Probabilistic Deep Learning for Solving Inverse Problems

Uncertainty Quantification Summer School

Assad A Oberai

USC

August 4th 2023

Includes work with

- Deep Ray
- Javier Murgoitio Esandi
- Harisankar Ramaswamy
- Bryan Shaddy
- Agnimitra Dasgupta

Outline

Table of Contents

Generative Problem

Given $\boldsymbol{x}^{(i)} \sim P_{\text{data}}(\boldsymbol{x})$ generate more samples.

Utility:

- Data augmentation.
- New designs.
- Reduced representation.

Conditional Generative Problem

Let $\boldsymbol{X}$ denote a random vector to be inferred, and $\boldsymbol{Y}$ denote the measured random vector. Given

1. The measurement $\boldsymbol{Y} = \hat{\boldsymbol{y}}$.
2. Some prior knowledge of $\boldsymbol{X}$ encoded in $P_{\text{data}}(\boldsymbol{x})$.
3. The forward map $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\eta})$.

generate samples from $P_{\text{data}}(\boldsymbol{x}|\hat{\boldsymbol{y}})$.

Utility:

- Inverse problems with a known forward map.
- Data assimilation.

Can be reformulated .....

Conditional Generative Problem

Steps

1. Sample $\boldsymbol{x}^{(i)} \sim P_{\text{data}}(\boldsymbol{x})$.
2. Compute $\boldsymbol{y}^{(i)} = \boldsymbol{f}(\boldsymbol{x}^{(i)}, \eta)$.
3. Collect $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) \sim P_{\text{data}}(\boldsymbol{x}, \boldsymbol{y})$.

Reduced problem: given

- The measurement $\boldsymbol{Y} = \hat{\boldsymbol{y}}$.
- $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) \sim P_{\text{data}}(\boldsymbol{x}, \boldsymbol{y})$

generate samples from $P_{\text{data}}(\boldsymbol{x}|\hat{\boldsymbol{y}})$.

Outline

- Wasserstein Generative Adversarial Networks (WGAN).
- Conditional Wasserstein Generative Adversarial Networks (CWGAN).
- Generative Diffusion Networks.
- Conditional Generative Diffusion Networks.

Table of Contents

Architecture

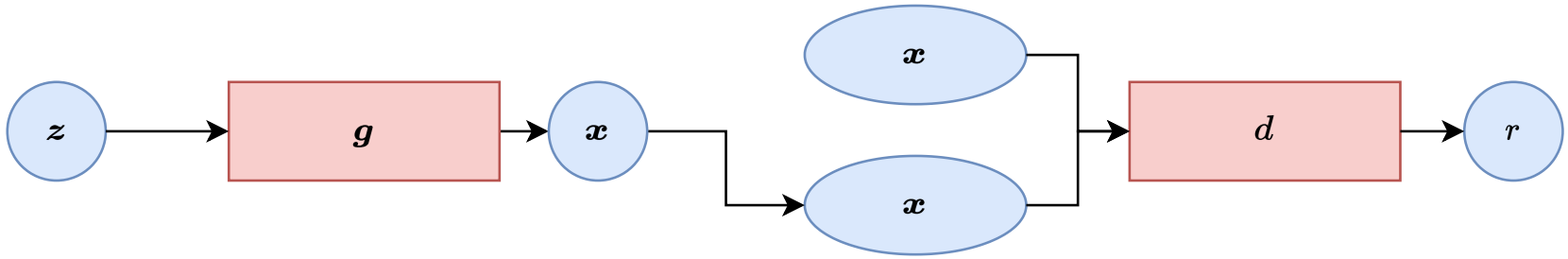


Figure: Schematic of a WGAN.

Key references:

- Villani, Cédric. Optimal transport: old and new. Vol. 338. Berlin: springer, 2009.
- Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." International conference on machine learning. PMLR, 2017.
- Gulrajani, Ishaan, et al. "Improved training of wasserstein gans." Advances in neural information processing systems 30 (2017).

WGAN: Generator and Critic

Properties of the generator:

- $\boldsymbol{g} : \mathbb{R}^{N_z} \times \mathbb{R}^{N_\theta} \to \mathbb{R}^{N_x}$
- $\boldsymbol{z} \sim P_Z$ and $N_z \ll N_x$
- $\boldsymbol{x} \sim P_g \Rightarrow \boldsymbol{x} = \boldsymbol{g}(\boldsymbol{z}; \boldsymbol{\theta}), \boldsymbol{z} \sim P_Z$.

Properties of the critic:

- $d : \mathbb{R}^{N_x} \times \mathbb{R}^{N_\phi} \to \mathbb{R}, \quad d : \boldsymbol{x} \mapsto r$
- $r = d(\boldsymbol{x}; \boldsymbol{\phi})$.

"Loss" function:

$$L(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{P_{\text{data}}} \left[d(\boldsymbol{x}; \boldsymbol{\phi}) \right] - \mathbb{E}_{P_Z} \left[d(\boldsymbol{g}(\boldsymbol{z}; \boldsymbol{\theta}) \boldsymbol{\phi}) \right].$$

Optimality

Optimal values (min-max problem):

$$(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \ \underset{\boldsymbol{\phi}, d \in \mathrm{Lip}}{\operatorname{argmax}} \ L(\boldsymbol{\theta}, \boldsymbol{\phi}).$$

With infinite data, infinite critic capacity achieving the maximum implies,

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \ W_1(P_{\mathrm{data}}, P_g).$$

If this minima is achieved $W_1(P_{\mathrm{data}}, P_{g^*}) = 0$. This implies weak convergence of the measures. Which in turn means,

$$\mathbb{E}_{P_{\mathrm{data}}}\left[l(\boldsymbol{x})\right] = \mathbb{E}_{P_Z}\left[l(\boldsymbol{g}(\boldsymbol{z}; \boldsymbol{\theta}^*))\right], \forall l \in C_b(\Omega_X).$$

Implementation of WGAN

- In practice, the "loss" term is approximated as:

$$
\begin{aligned}
L(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \frac{1}{N}\left(\sum_{i=1}^{N} d(\boldsymbol{x}^{(i)}; \boldsymbol{\phi}) - \sum_{i=1}^{N} d(\boldsymbol{g}(\boldsymbol{z}^{(i)}; \boldsymbol{\theta}); \boldsymbol{\phi})\right) + \lambda P(\boldsymbol{\phi}). \\
P(\boldsymbol{\phi}) &= \frac{1}{N}\sum_{i=1}^{N}(\|\nabla d(\boldsymbol{h}^{(i)}; \boldsymbol{\phi})\|_2 - 1)^2 \\
\boldsymbol{h}^{(i)} &= \delta^{(i)}\boldsymbol{x}^{(i)} + (1 - \delta^{(i)})\boldsymbol{g}(\boldsymbol{z}^{(i)}; \boldsymbol{\theta}), \quad \delta^{(i)} \sim U(0, 1).
\end{aligned}
$$

- The min-max problem is hard to solve. Use stochastic gradient descent-ascent. Five steps of ascent followed by one descent.
- Hard to track convergence. Look for stability of the "loss" function and track some statistic of the generated sample.

Implementation of WGAN

- If $\boldsymbol{x}$ is a vector, the generator and the critic are MLPs.
- If $\boldsymbol{x}$ is an image (discrete field), the generator is a U-Net with transpose convolutions. The critic in a CNN - like a classifier.

Table of Contents

Architecture

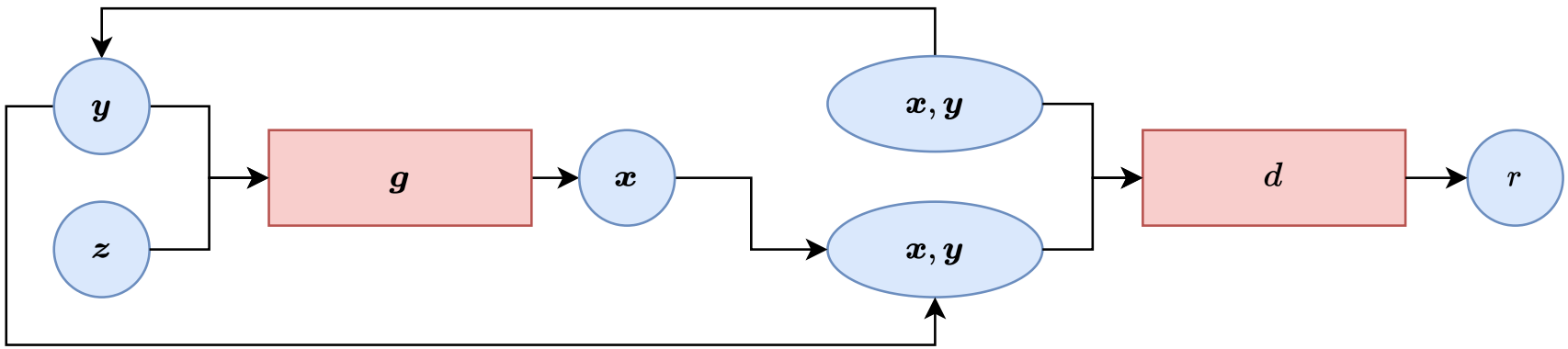


Figure: Schematic of a conditional WGAN.

Key references:

- Adler, Jonas, and Ozan Öktem. "Deep bayesian inversion." arXiv preprint arXiv:1811.05910 (2018).
- Ray, Deep, et al. "The efficacy and generalizability of conditional GANs for posterior inference in physics-based inverse problems." arXiv preprint arXiv:2202.07773 (2022).
- Ray, Deep, et al. "Solution of physics-based inverse problems using conditional generative adversarial networks with full gradient penalty." arXiv preprint arXiv:2306.04895 (2023).

CWGAN: Generator and Critic

Properties of the generator:

- $\boldsymbol{g} : \mathbb{R}^{N_z} \times \mathbb{R}^{N_y} \times \mathbb{R}^{N_\theta} \to \mathbb{R}^{N_x}$
- $\boldsymbol{z} \sim P_Z$ and $N_z \ll N_x$
- $\boldsymbol{x} \sim P_g(\boldsymbol{x}|\boldsymbol{y}) \Rightarrow \boldsymbol{x} = \boldsymbol{g}(\boldsymbol{z}, \boldsymbol{y}; \boldsymbol{\theta}), \boldsymbol{z} \sim P_Z$.

