim q_t}\left[\left\|s_t(X)-s_t^\star(X)\right\|_2^2\right]\le\varepsilon_{\mathsf{score}}^2.$$

Needed for both stochastic and deterministic samplers

- Jacobian error: denote by $J_{s_t^\star}=\frac{\partial s_t^\star}{\partial x}$ and $J_{s_t}=\frac{\partial s_t}{\partial x}$ the Jacobian matrices of $s_t^\star(\cdot)$ and $s_t(\cdot)$, which obey

$$\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{X\sim q_t}\left[\left\|J_{s_t}(X)-J_{s_t^\star}(X)\right\|\right]\le\varepsilon_{\mathsf{Jacobi}}.$$

Needed for deterministic samplers (counterexamples exist)

Non-asymptotic rates with score estimation errors

Theorem (Li, Wei, Chen, Chi, ICLR 2024)

With score estimation errors,

- *For the deterministic sampler (DDIM-type/prob. flow ODE),*

$$\mathsf{TV}\big(q_1, p_1\big) \lesssim \frac{d^2}{T} + \sqrt{d}\varepsilon_{\mathsf{score}} + d\varepsilon_{\mathsf{Jacobi}} \quad \textit{up to log factor.}$$

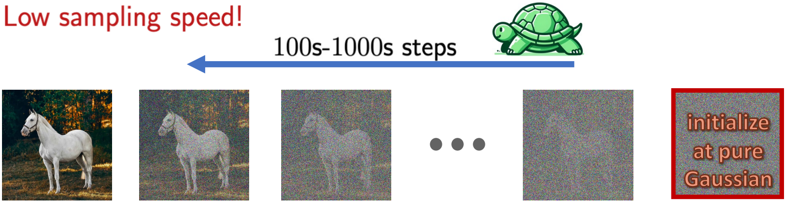
- *For the stochastic sampler (DDPM-type),*

$$\mathsf{TV}\big(q_1, p_1\big) \lesssim \frac{d^2}{\sqrt{T}} + \sqrt{d}\varepsilon_{\mathsf{score}} \qquad \textit{up to log factor.}$$

- The dependency with d can be improved to d: see (Benton et al, 2024) for the stochastic sampler, and (Li et al., 2024) for the deterministic sampler.

Fast convergence for general data distribution, as long as we have good score estimates.

Acceleration?



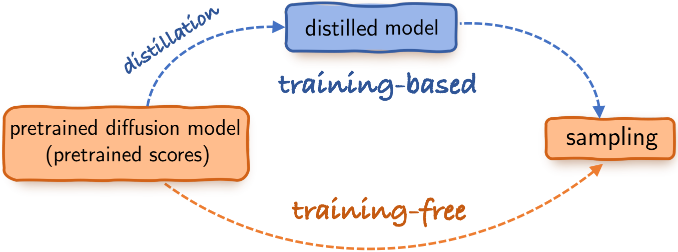
Low sampling speed!

100s-1000s steps

initialize at pure Gaussian

50k images: DDPM (20h) *vs.* single-step GANs ($<$ 1min)

Acceleration?



distillation

distilled model

training-based

pretrained diffusion model (pretrained scores)

sampling

training-free

- **Training-based methods:** progressive distillation (Salimans et al., 2022), consistency model (Song et al., 2023)...

  additional training steps are required

- **Training-free methods:** DPM-Solver/++ (Lu et al., 2022ab), UniPC (Zhao et al., 2023)...

Can we develop training-free deterministic (resp. stochastic) samplers that converge provably faster than DDIM (resp. DDPM)?

