Deep Learning Approaches for Inverse Problems

Deep Ray
Department of Aerospace & Mechanical Engineering
University of Southern California

Email: deepray@usc.edu
Website: deepray.github.io

ICTS-TIFR Workshop on Inverse Problems and Related Topics
October 25-29, 2021

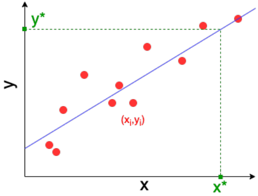
USC Viterbi
School of Engineering

What is machine learning?

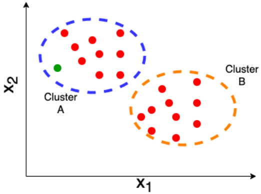
- Collect a set of data

$$\mathcal{S} = \{\boldsymbol{x}_i : 1 \le i \le n\} \quad \text{or} \quad \mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) : 1 \le i \le n\}.$$

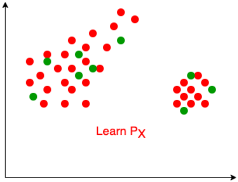
- Train an algorithm to discover patterns or relation between samples.
- Use algorithm to make future prediction on new data.



linear regression



clustering

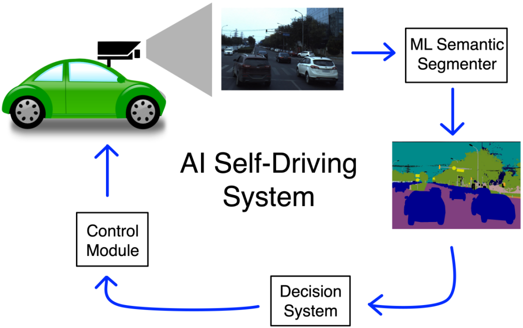


generator

Algorithms: regression methods, support vector machines, decision trees, k-means clustering, deep neural networks.

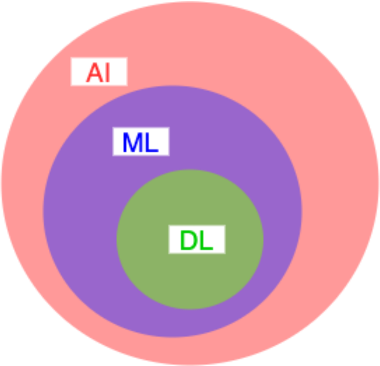
What is artificial intelligence?

- Systems with ‘human-like” intelligence.
- Machine learning + something more ...



What is artificial intelligence?

- Systems with ‘human-like” intelligence.
- Machine learning + something more ...



Outline

Outline

Multilayer perceptrons

Approximate an unknown function

$$\boldsymbol{f} : \boldsymbol{x} \in \Omega_X \mapsto \boldsymbol{y} \in \Omega_Y, \quad \Omega_X \subset \mathbb{R}^m, \; \Omega_Y \subset \mathbb{R}^n.$$

Assume we only have $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) : \boldsymbol{y}_i = \boldsymbol{f}(\boldsymbol{x}_i), 1 \leq i \leq N\}$.

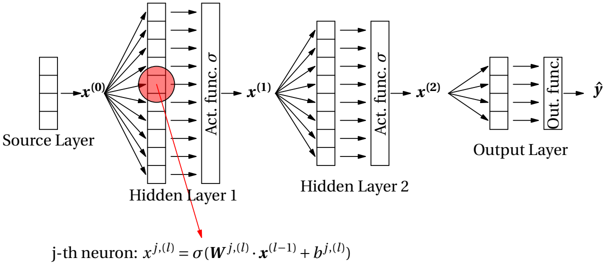
Multilayer perceptrons

Approximate an unknown function

$$\boldsymbol{f} : \boldsymbol{x} \in \Omega_X \mapsto \boldsymbol{y} \in \Omega_Y, \quad \Omega_X \subset \mathbb{R}^m,\ \Omega_Y \subset \mathbb{R}^n.$$

Assume we only have $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) : \boldsymbol{y}_i = \boldsymbol{f}(\boldsymbol{x}_i), 1 \leq i \leq N\}$.

