

Learning Low-Dimensional Distributions via Denoising

Druv Pai

UC Berkeley



Lectures so Far

- **History** of the pursuit and study of intelligence.
- **Analytic solutions** for learning low-dimensional linear/Gaussian mixtures via (unrolled) optimization.

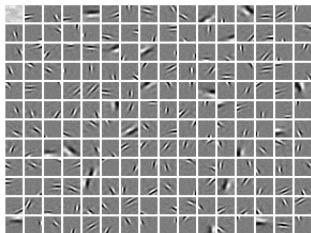
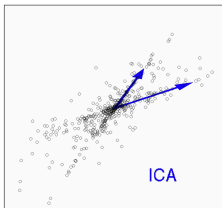
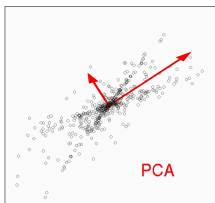
This lecture: denoising!

Learning and sampling from general low-dimensional distributions via learning to remove noise.

Recall: Classical Approaches

Assume: data drawn from a well-known template distribution:

- A single subspace (PCA)
- A union of a few subspaces (ICA)
- Sparsely generated from a dictionary (DL)



What about *general data distributions*?

Entropy

Goal of learning: *capture* and *encode* the low-dimensional structures of the data distributions from finite samples.

Entropy: a standard complexity measure of distributions.

$$h(\boldsymbol{x}) = \mathbb{E}[-\log p(\boldsymbol{x})]$$

- If \boldsymbol{x} is discrete, $h(\boldsymbol{x})$ = expected number of bits to optimally encode the values \boldsymbol{x} takes = **coding rate**.
- If \boldsymbol{x} has low-dimensional support in \mathbb{R}^D , $h(\boldsymbol{x}) = -\infty$.

Comparing Encoding Schemes for a Sample Set

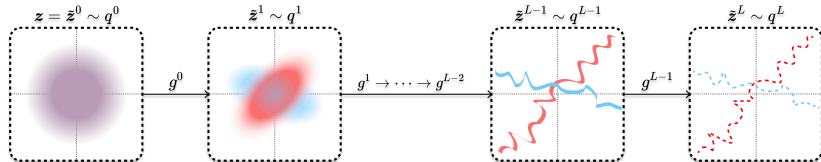
Given samples $\{\mathbf{x}_i\}_{i=1}^N \subseteq \mathbb{R}^D$ drawn from an unknown distribution p , to encode them, we can use codebooks for:

- The empirical distribution;
- An isotropic Gaussian distribution $p^n = \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ where σ^2 is large enough to cover all samples;
- A Gaussian distribution $p^e = \mathcal{N}(\mathbf{0}, (1/N) \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^\top)$;
- ...;
- Or a distribution $q \approx p$.

$$h(p^n) > h(p^e) > \cdots > h(q)$$

Two Possibilities for Learning a General Distribution

- Among a large family of possible coding schemes for the data, continually search for **better schemes with lower coding rate/entropy**.
- Starting from a high-entropy template (e.g., Gaussian), gradually **transform it towards the data distribution** by removing entropy.



Compression is the mechanism of learning.

How to Incrementally Remove Entropy via Denoising

General methodology:

- First, gradually *increase* the entropy of data by adding noise.
- Then, learn an *approximate inverse* to this mapping.

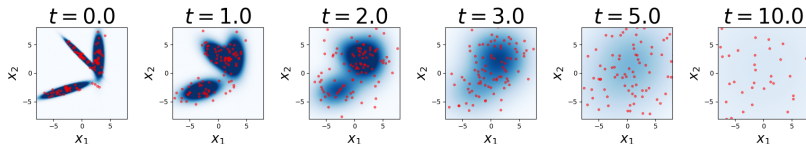
Leverage knowledge of the (*low-dimensional*) *structure of the data distribution* to learn a good inverse!