Acceleration of DDIM via high-order ODE discretization

Solving the probability flow ODE ($\overline{\alpha}_t := \prod_{k=1}^{t} \alpha_k$ with $\alpha_t = 1 - \beta_t$):

$$X(\overline{\alpha}_{t-1}) = \frac{1}{\sqrt{\alpha_t}} X(\overline{\alpha}_t) + \frac{\sqrt{\overline{\alpha}_{t-1}}}{2} \int_{\overline{\alpha}_t}^{\overline{\alpha}_{t-1}} \frac{1}{\sqrt{\gamma^3}} \underbrace{s^{\star}_{\gamma}\big(X(\gamma)\big)}_{\text{approximated by?}} \mathrm{d}\gamma$$

Scheme 1: $s^{\star}_{\gamma}\big(X(\gamma)\big) \approx s^{\star}_{\overline{\alpha}_t}\big(X(\overline{\alpha}_t)\big) \approx s_t(X_t)$

$$X(\overline{\alpha}_{t-1}) \approx \frac{1}{\sqrt{\alpha_t}}\Big(X(\overline{\alpha}_t) + \frac{1-\alpha_t}{2} s_t(X_t)\Big) \quad \text{original DDIM}$$

Scheme 2: $s^{\star}_{\gamma}\big(X(\gamma)\big) \approx s_t(X_t) + \frac{\gamma - \overline{\alpha}_t}{\overline{\alpha}_t - \overline{\alpha}_{t+1}} \big(s_t(X_t) - s_{t+1}(X_{t+1})\big)$

$$X(\overline{\alpha}_{t-1}) \approx \frac{1}{\sqrt{\alpha_t}} \left(X(\overline{\alpha}_t) + \frac{1-\alpha_t}{2} s_t(X_t) \right) + \frac{1}{\sqrt{\alpha_t}} \left(\frac{(1-\alpha_t)^2}{4(1-\alpha_{t+1})} \Big(s_t(X_t) - \sqrt{\alpha_{t+1}} s_{t+1}(X_{t+1}) \Big) \right) \quad \textbf{Ours}$$

DPM-Solver-2 (Lu et al, 2022a): to construct second-order ODE solver

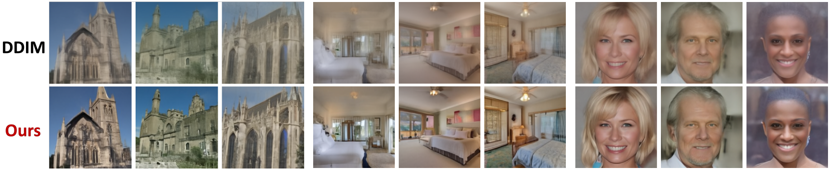
Non-asymptotic rate of accelerated deterministic sampler

Theorem (Li et al. 2024, informal)

The accelerated deterministic sampler obeys

$$\mathsf{TV}\left(p_{X_1}, p_{Y_1}\right) \lesssim \frac{d^6}{T^2} + \sqrt{d}\varepsilon_{\mathsf{score}} + d\varepsilon_{\mathsf{Jacobi}}$$

- Improved rate $\widetilde{O}(1/T^2)$ compared with vanilla DDIM $\widetilde{O}(1/T)$



Sampled images with 5 NFEs: crisper and less noisy

Acceleration of DDPM via higher-order approximation

Characterizing $p_{X_{t-1}|X_t}$:

$$p_{X_{t-1}|X_t=x_t} \approx \mathcal{N}\left(\mu_t^\star(x_t), \Sigma_t^\star(x_t)\right)$$

- $\mu_t^\star\left(x_t\right) := \frac{1}{\sqrt{\alpha_t}}\left(x_t + \left(1-\alpha_t\right)s_t^\star\left(x_t\right)\right)$
- $\Sigma_t^\star(x_t) = (1-\alpha_t)\underbrace{\Big(I + \frac{1-\alpha_t}{2}J_t^\star\left(x_t\right)\Big)\Big(I + \frac{1-\alpha_t}{2}J_t^\star\left(x_t\right)\Big)^\top}_{\text{simple approximation } I \text{ in DDPM analysis}}$

Constructing $p_{Y_{t-1}|Y_t=x_t} \approx \mathcal{N}(\mu_t^\star(x_t), \Sigma_t^\star(x_t))$:

$$Y_{t-1} = \frac{1}{\sqrt{\alpha_t}}\Bigg(\underbrace{Y_t + \sqrt{\tfrac{1-\alpha_t}{2}}Z_t}_{=:\Phi(Y_t,Z_t)} + \sqrt{\tfrac{1-\alpha_t}{2}}Z_t^+ \quad \text{applying DDPM at } \Phi(Y_t, Z_t)$$

$$+ (1-\alpha_t)\underbrace{\Big(s_t^\star(Y_t) - \sqrt{\tfrac{(1-\alpha_t)}{2}}J_t^\star(Y_t)Z_t\Big)}_{\approx s_t^\star\left(\Phi(Y_t,Z_t)\right)}\Bigg) \quad \textbf{Ours}$$