Consider MLP with a source layer, L hidden layers and an output layer.



In layer l, $1 \leq l \leq L+1$, define

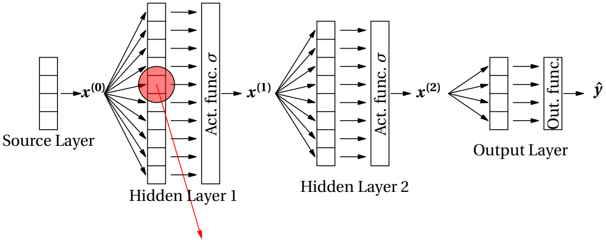
- The weight matrix $\boldsymbol{W}^{(l)}$ and bias vector $\boldsymbol{b}^{(l)}$.
- The affine transform $\mathcal{A}^{(l)}(\boldsymbol{x}^{(l-1)}) = \boldsymbol{W}^{(l)}\boldsymbol{x}^{(l-1)} + \boldsymbol{b}^{(l)}$.
- The output $\boldsymbol{x}^{(l)} = \sigma(\mathcal{A}^{(l)}(\boldsymbol{x}^{(l-1)}))$, σ applied component-wise.

Approximate an unknown function

$$\boldsymbol{f} : \boldsymbol{x} \in \Omega_X \mapsto \boldsymbol{y} \in \Omega_Y, \quad \Omega_X \subset \mathbb{R}^m, \ \Omega_Y \subset \mathbb{R}^n.$$

Assume we only have $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) : \boldsymbol{y}_i = \boldsymbol{f}(\boldsymbol{x}_i), 1 \leq i \leq N\}$.

Consider MLP with a source layer, L hidden layers and an output layer.

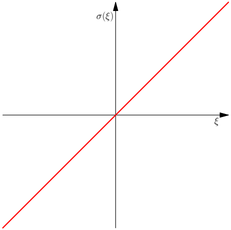


j-th neuron: $x^{j,(l)} = \sigma(\boldsymbol{W}^{j,(l)} \cdot \boldsymbol{x}^{(l-1)} + b^{j,(l)})$

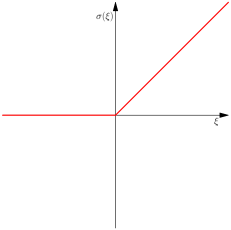
The full MLP defined by

$$\mathcal{F}(\boldsymbol{x}; \boldsymbol{\theta}) = \mathcal{A}^{(L+1)} \circ \sigma \circ \mathcal{A}^{(L)} \circ \sigma \circ \mathcal{A}^{(L-1)} \circ \cdots \circ \sigma \circ \mathcal{A}^{(1)}(\boldsymbol{x})$$

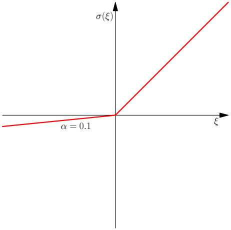
with trainable parameters $\boldsymbol{\theta} = \{\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}\}_{l=1}^{L+1} \in \mathbb{R}^{N_\theta}$.



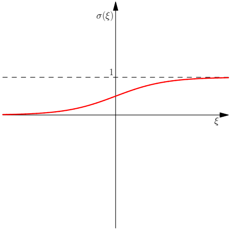
Linear: $\sigma(\xi) = \xi$



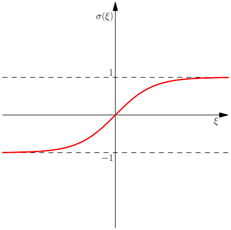
ReLU: $\sigma(\xi) = \max(0, \xi)$



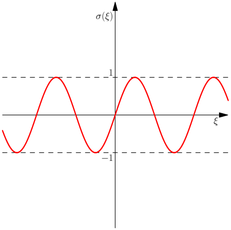
Leaky ReLU: $\sigma(\xi) = \max(0, \xi) + \alpha \min(0, \xi)$



Logistic: $\sigma(\xi) = \frac{1}{1+e^{-\xi}}$



$\sigma(\xi) = \tanh(\xi)$



$\sigma(\xi) = \sin(\xi)$

Question: What would have if we used the linear activation function?