Properties of the critic:

- $d : \mathbb{R}^{N_x} \times \mathbb{R}^{N_y} \times \mathbb{R}^{N_\phi} \to \mathbb{R}$
- $r = d(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\phi})$.

"Loss" function:

$$L(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{P_{\text{data}}(\boldsymbol{x},\boldsymbol{y})} \left[d(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\phi}) \right] - \mathbb{E}_{\boldsymbol{z} \sim P_Z; \boldsymbol{y} \sim P_Y} \left[d(\boldsymbol{g}(\boldsymbol{z}, \boldsymbol{y}; \boldsymbol{\theta}); \boldsymbol{\phi}) \right].$$

Optimality

Optimal values (min-max problem):

$$(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \ \underset{\boldsymbol{\phi}, d \in \mathrm{Lip}}{\operatorname{argmax}} \ L(\boldsymbol{\theta}, \boldsymbol{\phi}).$$

With infinite data, infinite critic capacity achieving the maximum implies,

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \ W_1(P_{\mathrm{data}}, P_g P_Y).$$

If this minima is achieved $W_1(P_{\mathrm{data}}, P_{g^*} P_y) = 0$. This implies weak convergence of the measures. Which in turn means,

$$\mathbb{E}_{P_{\mathrm{data}}} \left[l(\boldsymbol{x}, \boldsymbol{y}) \right] = \mathbb{E}_{\boldsymbol{z} \sim P_Z, \boldsymbol{y} \sim P_Y} \left[l(\boldsymbol{g}(\boldsymbol{z}, \boldsymbol{y}; \boldsymbol{\theta}^*), \boldsymbol{y}) \right], \forall l \in C_b(\Omega_X \times \Omega_Y).$$

Let $\hat{\boldsymbol{y}}$ be given for which $P_Y(\hat{\boldsymbol{y}}) \neq 0$. Let the true and approximated conditional expectations be bounded. Then given $\epsilon > 0$ and $q \in C_b(\Omega_X)$, there exists a positive integer $\widetilde{N} := \widetilde{N}(\hat{\boldsymbol{y}}, q, \epsilon)$ such that

$$\left| \mathbb{E}_{P_{\text{data}}(\boldsymbol{x}|\hat{\boldsymbol{y}})} [q(\boldsymbol{x})] - \mathbb{E}_{P_{g^*}(\boldsymbol{x}|\hat{\boldsymbol{y}})} [q(\boldsymbol{x})] \right| < \epsilon \quad \forall\ N_\theta \geq \widetilde{N}$$

This implies

$$\mathbb{E}_{P_{\text{data}}(\boldsymbol{x}|\hat{\boldsymbol{y}})} [q(\boldsymbol{x})] \approx \mathbb{E}_{P_Z} [q(\boldsymbol{g}(\boldsymbol{z}, \hat{\boldsymbol{y}}; \boldsymbol{\theta}^*))] .$$

Alternate Sampling Strategy

Under the same conditions as the previous theorem, given $\epsilon > 0$ and $q \in C_b(\Omega_X)$, there exists a positive real number $\bar{\sigma} := \bar{\sigma}(\hat{\boldsymbol{y}}, q, \epsilon)$ and a positive integer with $\bar{N} := \bar{N}(\bar{\sigma})$ such that

$$\left| \mathbb{E}_{P_{\text{data}}(\boldsymbol{x}|\hat{\boldsymbol{y}})} [q(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{y} \sim P_{Y_\sigma}, \boldsymbol{x} \sim P_{g^*}(\boldsymbol{x}|\hat{\boldsymbol{y}})} [q(\boldsymbol{x})] \right| < \epsilon \quad \forall \; N_\theta \geq \bar{N}$$

Here $P_{Y_\sigma} = \mathcal{N}(\hat{\boldsymbol{y}}, \sigma \boldsymbol{I})$.

This implies

$$\mathbb{E}_{P_{\text{data}}(\boldsymbol{x}|\hat{\boldsymbol{y}})} [q(\boldsymbol{x})] \approx \mathbb{E}_{\boldsymbol{z} \sim P_Z, \boldsymbol{y} \sim P_{Y_\sigma}} [q(\boldsymbol{g}(\boldsymbol{z}, \boldsymbol{y}; \boldsymbol{\theta}^*))] .$$

Implementation of CWGAN

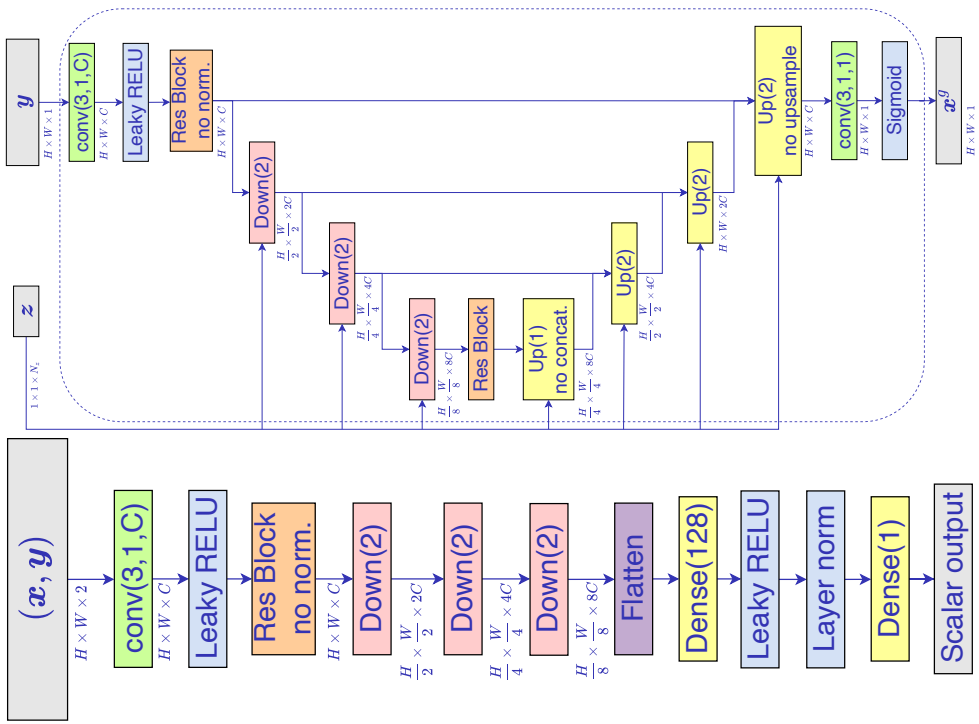
- In practice, the "loss" term is approximated as:

$$
\begin{aligned}
L(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \frac{1}{N}\left(\sum_{i=1}^{N} d(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\phi}) - \sum_{i=1}^{N} d(\boldsymbol{g}(\boldsymbol{z}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}; \boldsymbol{\phi})\right) \\
&\quad + \lambda P(\boldsymbol{\phi}). \\
P(\boldsymbol{\phi}) &= \frac{1}{N}\sum_{i=1}^{N} (\|\nabla d(\boldsymbol{h}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\phi})\|_2 - 1)^2 \\
\boldsymbol{h}^{(i)} &= \delta^{(i)} \boldsymbol{x}^{(i)} + (1 - \delta^{(i)}) \boldsymbol{g}(\boldsymbol{z}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}), \quad \delta^{(i)} \sim U(0, 1).
\end{aligned}
$$

- In the second term the gradient is wrt. both arguments.
- The min-max problem is hard to solve. Use stochastic gradient descent-ascent. Five steps of ascent followed by one descent.
- Hard to track convergence. Look for stability of the "loss" function and track statistics of generated samples. For a given $\boldsymbol{y}$ compute the generated $\boldsymbol{x}$'s and compare with the true $\boldsymbol{x}$.

Implementation of CWGAN

- If $\boldsymbol{X}$ and $\boldsymbol{Y}$ are vectors, the generator and the critic take the form of an MLP.
- If $\boldsymbol{X}$ and $\boldsymbol{Y}$ are images (discrete fields), the generator is a U-Net. Further $\boldsymbol{z}$ is used to perform conditional instance normalization. The critic is a CNN - like a classifier.