Non-asymptotic rate of accelerated stochastic sampler

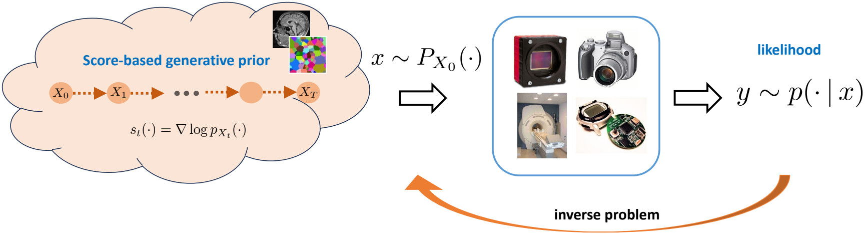
Theorem (Li et al. 2024, informal)

The accelerated stochastic sampler obeys

$$\mathsf{TV}\big(p_{X_1}, p_{Y_1}\big) \lesssim \frac{d^3}{T} + \sqrt{d}\varepsilon_{\mathsf{score}}.$$

- Improved rate $\widetilde{O}(1/T)$ compared with vanilla DDPM $\widetilde{O}(1/\sqrt{T})$
- ℓ_2 score error assumption suffices (no need of Jacobian errors)

Score-based diffusion model for inverse problems



Posterior sampling: sample from

$$p(\cdot|y) \propto p(\cdot)\,p(y\,|\,x) = \underbrace{p(\cdot)}_{\texttt{prior}}\ \exp\underbrace{(\mathcal{L}(\cdot\,;\,y))}_{\texttt{log-likelihood}}$$

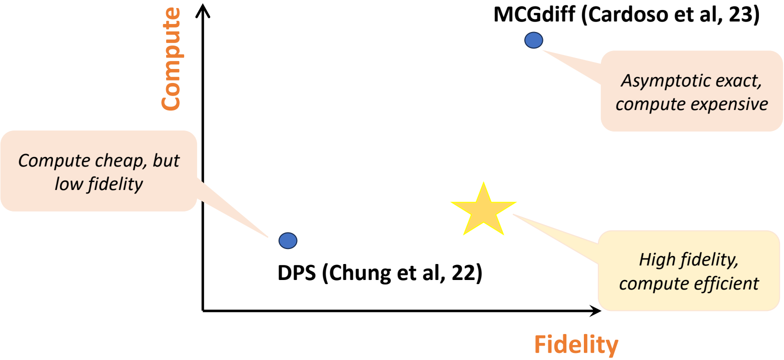
Score-based implicit prior: the data prior $p(\cdot)$ is accessed through its *unconditional* score functions $s_t(\cdot) = \nabla \log p_{X_t}(\cdot)$.

A highly incomplete list of prior work

- (Song et al., 2021)
- (Laumont et al., 2022)
- (Kawar et al., 2022)
- (Trippe et al., 2022)
- (Graikos et al., 2022)
- (Chung et al., 2023)
- (Cardoso et al., 2023)
- (Song et al., 2023)
- (Mardani et al., 2023)
- (Feng et al., 2023)
- (Chen et al., 2023)
- (Coeurdoux et al., 2023)
- (Wu et al., 2022)
- (Dou and Song, 2024)
- ...

Majority of the existing algorithms are heuristic and/or tailored to linear inverse problems.