Approximate an unknown function

$$\boldsymbol{f} : \boldsymbol{x} \in \Omega_X \mapsto \boldsymbol{y} \in \Omega_Y, \quad \Omega_X \subset \mathbb{R}^m, \, \Omega_Y \subset \mathbb{R}^n.$$

Assume we only have $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) : \boldsymbol{y}_i = \boldsymbol{f}(\boldsymbol{x}_i), 1 \leq i \leq N\}$.

Define a loss function, say MSE

$$\Pi(\boldsymbol{\theta}) = \frac{1}{N} \sum_{\substack{i=1 \\ (\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{S}}}^{N} \|\boldsymbol{y}_i - \mathcal{F}(\boldsymbol{x}_i; \boldsymbol{\theta})\|^2.$$

Train the network by solving the optimization problem – using back-propagation

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \Pi(\boldsymbol{\theta}).$$

Then $\mathcal{F}(\boldsymbol{x}; \boldsymbol{\theta}^*) \approx \boldsymbol{f}(\boldsymbol{x})$.

Also need to tune network hyper-parameters:

- Width • Depth (L) • Activation function σ • Optimizer • Stopping criteria
- Loss function • Regularization • Dataset • “A cool name for your network!"

Some remarks:

- Typically $\mathcal{S}$ is split into Training (to find $\boldsymbol{\theta}^*$), Validation (to tune hyper-parameters) and Test set.
- $\Pi(\boldsymbol{\theta})$ non-linear, non-convex – multiple re-trains with different $\boldsymbol{\theta}$ initializations.
- Training set further split into mini-batches.
- More sophisticated networks architectures available
  - Convolution neural networks – for image data.
  - Residual networks – useful for constructing deep networks.
  - U-Nets – for image-to-image tasks.
  - Autoencoders – for dimension reduction.
  - Generative Adversarial Networks (GANs) – learning distribution of data.
  - ...

Outline

Consider MLPs with L hidden layers each of width H. Let $K \subset \mathbb{R}^m$ be compact.

Theorem (Pinkus, 1999)

Let $f : K \to \mathbb{R}$ with $f \in C(K)$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be continuous non-polynomial function. Given $\epsilon > 0$, there exists an MLP $\mathcal{F}$ with a single hidden layer ($L = 1$), width H and activation function σ such that

$$\|\mathcal{F} - f\|_\infty < \epsilon.$$

Note:

- We are not assured any bound on H.
- All continuous activations shown earlier will work, except the linear activation.

Consider MLPs with L hidden layers each of width H. Let $K \subset \mathbb{R}^m$ be compact.

Theorem (Kidger and Lyons, 2020)

Let $\boldsymbol{f} : K \to \mathbb{R}^n$ with $\boldsymbol{f} \in C(K)$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be any non-affine continuous function which is continuously differentiable at some $\xi_0 \in \mathbb{R}$ with $\sigma'(\xi_0) \neq 0$. Then given $\epsilon > 0$, there exists an MLP $\mathcal{F}$ with L hidden layer each of width $H = m + n + 2$ such that

$$\|\boldsymbol{\mathcal{F}} - \boldsymbol{f}\|_\infty < \epsilon.$$

Note:

- Result holds for vector-valued functions.
- This time we have a bound on H but not on L.
- All continuous activations shown earlier will work, except the linear activation.

Consider MLPs with L hidden layers each of width H. Let $K \subset \mathbb{R}^m$ be compact.