Example 1

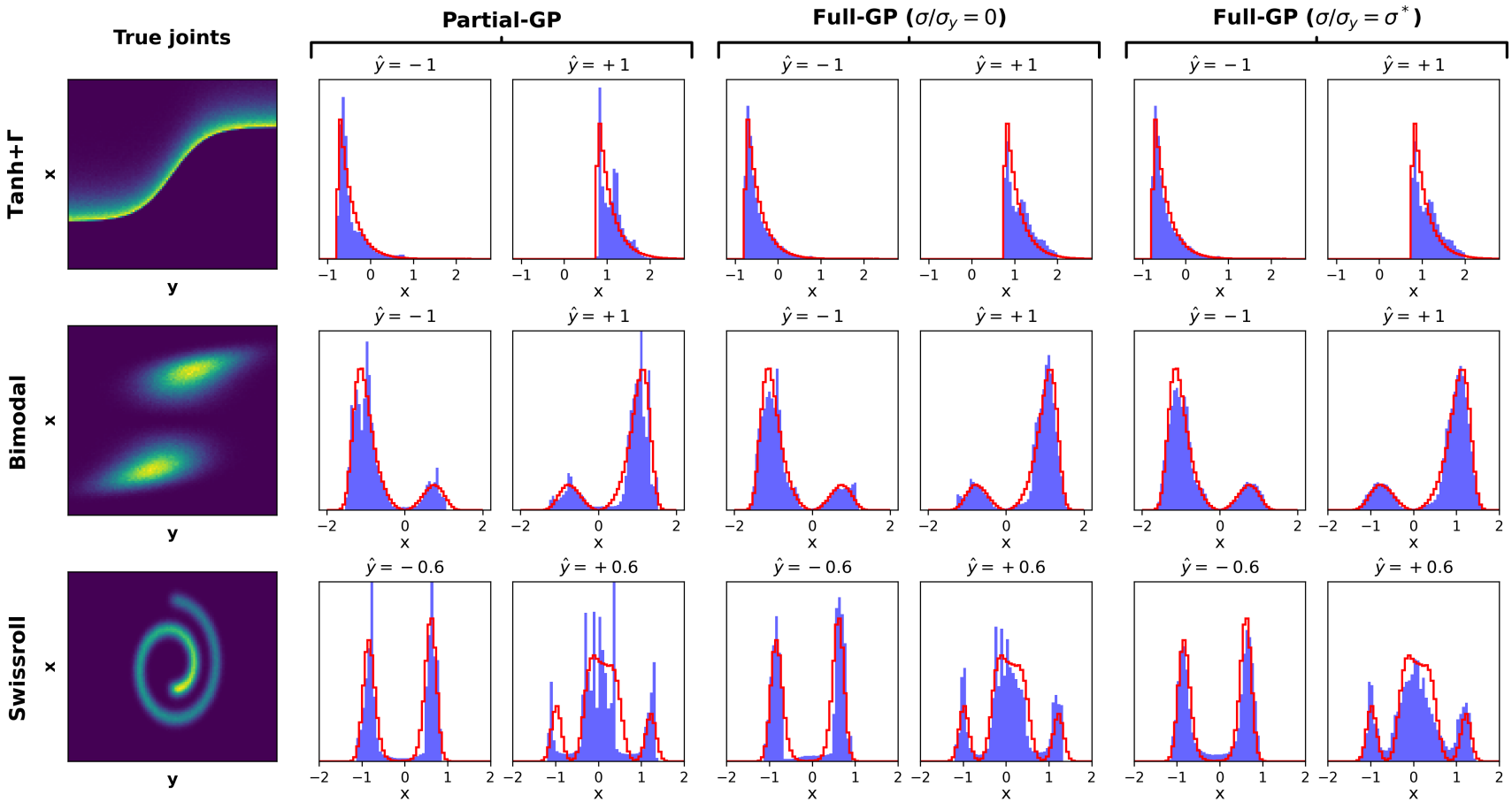
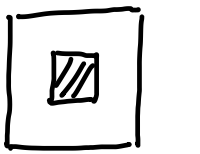


Figure: Histograms of 10^5 samples from $P_{g^*}(\boldsymbol{x}|\hat{\boldsymbol{y}})$. Red: outline the true conditional dist.

Example 2: Inverse heat equation



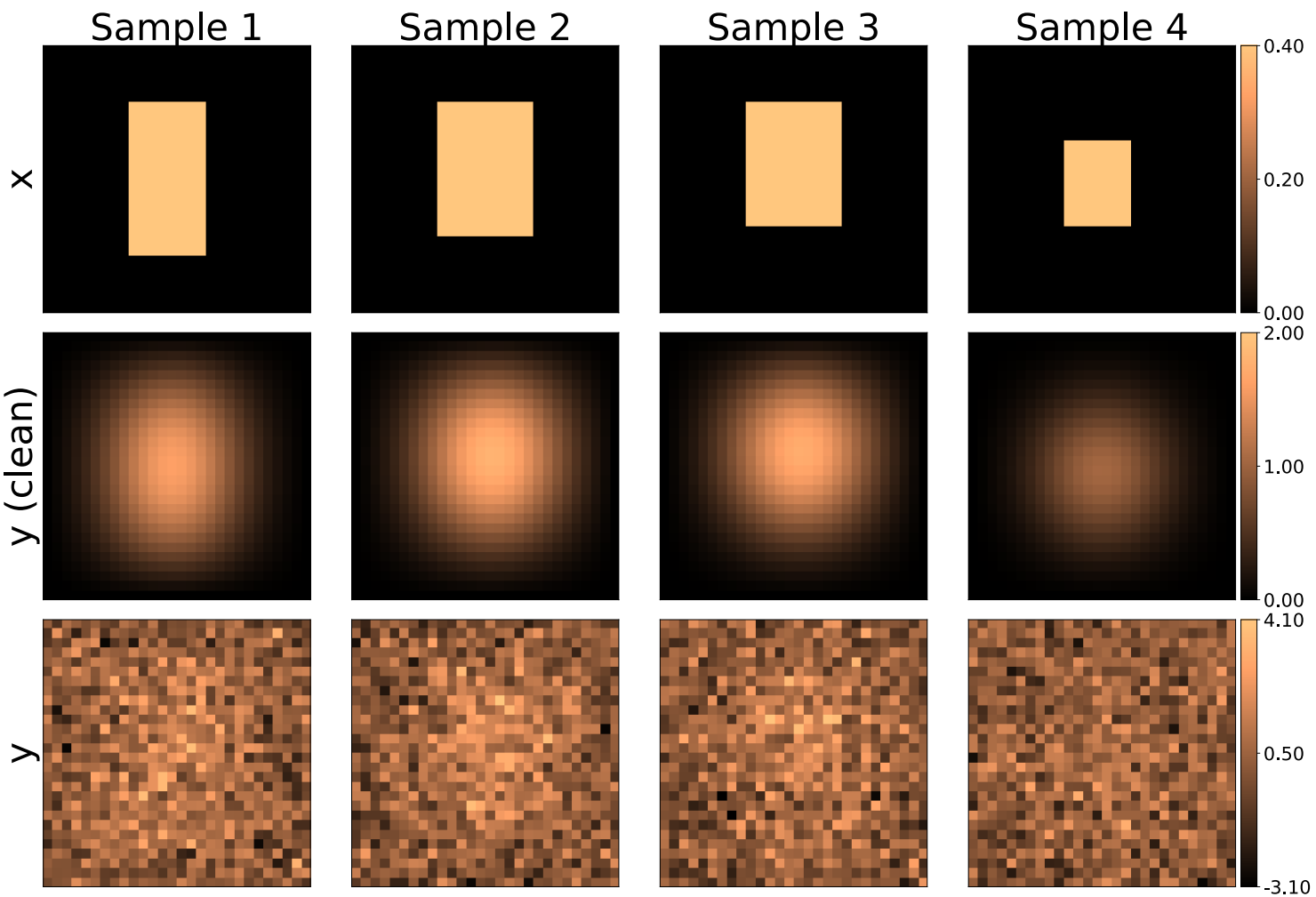


Figure: Samples from the rectangular prior dataset.

Example 2: Inverse heat equation

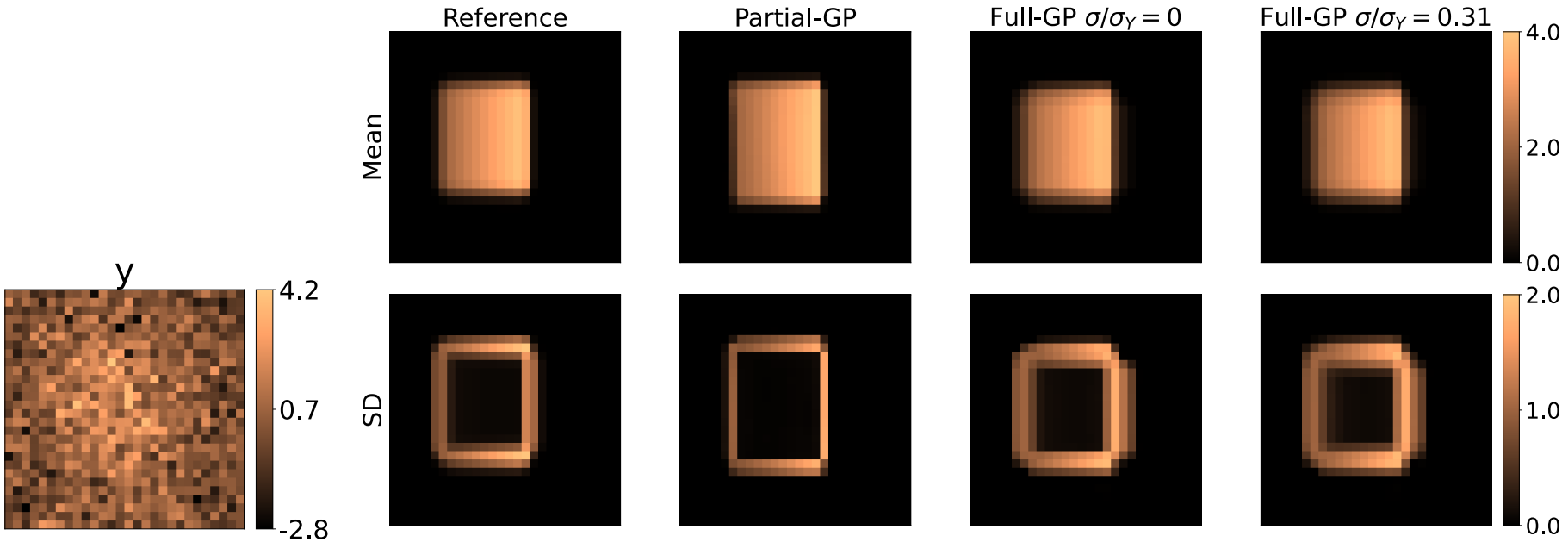


Figure: Measurement, Mean and SD from models and reference

Example 2: Inverse heat equation

Table: L^2 error in posterior statistics

Posterior Statistics	**Partial-GP**	**Full-GP**	
		$\sigma/\sigma_Y = 0$	$\sigma/\sigma_Y = 0.31$
Mean	0.441	0.420	0.406
SD	0.460	0.454	0.435