Diffusion Increases Entropy

Simplest recipe to add entropy: Add isotropic Gaussian noise.

$$\mathbf{x}_t := \mathbf{x} + t\mathbf{g}, \quad t \in [0, T]$$

with $\mathbf{g} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ independent of \mathbf{x} .



Theorem (de Bruijn, Informal): Under certain natural technical conditions for \mathbf{x} ,

$$\frac{d}{dt} h(\mathbf{x}_t) \geq 0, \quad \forall t \in (0, T)$$

Denoising

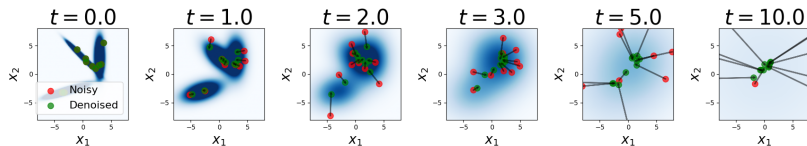
Simplest recipe to *remove* entropy: **denoise!**

For any $t \geq 0$, the *conditional expectation* $\bar{x}^*(t, \cdot) := \mathbb{E}[x \mid x_t = \cdot]$ is an *optimal denoiser*:

$$\bar{x}^*(t, \cdot) \in \arg \min_{\bar{x}(t, \cdot)} \mathbb{E}_{x, x_t} \|\mathbf{x} - \bar{x}(t, x_t)\|_2^2$$

Tweedie's Formula ([Miy+61; Rob64]): For $t \geq 0$,

$$\mathbb{E}[x \mid x_t = \xi] = \xi + \underbrace{t^2 \nabla \log p_t(\xi)}_{\text{score function at time } t}$$



Gaussian Mixture Denoising

Important, topical class of examples for this tutorial.

$$\mathbf{x} \sim \sum_{k=1}^K \pi_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$\implies \bar{\mathbf{x}}^*(t, \mathbf{x}_t) = \sum_{k=1}^K \frac{\pi_k \phi(\mathbf{x}_t; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k + t^2 \mathbf{I})}{\sum_{j=1}^K \pi_j \phi(\mathbf{x}_t; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j + t^2 \mathbf{I})} (\boldsymbol{\mu}_k + \boldsymbol{\Sigma}_k (\boldsymbol{\Sigma}_k + t^2 \mathbf{I})^{-1} (\mathbf{x}_t - \boldsymbol{\mu}_k))$$

$$\mathbf{x} \sim \frac{1}{K} \sum_{k=1}^K \mathcal{N}(\mathbf{0}, \mathbf{U}_k \mathbf{U}_k^\top)$$

$$\implies \bar{\mathbf{x}}^*(t, \mathbf{x}_t) = \frac{1}{1+t^2} \sum_{k=1}^K \frac{\exp(\frac{1}{2t^2(1+t^2)} \|\mathbf{U}_k^\top \mathbf{x}_t\|_2^2)}{\sum_{j=1}^K \exp(\frac{1}{2t^2(1+t^2)} \|\mathbf{U}_j^\top \mathbf{x}_t\|_2^2)} \mathbf{U}_k \mathbf{U}_k^\top \mathbf{x}_t,$$

$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{U} \mathbf{U}^\top) \implies \bar{\mathbf{x}}^*(t, \mathbf{x}_t) = \frac{1}{1+t^2} \mathbf{U} \mathbf{U}^\top \mathbf{x}_t \propto \text{PCA denoiser}$$

When Does Denoising Work?

Suppose $x \sim \mathcal{U}\{+1, -1\}$. Then

$$x_t \sim \frac{1}{2} \mathcal{N}(-1, t^2) + \frac{1}{2} \mathcal{N}(+1, t^2).$$

Can compute score and use Tweedie's formula to get:

$$\bar{x}^*(t, x_t) = \frac{\phi(x_t; +1, t^2) - \phi(x_t; -1, t^2)}{\phi(x_t; +1, t^2) + \phi(x_t; -1, t^2)} = \tanh\left(\frac{x_t}{t^2}\right).$$

- For $t \gg 0$, $\bar{x}^* \approx 0$ for many inputs, off-support.
- For $t \approx 0$, $\bar{x}^* \approx \{+1, -1\}$ for many inputs.