Theorem (Yarotsky, 2021)

Let $f : K \to \mathbb{R}$ with $f \in C^{p,\alpha}(K)$. Define $r = p + \alpha$. Then, there exists an MLP with ReLU activation, width $H = 2m + 10$ and N total trainable parameters, i.e., $\boldsymbol{\theta} \in \mathbb{R}^N$, such that,

$$\|\mathcal{F} - f(\boldsymbol{x})\|_\infty < c_{r,m} \left(\frac{\log(N)}{N} \right)^{2r/m}$$

where the constant $c_{r,m}$ depends on r and m.

Note:

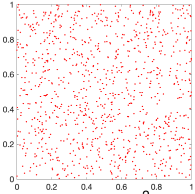
- For an error threshold ϵ, we have a bound on H and $L(\epsilon)$.
- The error is lower for higher regularity.
- The error can be is larger for higher-dimensional input domain.
- The error decay is exponential if a combination of ReLU and Sine activation are used.

Generating samples from a probability distribution

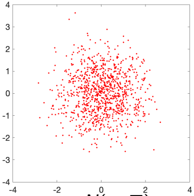
Given: A set $\mathcal{S} = \{\boldsymbol{x}_i : \boldsymbol{x}_i \in \Omega_X \subset \mathbb{R}^{N_X},\ 1 \leq i \leq n\}$ of samples from some $\mathcal{P}_X$.
Goal: Discover $\mathcal{P}_X$ from $\mathcal{S}$ and generate new samples .

Given: A set $\mathcal{S} = \{\boldsymbol{x}_i : \boldsymbol{x}_i \in \Omega_X \subset \mathbb{R}^{N_X},\ 1 \leq i \leq n\}$ of samples from some $\mathcal{P}_X$.
Goal: Discover $\mathcal{P}_X$ from $\mathcal{S}$ and generate new samples .

$\boldsymbol{x}_i \in \mathbb{R}^2$:



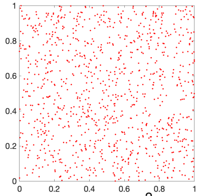
$\boldsymbol{x} \sim \mathcal{U}([0,1]^2)$



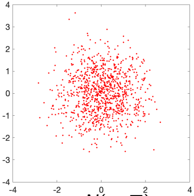
$\boldsymbol{x} \sim N(\mu, \Sigma)$

Given: A set $\mathcal{S} = \{\boldsymbol{x}_i : \boldsymbol{x}_i \in \Omega_X \subset \mathbb{R}^{N_X},\ 1 \leq i \leq n\}$ of samples from some $\mathcal{P}_X$.
Goal: Discover $\mathcal{P}_X$ from $\mathcal{S}$ and generate new samples .

$\boldsymbol{x}_i \in \mathbb{R}^2$:

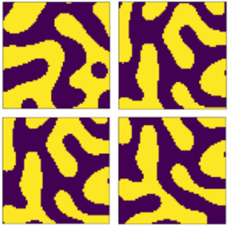


$\boldsymbol{x} \sim \mathcal{U}([0,1]^2)$

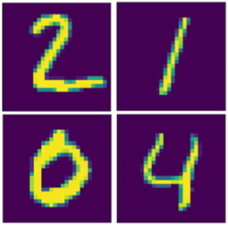


$\boldsymbol{x} \sim N(\mu, \Sigma)$

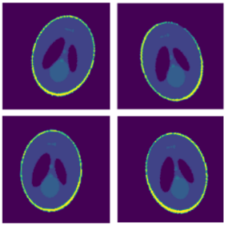
$\boldsymbol{x}_i \in \mathbb{R}^{N \times N}$:
(images)



Binary phase microstructure



Handwritten MNIST digits

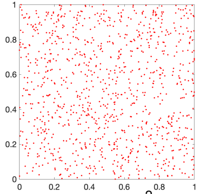


Shepp-Logan phantom

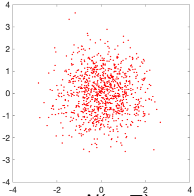
Representing this data in the form of a prior is hard!