Example 3: Inverse Helmholtz equation

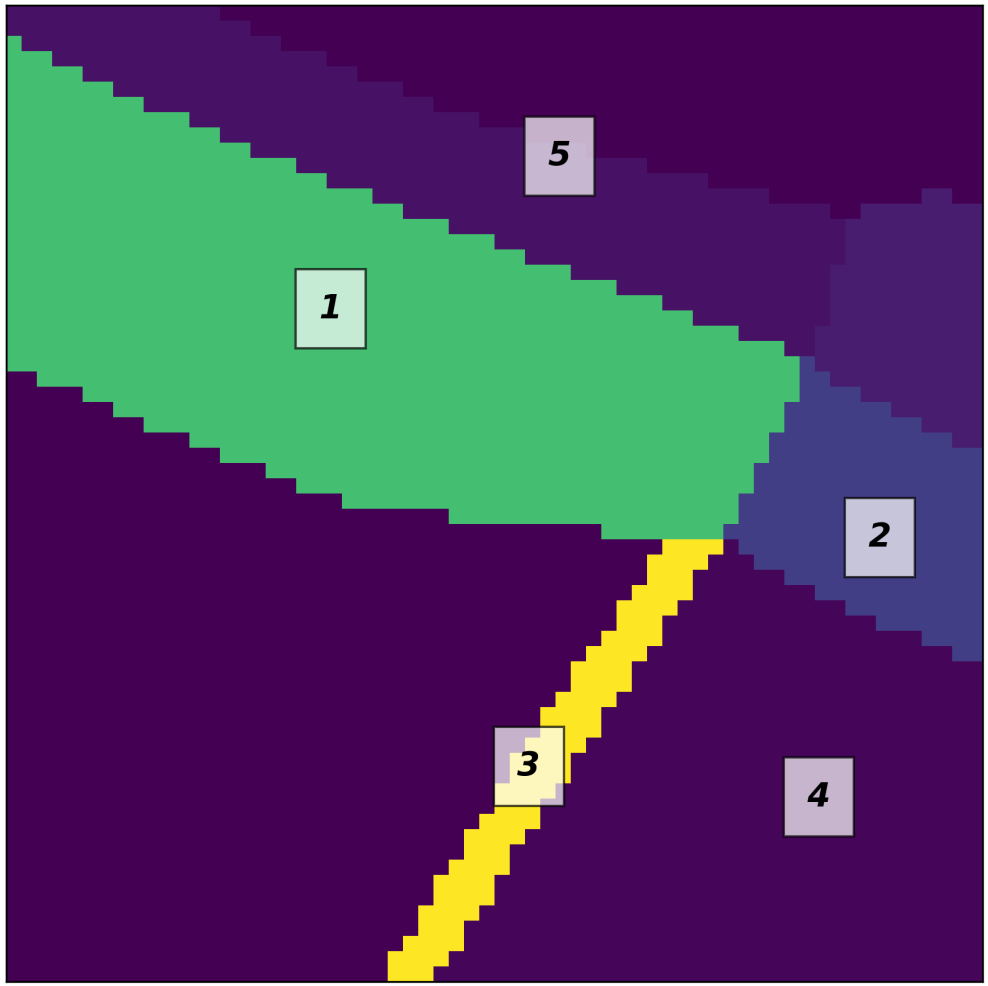


Figure: Geometry of the optical nerve head.

Example 3: Inverse Helmholtz equation

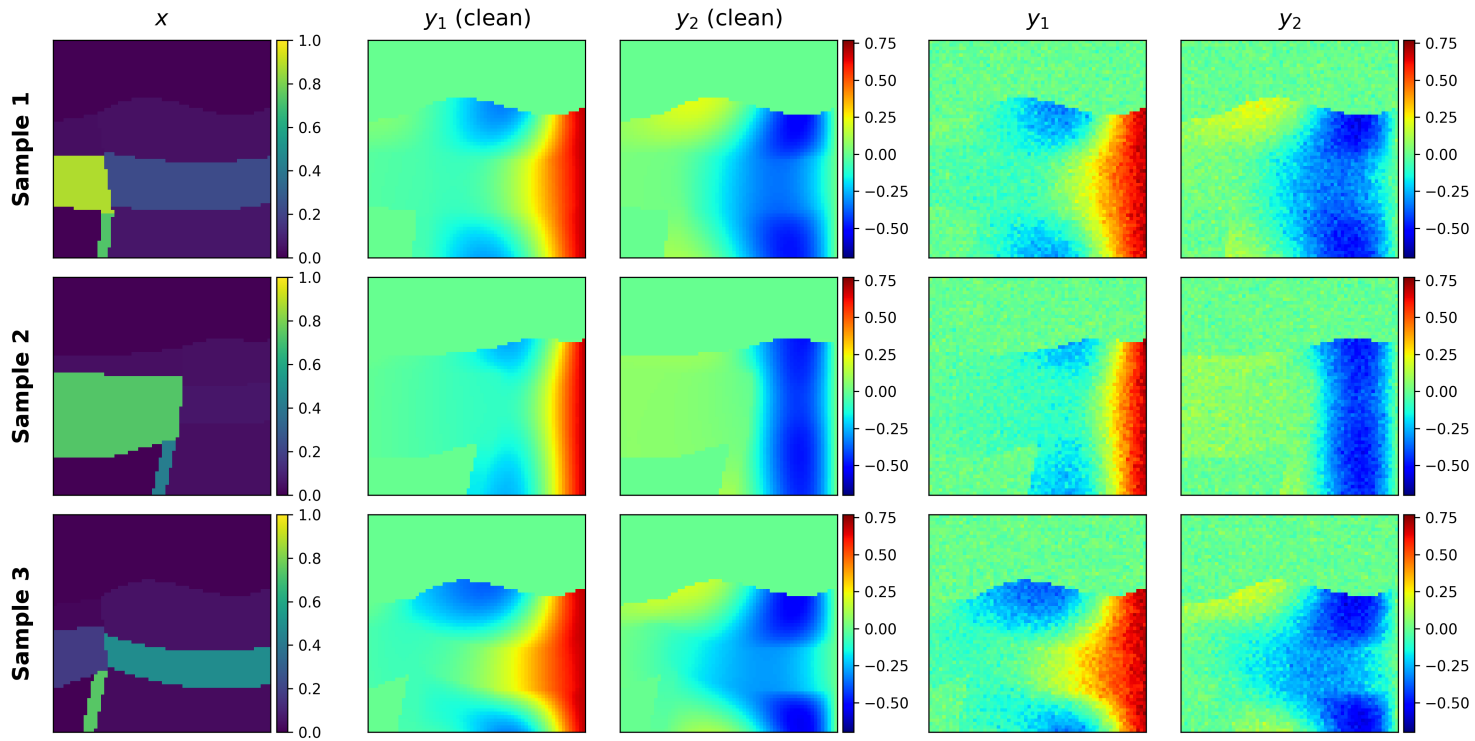


Figure: Training samples for the inverse Helmholtz problem.

Example 3: Inverse Helmholtz equation

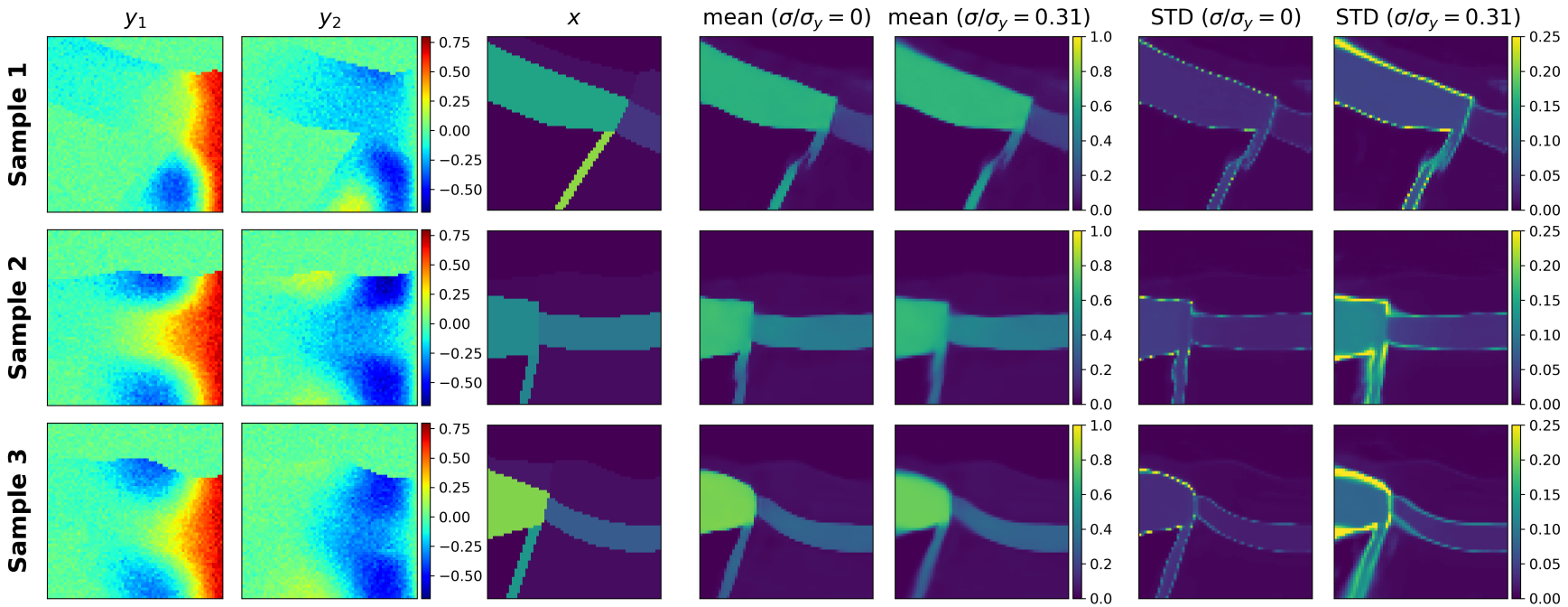


Figure: Point-wise mean and standard deviation.