Towards provably efficient and accurate inversion


Compute
MCGdiff (Cardoso et al, 23)
Asymptotic exact, compute expensive
Compute cheap, but low fidelity
DPS (Chung et al, 22)
High fidelity, compute efficient
Fidelity

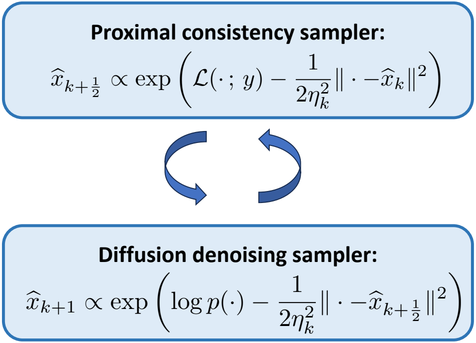
Goal: develop provably compute-efficient and high-fidelity diffusion-based inversion methods for arbitrary forward model.

Our approach: diffusion plug-and-play (DPnP)

Inspired by (Bouman and Buzzard, 2023; Vono et al., 2019; Lee et al., 2021)

$$p(\cdot|y) \propto \exp\Big(\log p(\cdot) + \mathcal{L}(\cdot\,;\,y)\Big)$$

Given an <u>annealing schedule</u> $\{\eta_k\}$,



Proximal consistency sampler:

$$\widehat{x}_{k+\frac{1}{2}} \propto \exp\left(\mathcal{L}(\cdot\,;\,y) - \frac{1}{2\eta_k^2}\|\cdot - \widehat{x}_k\|^2\right)$$

Diffusion denoising sampler:

$$\widehat{x}_{k+1} \propto \exp\left(\log p(\cdot) - \frac{1}{2\eta_k^2}\|\cdot - \widehat{x}_{k+\frac{1}{2}}\|^2\right)$$

✓ Readily implementable by, e.g., MALA

? How do we implement this step using diffusion score functions?

Diffusion denoising sampler

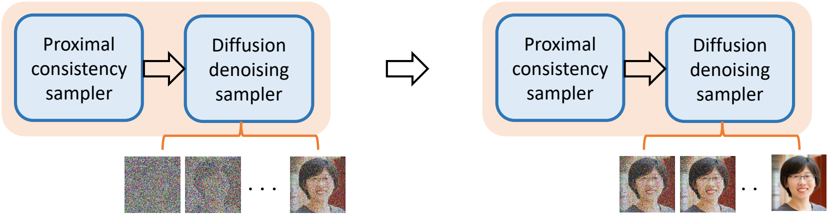
Posterior sampling for AWGN denoising:

$$\exp\left(\log p(x) - \frac{1}{2\eta_k^2}\|x - \widehat{x}_{k+\frac{1}{2}}\|^2)\right) \propto p(x^\star \,|\, x^\star + \eta_k w = \widehat{x}_{k+\frac{1}{2}})$$

where $w \sim \mathcal{N}(0, I_d)$.

- Key insight: this can be solved by diffusion!
    - stochastic/deterministic samplers via reversing properly defined forward processes (e.g., heat flow or Ornstein-Uhlenbeck process), whose score functions can be mapped from $s_t(\cdot)$.
- The resulting update rules are similar to, but not the same as, the ones used for generation.

Schematic view of DPnP


Proximal consistency sampler
Diffusion denoising sampler
. . .
Proximal consistency sampler
Diffusion denoising sampler
. .

- Each iteration of DPnP contains a "full" reverse denoising process with multiple denoising steps.
- But, it can be easily combined with acceleration schemes, such as distillation, to speed up.

Our theory

Theorem (Xu and Chi, 2024)

Set constant $\eta_k = \eta > 0$. *Define a stationary distribution* π_η *by*

$$\pi_\eta(x) \propto p(x) q_\eta(x), \qquad q_\eta(x) = \mathrm{e}^{\mathcal{L}(\cdot;\, y)} * p_{\eta\epsilon}(x),$$

where $\epsilon \sim \mathcal{N}(0, I_d)$ *and* $*$ *denotes convolution. There exists* $\lambda := \lambda(p, \mathcal{L}, \eta) \in (0,1)$, *such that for any accuracy level* $\epsilon > 0$, *with* $K \asymp \frac{1}{1-\lambda} \log(1/\epsilon)$, *we have*

$$\mathsf{TV}(p_{\widehat{x}_K}, \pi_\eta) \lesssim \underbrace{\epsilon \sqrt{\chi^2(p_{\widehat{x}_1} \,\|\, \pi_\eta)}}_{\text{init error}} + \underbrace{\frac{1}{1-\lambda}(\epsilon_{\mathsf{DDS}} + \epsilon_{\mathsf{PCS}}) \log\left(\frac{1}{\epsilon}\right)}_{\text{sampler error}},$$