Denoising requires **several** steps!

Natural Iterative Denoising Algorithm

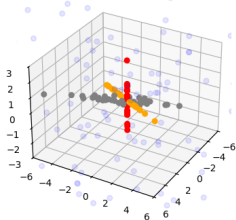
Denoise a **little bit of noise** at a time.

- Break $[0, T]$ up into $L + 1$ timesteps $0 = t_0 < \dots < t_L = T$, each interval of width $\delta := T/L$ (hence $t_\ell = (\ell/L)T$).
- Initialize $\hat{\mathbf{x}}_T \sim \mathbf{x}_T$.
- For ℓ running backwards from $\ell = L$ to $\ell = 1$, compute:

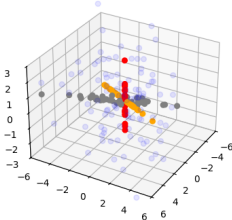
$$\begin{aligned}\hat{\mathbf{x}}_{t_{\ell-1}} &:= \mathbb{E}[\mathbf{x}_{t_{\ell-1}} \mid \mathbf{x}_{t_\ell} = \hat{\mathbf{x}}_{t_\ell}] \\ &= \mathbb{E}[\mathbf{x} + t_{\ell-1}\mathbf{g} \mid \mathbf{x}_{t_\ell} = \hat{\mathbf{x}}_{t_\ell}] = \mathbb{E}\left[\mathbf{x} + t_{\ell-1} \cdot \frac{\mathbf{x}_t - \mathbf{x}}{t_\ell} \mid \mathbf{x}_{t_\ell} = \hat{\mathbf{x}}_{t_\ell}\right] \\ &= \frac{t_{\ell-1}}{t_\ell} \hat{\mathbf{x}}_t + \left(1 - \frac{t_{\ell-1}}{t_\ell}\right) \mathbb{E}[\mathbf{x} \mid \mathbf{x}_t = \hat{\mathbf{x}}_t] \\ &= \left(1 - \frac{1}{\ell}\right) \hat{\mathbf{x}}_t + \frac{1}{\ell} \mathbb{E}[\mathbf{x} \mid \mathbf{x}_t = \hat{\mathbf{x}}_t].\end{aligned}$$

Simple Denoising Algorithm

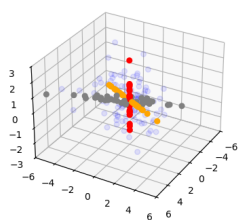
$\ell = L = 500 \mid t_\ell = 5.00$



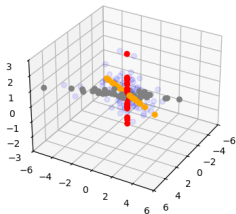
$\ell = 200 \mid t_\ell = 2.00$



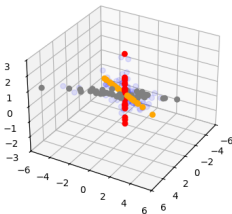
$\ell = 100 \mid t_\ell = 1.00$



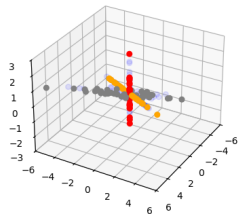
$\ell = 50 \mid t_\ell = 0.50$



$\ell = 25 \mid t_\ell = 0.25$



$\ell = 0 \mid t_\ell = 0.00$

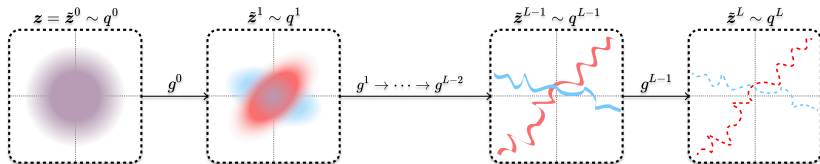


Denoising Reduces Entropy

Simplest recipe to *remove* entropy: **denoise!**

Theorem (Informal): Under certain natural technical conditions for x , each denoising step reduces entropy, i.e.,

$$h(\mathbb{E}[\mathbf{x}_s \mid \mathbf{x}_t]) < h(\mathbf{x}_t)$$



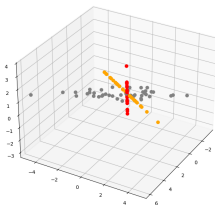
Iterative Denoising for Sampling

Our idealized algorithm needs a few changes to be practical.