Example 4: Inverse elasticity for tumor spheroids

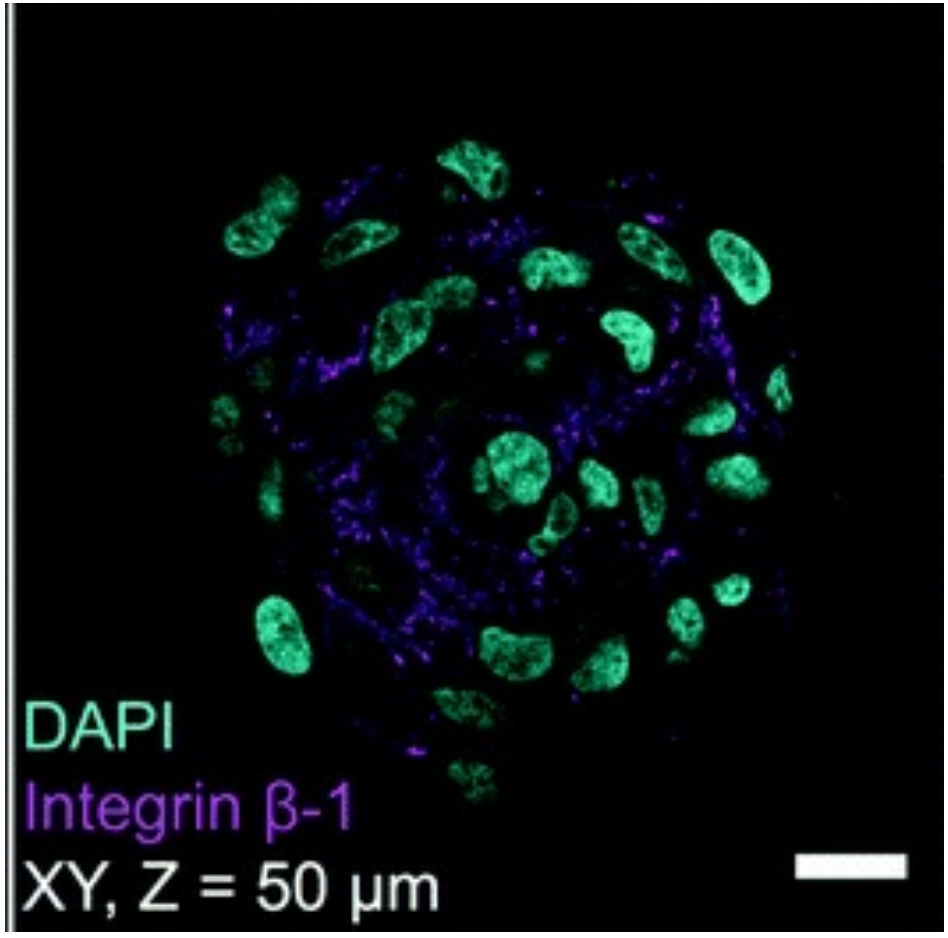


Figure: Confocal microscopy image of a tumor spheroid.

Example 4: Inverse elasticity for tumor spheroids (Kennedey & Foo)

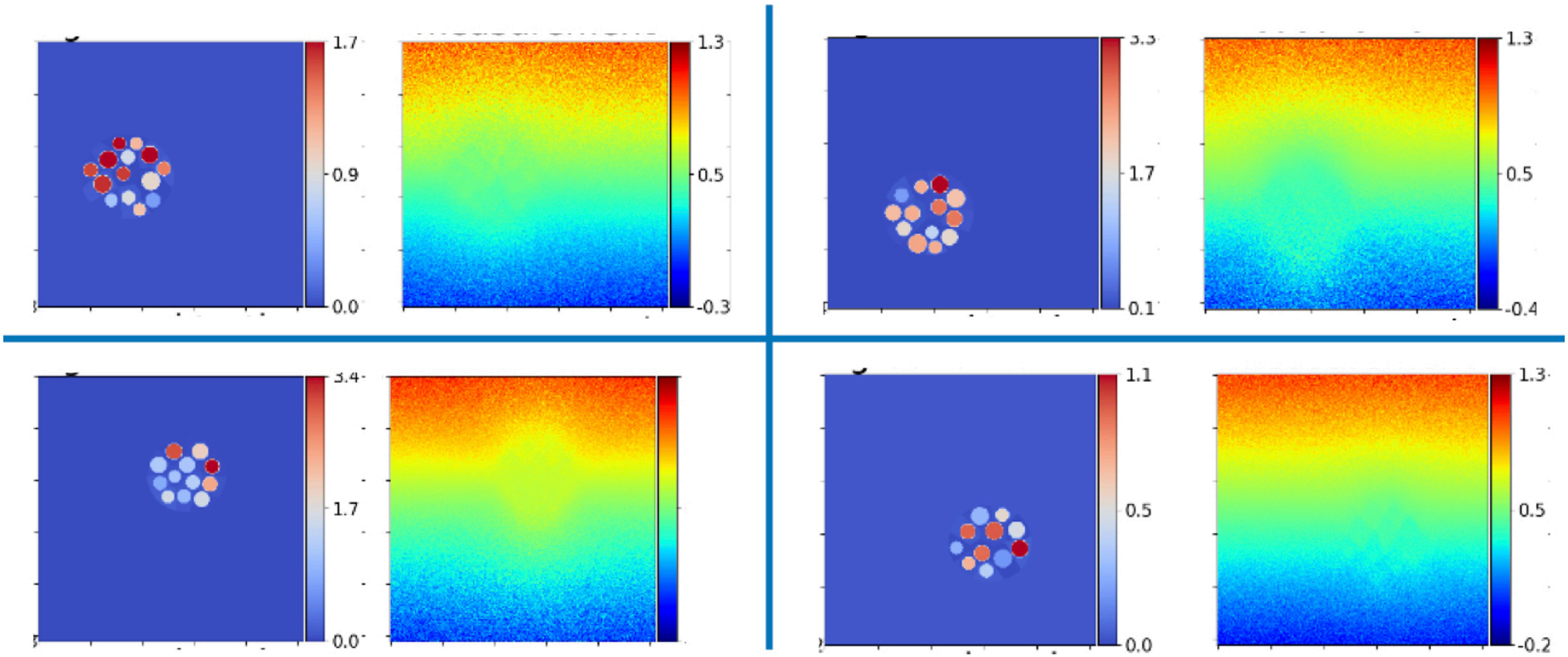


Figure: Training samples.

Example 4: Inverse elasticity for tumor spheroids

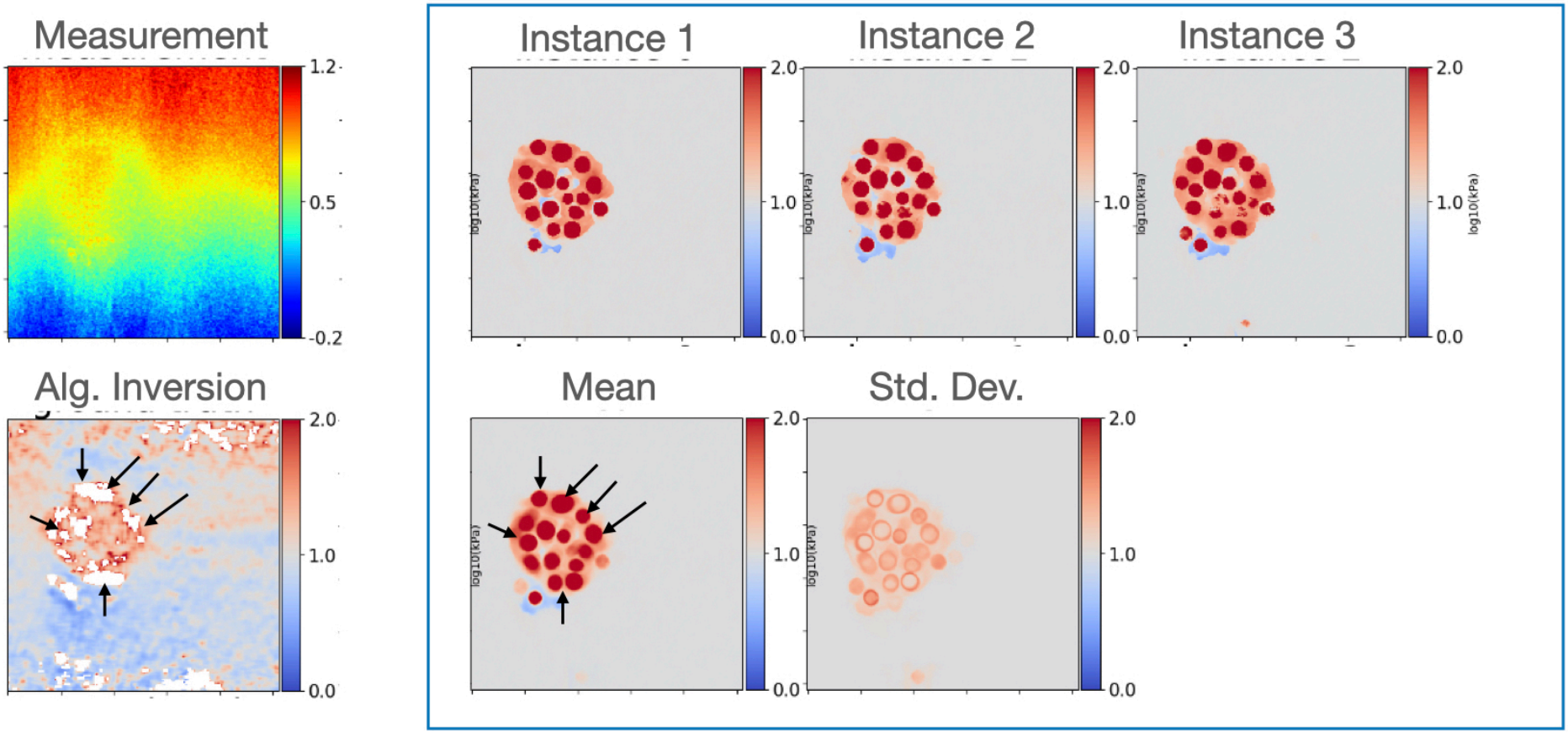
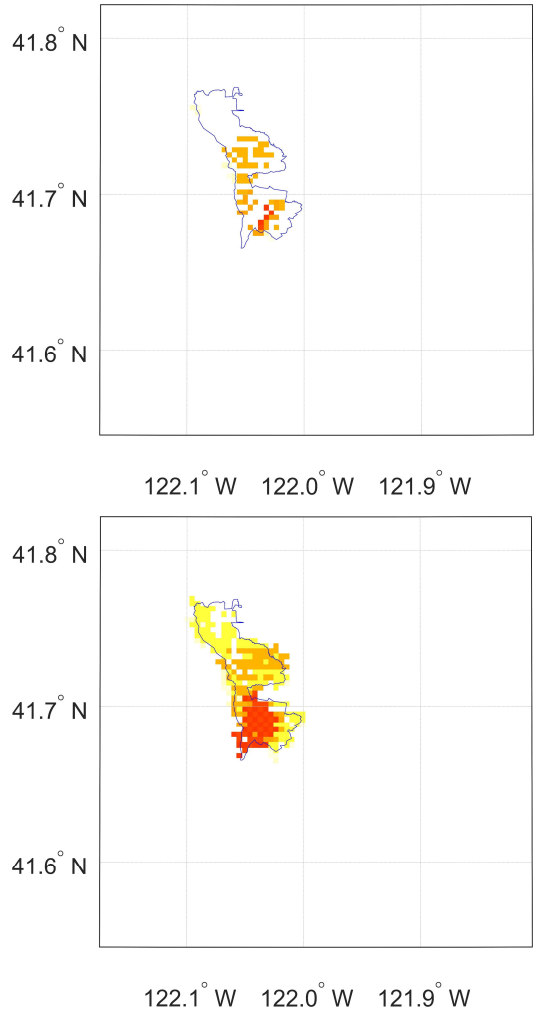


Figure: Generated samples, point-wise mean and standard deviation.