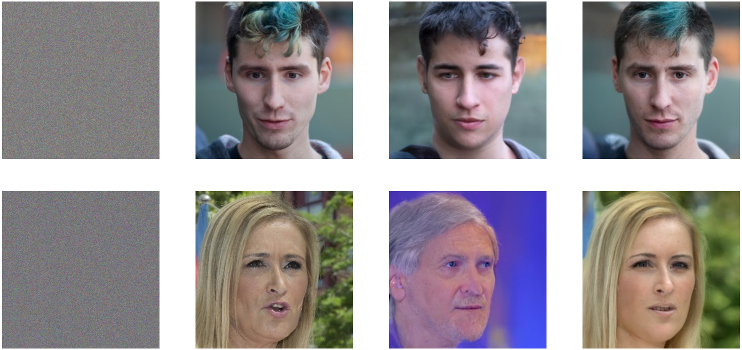
where ϵ_{PCS} *and* ϵ_{DDS} *are the total variation error of* PCS *and* DDS.

- A diminishing schedule $\{\eta_k\}$ ensures asymptotic consistency.

DPnP is the first provably-robust posterior sampling method for nonlinear inverse problems using unconditional diffusion priors.

Numerical experiments

Phase retrieval: recover an unknown image from the magnitude of its masked Fourier transform.



Observation

Ground truth

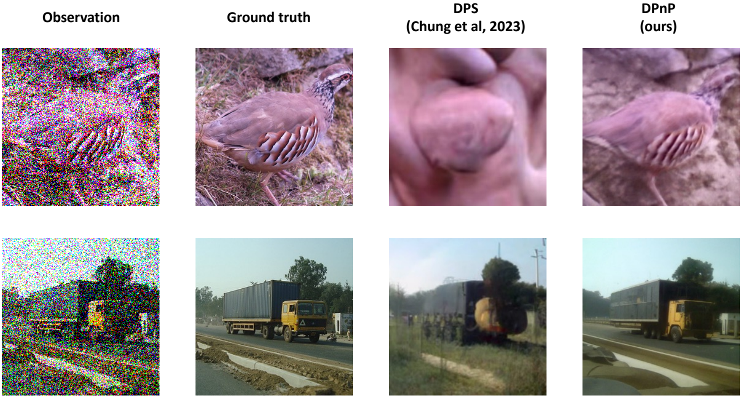
DPS (Chung et al, 2023)

DPnP (ours)

DPnP recovers the fine-grained details more faithfully.

Numerical experiments

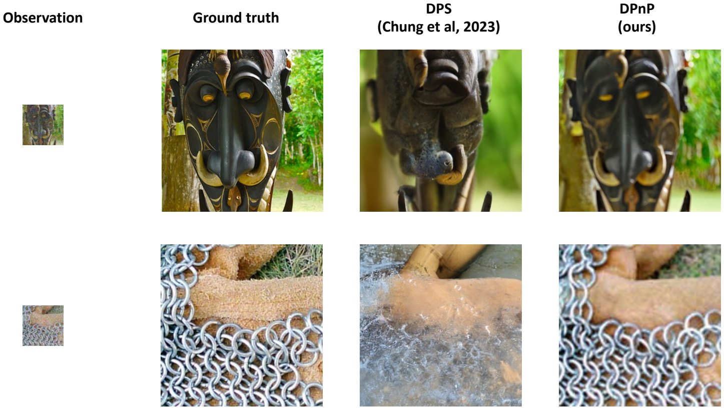
Quantized sensing: recover an unknown image from its one-bit dithered measurements.



DPnP recovers the fine-grained details more faithfully.

Numerical experiments

Super resolution: recover an unknown image from its 4x downsampled version.



DPnP recovers the fine-grained details more faithfully.