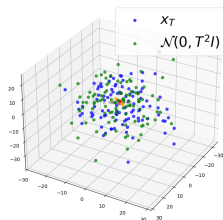
- **Problem:** Cannot initialize at x_T .
- **Fix:** Initialize at $\mathcal{N}(\mathbf{0}, T^2 \mathbf{I})$ for large T .

$$\begin{aligned}\frac{x_T}{T} &= \frac{x + Tg}{T} = \frac{x}{T} + g \approx g \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \implies x_T &\approx Tg \sim \mathcal{N}(\mathbf{0}, T^2 \mathbf{I}).\end{aligned}$$

Original Data



Noise Approximations ($T = 10.0$)



Changes to Denoising Algorithm

- Different discretizations, e.g., $t_\ell = C_1(e^{C_2 t} - 1)$;
- Different dynamics model, e.g., $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \mathbf{g}$;
 - Tweedie's formula becomes

$$\mathbb{E}[\mathbf{x} \mid \mathbf{x}_t] = \frac{1}{\alpha_t} (\mathbf{x}_t + \sigma_t^2 \nabla \log p_t(\mathbf{x}_t))$$

- The denoising iteration becomes

$$\hat{\mathbf{x}}_{t_{\ell-1}} = \frac{\sigma_{t_{\ell-1}}}{\sigma_{t_\ell}} \hat{\mathbf{x}}_{t_\ell} + \left(\alpha_{t_{\ell-1}} - \frac{\sigma_{t_{\ell-1}}}{\sigma_{t_\ell}} \right) \bar{\mathbf{x}}(t, \mathbf{x}_t)$$

- Combine training pipeline to train all denoisers for different t 's at once (say, using a deep network $\bar{\mathbf{x}}_\theta$):

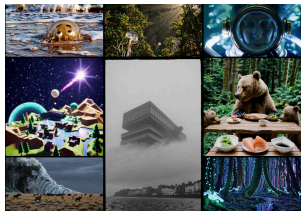
$$\min_{\theta} \mathbb{E}_t w(t) \mathbb{E}_{\mathbf{x}, \mathbf{x}_t} \|\mathbf{x} - \bar{\mathbf{x}}_\theta(t, \mathbf{x}_t)\|_2^2$$

- (Optional): Rescale the denoising target, e.g., “noise prediction” or “velocity prediction”.

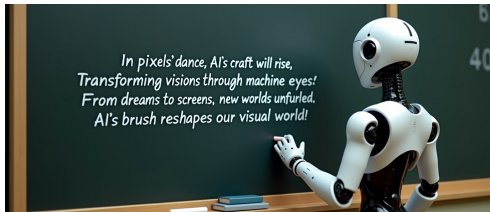
Implications for Diffusion Models



[HJA20]



Sora



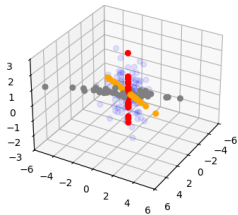
Flux

Connections to Practical Diffusion Models

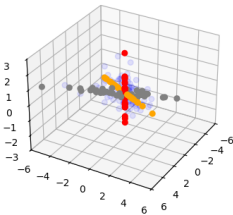
- Our simple iterative denoising algorithm is actually a famous diffusion model algorithm called Denoising Diffusion Implicit Models (DDIM) [SME20; De +25].
- Started with $t_\ell = (\ell/L)T$: *uniform discretization*.
- Started with $\alpha_t = 1, \sigma_t = t$: *Variance Exploding (VE) process*.
- $\alpha_t = \sqrt{1 - t^2}, \sigma_t = t$: *Variance Preserving (VP) process*.
- $\alpha_t = 1 - t, \sigma_t = t$: *Flow Matching (FM) process*.