Example 5: Early Trajectory of a Wildfire (Shaddy, Mandel, Farguel..)



(b) Tennant

Figure: Satellite measurements of active fire regions.

Example 5: Early Trajectory of a Wildfire

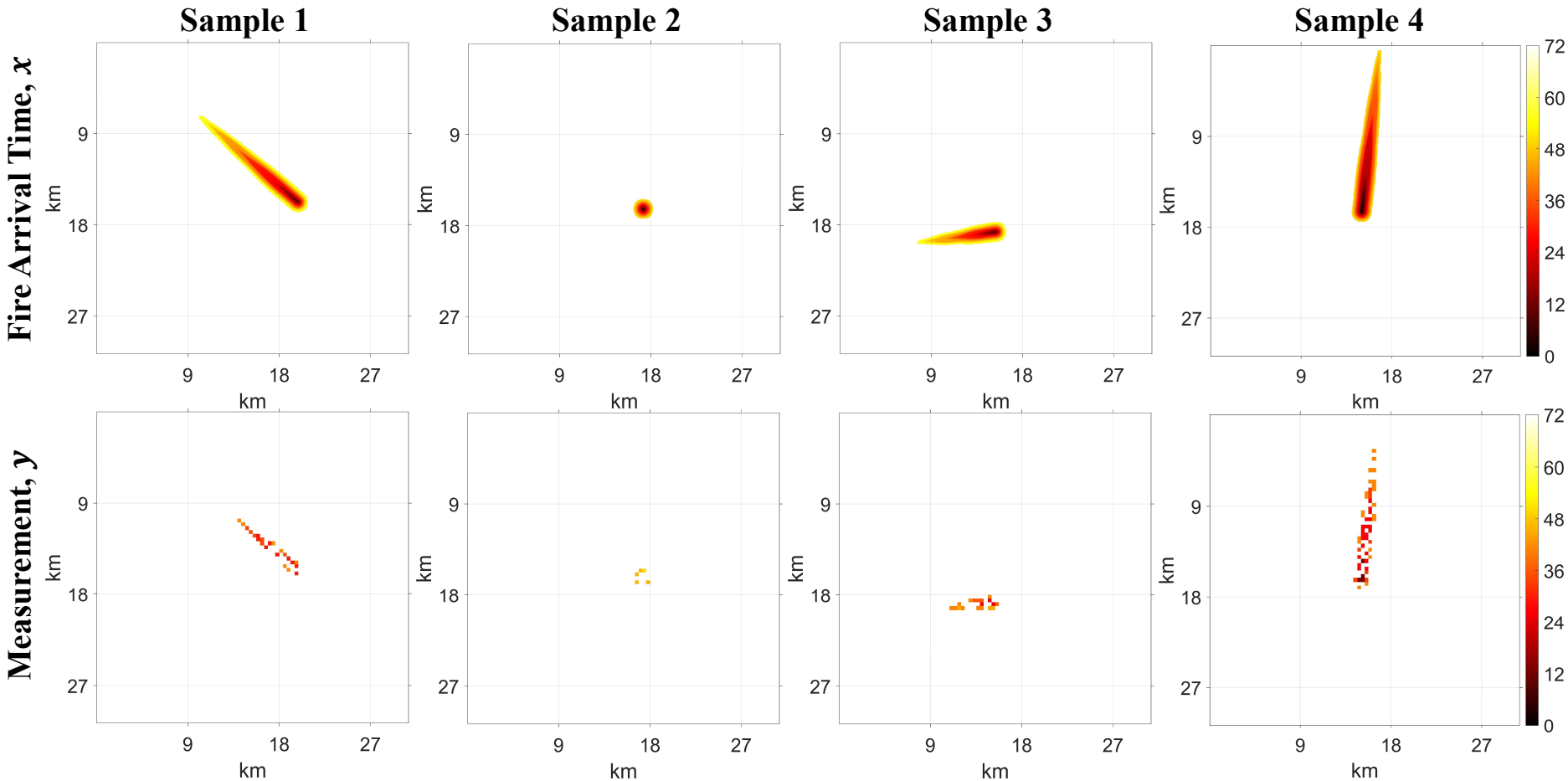
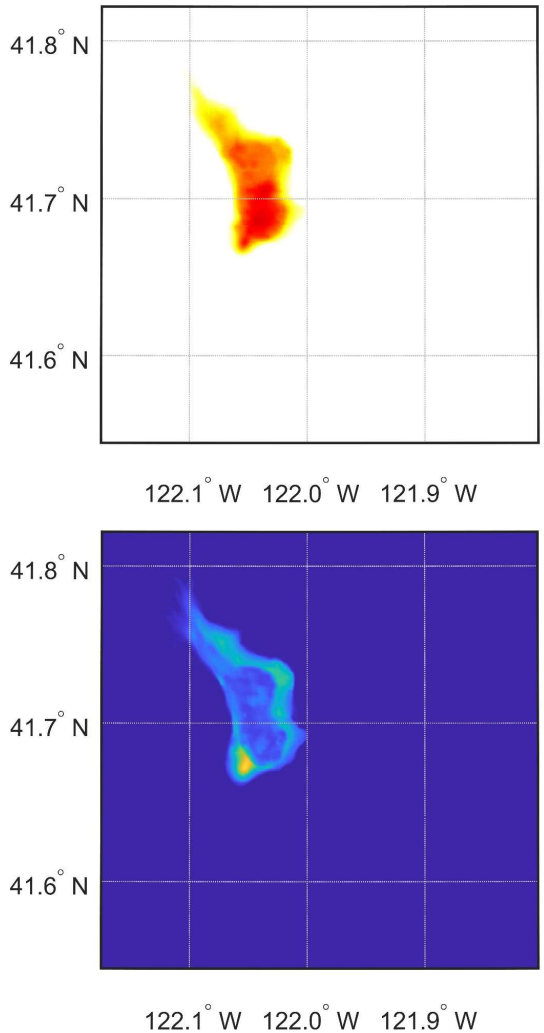


Figure: Training samples generated from WRF-SFIRE.

Example 5: Early Trajectory of a Wildfire



(b) Tennant

Figure: Fire arrival time map - mean and standard deviation.

Example 5: Early Trajectory of a Wildfire

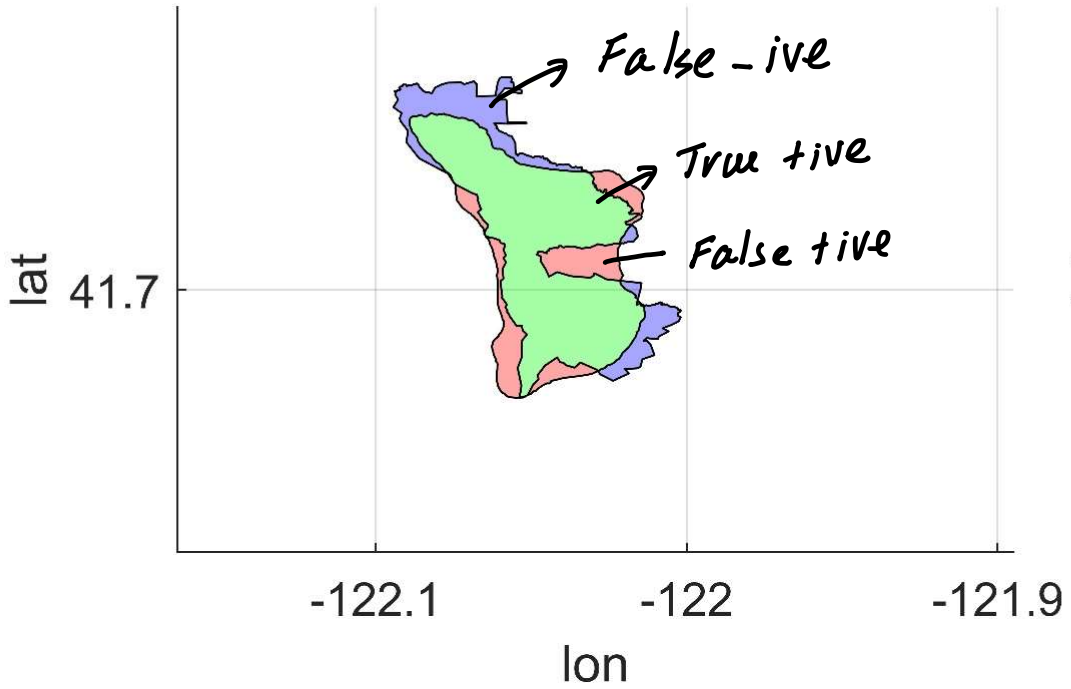


Figure: Validation against IR perimeter.