More metrics

Table: Performance on the ImageNet 256×256 validation dataset.

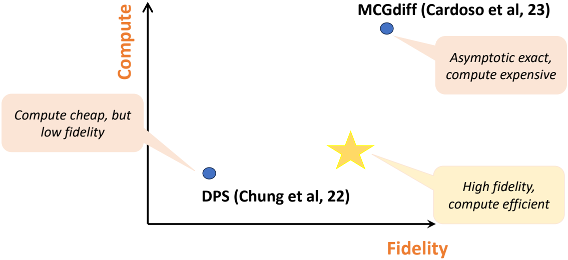
	Super-resolution (4x, linear)		Phase retrieval (nonlinear)		Quantized sensing (nonlinear)		Time per sample
Algorithm	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	
DPnP-DDIM (ours)	**0.416**	**21.6**	**0.562**	**13.4**	**0.363**	**23.0**	$\sim$ 240s
DPS	0.473	20.2	0.677	**13.4**	0.542	18.7	$\sim$ 150s
LGD-MC ($n = 5$)	**0.416**	20.9	0.592	12.8	0.384	22.3	$\sim$ 150s

Table: Performance on the FFHQ 256×256 validation dataset.

	Super-resolution (4x, linear)		Phase retrieval (nonlinear)		Quantized sensing (nonlinear)		Time per sample
Algorithm	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	
DPnP-DDIM (ours)	**0.301**	**24.2**	**0.376**	**22.4**	**0.293**	**24.2**	$\sim$ 90s
DPS	0.331	23.1	0.490	17.4	0.367	21.7	$\sim$ 60s
LGD-MC ($n = 5$)	0.318	23.9	0.522	16.4	0.317	23.9	$\sim$ 60s

DPnP achieves better performance with a bit more compute.

Summary: diffusion models


Compute
MCGdiff (Cardoso et al, 23)
Asymptotic exact, compute expensive
Compute cheap, but low fidelity
DPS (Chung et al, 22)
High fidelity, compute efficient
Fidelity

Diffusion models are showing great promise in generative AI for Science.

Selected references: nonconvex low-rank estimation

1. Nonconvex Optimization Meets Low-Rank Matrix Factorization: An Overview, Y. Chi, Y. M. Lu and Y. Chen, *IEEE Trans. on Signal Processing*, 2019.
2. Spectral Methods for Data Science: A Statistical Perspective, Y. Chen, Y. Chi, J. Fan and C. Ma, *Foundations and Trends in Machine Learning*, 2021.
3. Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent, T. Tong, C. Ma, and Y. Chi, *Journal of Machine Learning Research*, 2021.
4. Scaling and scalability: Provable nonconvex low-rank tensor estimation from incomplete measurements, T. Tong, C. Ma, A. Prater-Bennette, E. Tripp, and Y. Chi, *Journal of Machine Learning Research*, 2022.
5. Low-rank matrix recovery with scaled subgradient methods: Fast and robust convergence without the condition number, T. Tong, C. Ma, and Y. Chi, *IEEE Trans. on Signal Processing*, 2021.
6. The power of preconditioning in overparameterized low-rank matrix sensing, X Xu, Y Shen, Y Chi, and C Ma, *ICML*, 2023.

Selected references: diffusion models

1. Score-Based Generative Modeling through Stochastic Differential Equations, Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, *ICLR*, 2021.
2. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions, S. Chen, S. Chewi, J. Li, Y. Li, A. Salim, and A. Zhang, *ICLR*, 2023.
3. Towards Non-Asymptotic Convergence for Diffusion-Based Generative Models, G. Li, Y. Wei, Y. Chen and Y. Chi, *ICLR*, 2024.
4. Accelerating Convergence of Score-Based Diffusion Models, Provably, G. Li, Y. Huang, T. Efimov, Y. Wei, Y. Chi and Y. Chen, *ICML*, 2024.
5. Provably Robust Score-Based Diffusion Posterior Sampling for Plug-and-Play Image Reconstruction, X. Xu and Y. Chi, *preprint*, 2024.

Thanks!

https://users.ece.cmu.edu/~yuejiec/