Variance Preserving Process

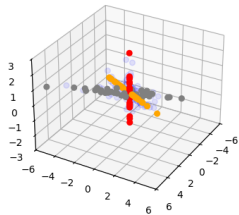
$\ell = L = 500 \mid t_\ell = 1.00$



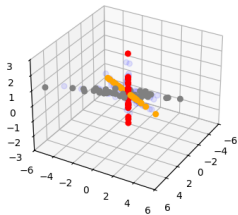
$\ell = 200 \mid t_\ell = 0.40$



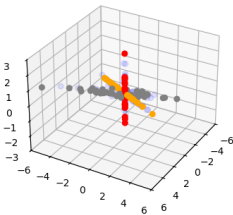
$\ell = 100 \mid t_\ell = 0.20$



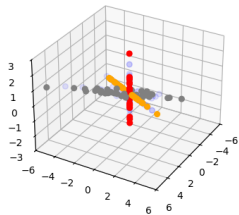
$\ell = 50 \mid t_\ell = 0.10$



$\ell = 25 \mid t_\ell = 0.05$



$\ell = 0 \mid t_\ell = 0.00$



Is It Theoretically Sound?

Define *total variation (TV) distance*:

$$\text{TV}(\mathbf{x}, \mathbf{y}) := \max_{A \subseteq \mathbb{R}^D} |\mathbb{P}[\mathbf{x} \in A] - \mathbb{P}[\mathbf{y} \in A]|.$$

Theorem ([LY24] Theorem 1, Simplified.) Suppose $\hat{\mathbf{x}}$ is sampled according to DDIM w/ VP process, denoiser $\bar{\mathbf{x}}_\theta$, and exponential discretization. Then

$$\text{TV}(\mathbf{x}, \hat{\mathbf{x}}) = \tilde{\mathcal{O}} \left(\underbrace{\frac{D}{L}}_{\text{discretization}} + \underbrace{\left(\frac{1}{L} \sum_{\ell=1}^L \frac{\alpha_{t_\ell}}{\sigma_{t_\ell}^2} \mathbb{E}_{\mathbf{x}, \mathbf{x}_{t_\ell}} \|\bar{\mathbf{x}}_\theta(t_\ell, \mathbf{x}_{t_\ell}) - \bar{\mathbf{x}}^*(t_\ell, \mathbf{x}_{t_\ell})\|_2^2 \right)}_{\text{avg. excess error of denoiser}} \right)^{1/2}$$

Goes to 0 as $L \rightarrow \infty$ and $\bar{\mathbf{x}}_\theta$ becomes perfect!

Better Guarantees With Low-Dimensional Structure

Theorem ([LY24] Theorem 2, Simplified.) Suppose \hat{x} is sampled according to DDIM w/ a **slightly modified** VP-like process, denoiser \bar{x}_θ , and exponential discretization. Then

$$\text{TV}(\mathbf{x}, \hat{\mathbf{x}}) = \tilde{O}\left(\underbrace{\frac{d}{L}}_{\text{discretization}} + \underbrace{\left(\frac{1}{L} \sum_{\ell=1}^L \frac{\alpha_{t_\ell}}{\sigma_{t_\ell}^2} \mathbb{E}_{\mathbf{x}, \mathbf{x}_{t_\ell}} \|\bar{\mathbf{x}}_\theta(t_\ell, \mathbf{x}_{t_\ell}) - \bar{\mathbf{x}}^*(t_\ell, \mathbf{x}_{t_\ell})\|_2^2\right)}_{\text{avg. excess error of denoiser}}\right)^{1/2}$$

where d is an appropriately-defined measure of the *intrinsic dimension* of the support of \mathbf{x} .