Table of Contents

Generative Diffusion networks

Key references:

- Song, Yang, and Stefano Ermon. "Generative modeling by estimating gradients of the data distribution." Advances in neural information processing systems 32 (2019).
- Hyvarinen, Aapo, and Peter Dayan. "Estimation of non-normalized statistical models by score matching." Journal of Machine Learning Research 6.4 (2005).
- Vincent, Pascal. "A connection between score matching and denoising autoencoders." Neural computation 23.7 (2011): 1661-1674.
- Batzolis, Georgios, et al. "Conditional image generation with score-based diffusion models." arXiv preprint arXiv:2111.13606 (2021).

Main idea

- Any attempt to approximate P_{data} has to ensure that the density must integrate to 1. Since N_x is large, this involves a challenging integral.
- On the other hand consider the score: $\boldsymbol{s}_{\text{data}} : \mathbb{R}^{N_x} \to \mathbb{R}^{N_x}$

$$\boldsymbol{s}_{\text{data}} = \nabla \log P_{\text{data}} = \nabla \log \left(\frac{f}{\mathbb{Z}} \right) = \nabla \left(\log f - \log \mathbb{Z} \right) = \frac{\nabla f}{f}.$$

  does not depend on the normalizing constant.
- And Lagevin equation offers a way to use the score to generate samples.
- Step 1: learn the score. Step 2: use Langevin equation to generate samples.

Step 1: Learning the score

- Let $\boldsymbol{s}(\boldsymbol{x};\boldsymbol{\theta})$ be an approximation to $\boldsymbol{s}_{\text{data}}(\boldsymbol{x})$.
- To find $\boldsymbol{\theta}$ minimize

$$L(\boldsymbol{\theta}) = \frac{1}{2}\int_{\Omega_X} |\boldsymbol{s}(\boldsymbol{x};\boldsymbol{\theta}) - \boldsymbol{s}_{\text{data}}(\boldsymbol{x})|^2 P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x}.$$

- Cannot work with this directly since $\boldsymbol{s}_{\text{data}}$ is not known. Work on this in the next page.....

Step 1: Learning the score

$$
\begin{aligned}
L(\theta) &= \frac{1}{2}\int_{\Omega_X} |\boldsymbol{s}(\boldsymbol{x};\theta) - \boldsymbol{s}_{\text{data}}(\boldsymbol{x})|^2 P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x} \\
&= \frac{1}{2}\int_{\Omega_X} |\boldsymbol{s}|^2 P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x} - \int_{\Omega_X} \boldsymbol{s}\cdot\boldsymbol{s}_{\text{data}} P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x} \\
&\quad + \frac{1}{2}\int_{\Omega_X} |\boldsymbol{s}_{\text{data}}|^2 P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x}. \\
&= \frac{1}{2}\int_{\Omega_X} |\boldsymbol{s}|^2 P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x} - \int_{\Omega_X} \boldsymbol{s}\cdot\frac{\nabla P_{\text{data}}}{P_{\text{data}}} P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x}. \\
&= \frac{1}{2}\int_{\Omega_X} |\boldsymbol{s}|^2 P_{\text{data}} d\boldsymbol{x} + \int_{\Omega_X} \nabla\cdot\boldsymbol{s} P_{\text{data}} d\boldsymbol{x} - \int_{\partial\Omega_X} \boldsymbol{s}\cdot n P_{\text{data}} d\boldsymbol{x}. \\
&= \mathbb{E}_{P_{\text{data}}}\left[\frac{1}{2}|\boldsymbol{s}|^2 + \nabla\cdot\boldsymbol{s}\right].
\end{aligned}
$$

So,

$$\begin{aligned} L(\boldsymbol{\theta}) &= \mathbb{E}_{P_{\text{data}}}\left[\frac{1}{2}|\boldsymbol{s}|^2 + \nabla \cdot \boldsymbol{s}\right] \\ &\approx \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}|\boldsymbol{s}(\boldsymbol{x}^{(i)};\boldsymbol{\theta})|^2 + \sum_{j=1}^{N_x}\frac{\partial}{\partial x_j}s_j(\boldsymbol{x}^{(i)};\boldsymbol{\theta})\right). \end{aligned}$$

- This will approximate the score, however, the approximation will be poor in regions where P_{data} is small.
- We will see in the next step we need a good approximation everywhere.

Step 2: Using the score to generate

- Given $\boldsymbol{x}_0 \sim P_0(\boldsymbol{x})$, consider the iterations

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \frac{\epsilon}{2} \nabla \log P_{\text{data}}(\boldsymbol{x}_{t-1}) + \sqrt{\epsilon} \boldsymbol{z}_t,$$

  where $\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$, and $\epsilon \to 0$ as $t \to \infty$, then in the limit $\boldsymbol{x}_t \sim P_{\text{data}}$.
- This equation is the Euler-Maruyama discretization of the Ito ODE

$$d\boldsymbol{X}_t = \frac{\epsilon}{2} \nabla \log P_{\text{data}}(\boldsymbol{x}) + \sqrt{\epsilon} \boldsymbol{z}_t,$$

  with $\boldsymbol{x}_0 \sim P_0(\boldsymbol{x})$.

Step 2: Using the score to generate

- The corresponding Fokker-Plank equation governs the evolution of the probability density of $\boldsymbol{X}_t$ and is given by,

$$\begin{aligned}\frac{\partial P_X(\boldsymbol{x},t)}{\partial t} &= -\nabla\cdot\left(\frac{\epsilon}{2}\nabla\log(P_{\text{data}}(\boldsymbol{x}))P_X(\boldsymbol{x},t)\right)\\ &\quad +\nabla\cdot\nabla\cdot\left(\frac{\epsilon}{2}\mathbf{1}P_X(\boldsymbol{x},t)\right),\end{aligned}$$

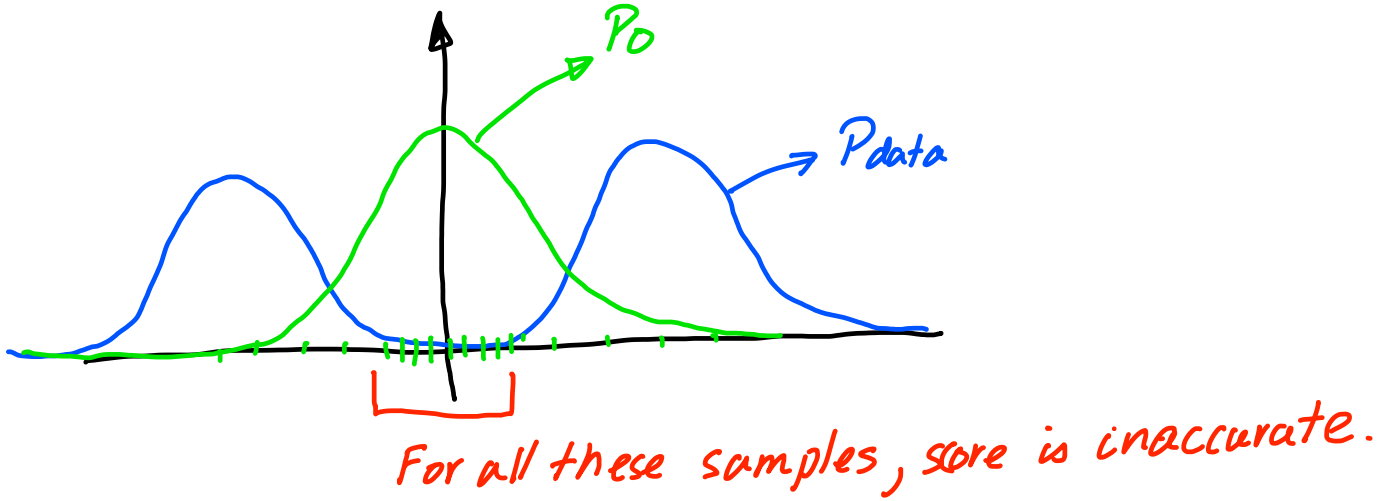
  with $P_X(\boldsymbol{x},0)\sim P_0(\boldsymbol{x})$.
- The steady state solution is given by

$$\nabla\cdot\left(-\frac{\nabla P_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x})}P_X(\boldsymbol{x},\infty)+\nabla P_X(\boldsymbol{x},\infty)\right)=0.$$

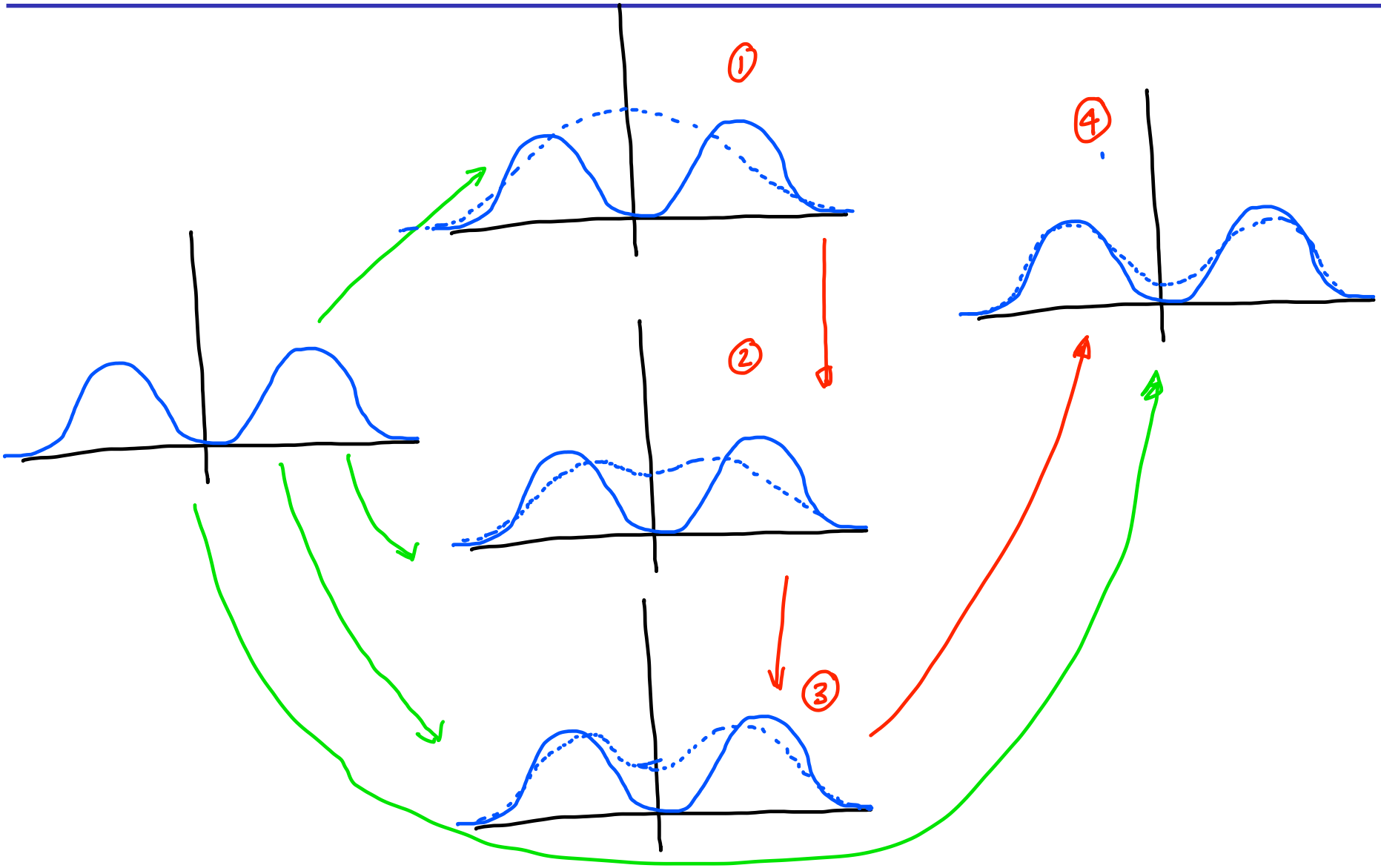
- A solution is $P_X(\boldsymbol{x},\infty)=P_{\text{data}}(\boldsymbol{x})$. It is unique!

Step 2: Using the score to generate

This would work however the approximation of the real score will be poor in regions where P_{data} is small.



Remedy



Remedy

Initially work with a smoother version P_{data} that has support near P_0.

- Consider

$$\begin{aligned} P_\sigma(\tilde{\boldsymbol{x}}) &= \int_{\Omega_X} P_{\text{data}}(\boldsymbol{x}) P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) d\boldsymbol{x} \\ P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) &= \exp(-\frac{|\tilde{\boldsymbol{x}} - x|^2}{2\sigma^2})/\mathbb{Z}. \end{aligned}$$

- Now we need a NN $\boldsymbol{s}(\tilde{\boldsymbol{x}}, \sigma; \boldsymbol{\theta})$ that will approximate

$$\boldsymbol{s}_\sigma(\tilde{\boldsymbol{x}}) = \nabla \log P_\sigma(\tilde{\boldsymbol{x}}).$$

- So we want to minimize

$$L(\boldsymbol{\theta}, \sigma) = \frac{1}{2} \int_{\Omega_X} |\boldsymbol{s}(\tilde{\boldsymbol{x}}, \sigma; \boldsymbol{\theta}) - \boldsymbol{s}_\sigma(\tilde{\boldsymbol{x}})|^2 P_\sigma(\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}}.$$

  But have access only to samples from P_{data}.

Step 1: Learning the score

$$L(\boldsymbol{\theta}, \sigma) = \int_{\Omega_X} \left(\frac{1}{2} |\boldsymbol{s}|^2 - \boldsymbol{s} \cdot \boldsymbol{s}_\sigma + \frac{1}{2} |\boldsymbol{s}_\sigma|^2 \right) P_\sigma(\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}}.$$

Consider the second term

$$\begin{aligned}
\text{2nd term} &= \int_{\Omega_X} \boldsymbol{s} \cdot \boldsymbol{s}_\sigma P_\sigma(\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} \\
&= \int_{\Omega_X} \boldsymbol{s} \cdot \nabla P_\sigma(\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} \\
&= \int_{\Omega_X} \boldsymbol{s} \cdot \nabla \Big(\int_{\Omega_X} P_{\text{data}}(\boldsymbol{x}) P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) d\boldsymbol{x} \Big) d\tilde{\boldsymbol{x}} \\
&= \int_{\Omega_X} \int_{\Omega_X} \boldsymbol{s} \cdot \nabla \Big(\log P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) \Big) P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x} d\tilde{\boldsymbol{x}}.
\end{aligned}$$

Consider the first term

$$\begin{aligned}
\text{1st term} &= \int_{\Omega_X} \frac{1}{2}|\boldsymbol{s}|^2 P_\sigma(\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} \\
&= \int_{\Omega_X} \frac{1}{2}|\boldsymbol{s}|^2 \Big(\int_{\Omega_X} P_{\text{data}}(\boldsymbol{x}) P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) d\boldsymbol{x} \Big) d\tilde{\boldsymbol{x}} \\
&= \int_{\Omega_X} \int_{\Omega_X} \frac{1}{2}|\boldsymbol{s}|^2 P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x} d\tilde{\boldsymbol{x}}.
\end{aligned}$$

Combine these two

$$
\begin{aligned}
L(\boldsymbol{\theta}, \sigma) &= \int_{\Omega_X} \int_{\Omega_X} \Big(\frac{1}{2} |\boldsymbol{s}|^2 - \boldsymbol{s} \cdot \nabla \log P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) \Big) P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x} d\tilde{\boldsymbol{x}} \\
&= \frac{1}{2} \int_{\Omega_X} \int_{\Omega_X} |\boldsymbol{s} - \nabla \log P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})|^2 P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x} d\tilde{\boldsymbol{x}} \\
&= \frac{1}{2} \int_{\Omega_X} \int_{\Omega_X} |\boldsymbol{s} + \frac{1}{\sigma^2}(\tilde{\boldsymbol{x}} - \boldsymbol{x})|^2 P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) P_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x} d\tilde{\boldsymbol{x}} \\
&\approx \frac{1}{2N} \sum_{i=1}^{N} |\boldsymbol{s}(\tilde{\boldsymbol{x}}^{(i)}, \sigma; \boldsymbol{\theta}) + \frac{1}{\sigma^2}(\tilde{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)})|^2
\end{aligned}
$$

Step 1: Learning the score

We will train $\boldsymbol{s}(\tilde{\boldsymbol{x}}, \sigma; \boldsymbol{\theta})$ for large and small values of σ. Therefore select $\sigma_j \in (\sigma_1, \cdots, \sigma_L)$ with $\frac{\sigma_{j-1}}{\sigma_j} = \rho > 1$. And then overall loss function for training the score-matching NN is

$$L(\boldsymbol{\theta}) \quad = \quad \frac{1}{2LN} \sum_{j=1}^{L} \sum_{i=1}^{N} |\boldsymbol{s}(\tilde{\boldsymbol{x}}^{(i)}, \sigma_j; \boldsymbol{\theta}) + \frac{1}{\sigma_j^2} (\tilde{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)})|^2$$

Step 2: Using the score to generate

Annealed Langevin equation:

- $\tilde{\boldsymbol{x}} \sim P_0$
- For $j = 1, \cdots, L$
- $\alpha_j = \epsilon \rho^{L-j}$
- For $t = 1, \cdots T$
  - $\tilde{\boldsymbol{x}}_t = \tilde{\boldsymbol{x}}_{t-1} + \frac{\alpha_j}{2} \boldsymbol{s}(\tilde{\boldsymbol{x}}_{t-1}, \sigma_j; \theta^*) + \sqrt{\alpha_j} \boldsymbol{z}_t$
- End (t)
- End (j)

Table of Contents

Main idea: in generative diffusion networks if we replace $P_{\text{data}}(\boldsymbol{x})$ by $P_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})$, then things would be fine. Let us re-examine Step 1 and Step 2 keeping this in mind.

Step 1: Learning the score

- Consider

$$P_\sigma(\tilde{\boldsymbol{x}}|\hat{\boldsymbol{y}}) = \int_{\Omega_X} P_{\text{data}}(\boldsymbol{x}|\boldsymbol{y}) P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) d\boldsymbol{x}$$

- Now we need a NN $\boldsymbol{s}(\tilde{\boldsymbol{x}}, \sigma, \boldsymbol{y}; \boldsymbol{\theta})$ that will approximate

$$\boldsymbol{s}_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{y}) = \nabla \log P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{y}).$$

- So we want to minimize

$$L(\boldsymbol{\theta}, \sigma) = \frac{1}{2} \int_{\Omega_Y} \int_{\Omega_X} |\boldsymbol{s}(\tilde{\boldsymbol{x}}, \sigma, \boldsymbol{y}; \boldsymbol{\theta}) - \boldsymbol{s}_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{y})|^2 P_\sigma(\tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} P_Y(\boldsymbol{y}) d\boldsymbol{y}.$$

  But have access only to samples from $P_{\text{data}}(\boldsymbol{x}, \boldsymbol{y})$.

Step 1: Learning the score

- Following the same steps....

$$
\begin{aligned}
L(\boldsymbol{\theta}, \sigma) &= \frac{1}{2} \int_{\Omega_Y} \int_{\Omega_X} \int_{\Omega_X} |\boldsymbol{s} + \frac{1}{\sigma^2}(\tilde{\boldsymbol{x}} - \boldsymbol{x})|^2 \\
&\qquad P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) P_{\text{data}}(\boldsymbol{x}|\boldsymbol{y}) P_Y(\boldsymbol{y}) d\boldsymbol{x} d\tilde{\boldsymbol{x}} d\boldsymbol{y} \\
&= \int_{\Omega_X} \int_{\Omega_Y} \int_{\Omega_X} |\boldsymbol{s} + \frac{1}{\sigma^2}(\tilde{\boldsymbol{x}} - \boldsymbol{x})|^2 \\
&\qquad P_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) P_{\text{data}}(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y} d\tilde{\boldsymbol{x}} \\
&\approx \frac{1}{2N} \sum_{i=1}^{N} |\boldsymbol{s}(\tilde{\boldsymbol{x}}^{(i)}, \sigma, \boldsymbol{y}^{(i)}; \boldsymbol{\theta}) + \frac{1}{\sigma^2}(\tilde{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)})|^2.
\end{aligned}
$$

- Step 2 remains unchanged.