Note: Using real VP still incurs some (mild) ambient dimension-dependence even with low-dimensional structure.

Denoising too Far Leads to Memorization

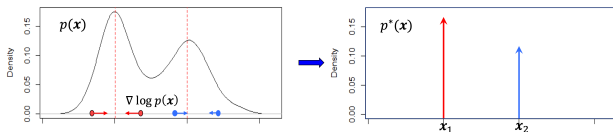
The *global minimizer* of the *finite sample training loss*

$$\min_{\bar{x}} \frac{1}{N} \sum_{i=1}^N \mathbb{E}_t w(t) \mathbb{E}_{\mathbf{x}_t^i} \|\mathbf{x}^i - \bar{x}(t, \mathbf{x}_t^i)\|_2^2$$

is a *denoiser* whose samples are always “*memorized*”:

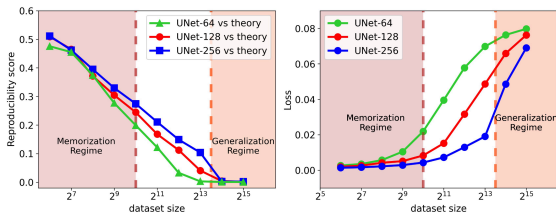
$$\bar{x}^{\text{mem}}(t, \mathbf{x}_t) = \sum_{i=1}^N \frac{e^{-\|\mathbf{x}_t - \alpha_t \mathbf{x}^i\|_2^2 / (2\sigma_t^2)}}{\sum_{j=1}^N e^{-\|\mathbf{x}_t - \alpha_t \mathbf{x}^j\|_2^2 / (2\sigma_t^2)}} \mathbf{x}^i$$

Sampling process reduces the entropy *too much*.



Why Don't Trained Models Memorize?

Some do, if they can approximate the memorizing denoiser:



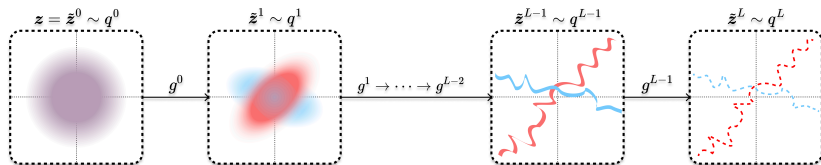
[Zha+24]

Competing theoretical explanations: *implicit bias*, *loss landscape*.

Overall: Want models and architectures that can approximate the true denoiser but are not powerful enough to approximate the memorizing one! *Scale is not all there is...?*

Summary

- *Compression* as a foundational criterion for learning.
- Denoising and diffusion as a mechanism for compression.
- Diffusion models, and (near) state-of-the-art samplers.



References I

- [De +25] Valentin De Bortoli, Alexandre Galashov, J Swaroop Guntupalli, et al. “Distributional diffusion models with scoring rules”. In: *arXiv preprint arXiv:2502.02483* (2025).
- [HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.
- [LY24] Gen Li and Yuling Yan. “O (d/T) convergence theory for diffusion probabilistic models under minimal assumptions”. In: *arXiv preprint arXiv:2409.18959* (2024).

References II

- [Miy+61] Koichi Miyasawa et al. “An empirical Bayes estimator of the mean of a normal population”. In: *Bull. Inst. Internat. Statist* 38.181-188 (1961), pp. 1–2.
- [Rob64] Herbert Robbins. “The empirical Bayes approach to statistical decision problems”. In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 1–20.
- [SME20] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising diffusion implicit models”. In: *arXiv preprint arXiv:2010.02502* (2020).
- [Wan+24] Peng Wang, Huijie Zhang, Zekai Zhang, et al. “Diffusion models learn low-dimensional distributions via subspace clustering”. In: *arXiv preprint arXiv:2409.02426* (2024).

References III

- [Zha+24] Huijie Zhang, Jinfan Zhou, Yifu Lu, et al. “The emergence of reproducibility and consistency in diffusion models”. In: International Conference on Machine Learning. 2024.