Given: A set $\mathcal{S} = \{\boldsymbol{x}_i : \boldsymbol{x}_i \in \Omega_X \subset \mathbb{R}^{N_X},\ 1 \leq i \leq n\}$ of samples from some $\mathcal{P}_X$.
Goal: Discover $\mathcal{P}_X$ from $\mathcal{S}$ and generate new samples .

$\boldsymbol{x}_i \in \mathbb{R}^2$:



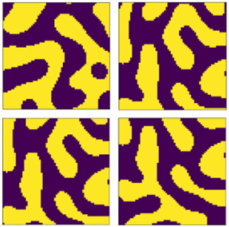
$\boldsymbol{x} \sim \mathcal{U}([0,1]^2)$



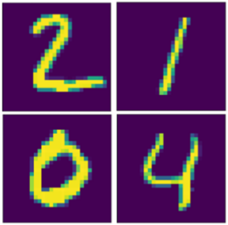
$\boldsymbol{x} \sim N(\mu, \Sigma)$

Data-driven generative algorithms

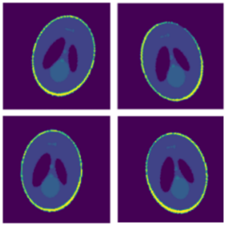
$\boldsymbol{x}_i \in \mathbb{R}^{N \times N}$:
(images)



Binary phase microstructure



Handwritten MNIST digits

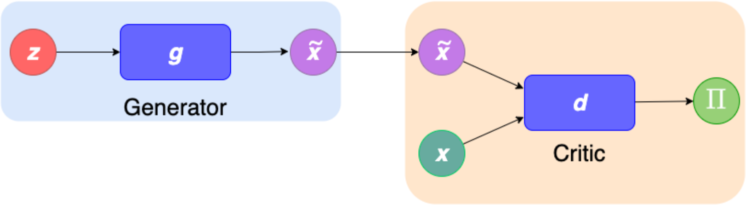


Shepp-Logan phantom

Representing this data in the form of a prior is hard!

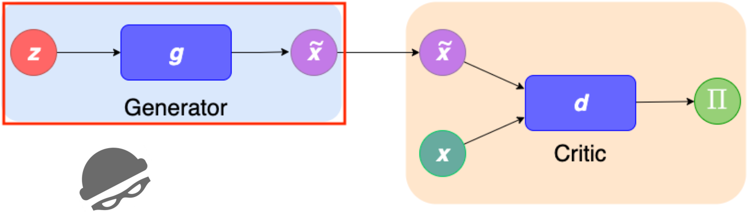
Generative adversarial network (GAN)

Designed by Goodfellow et al. (2014) to learn and sample from a target P_X.



Two networks with some suitable architectures.

Designed by Goodfellow et al. (2014) to learn and sample from a target P_X.



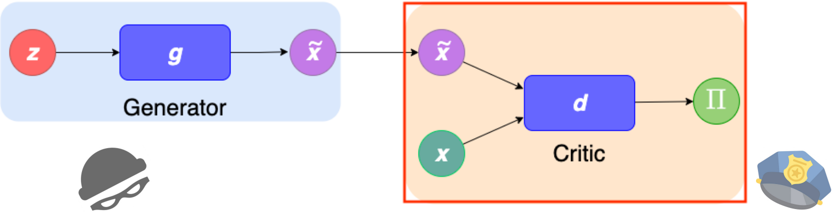
Two networks with some suitable architectures.

Generator network $\boldsymbol{g}(.;\boldsymbol{\theta})$:

- Generates fake samples $\tilde{\boldsymbol{x}}$
- $\boldsymbol{g} : \Omega_Z \to \Omega_X$.
- Latent variable $\boldsymbol{z} \in \Omega_Z \subset \mathbb{R}^{N_Z}$.
- $\boldsymbol{z} \sim P_Z$ simple distribution, e.g. Gaussian.
- $N_z \ll N_x$.

Generative adversarial network (GAN)

Designed by Goodfellow et al. (2014) to learn and sample from a target P_X.



Two networks with some suitable architectures.

Generator network $\boldsymbol{g}(.;\boldsymbol{\theta})$:

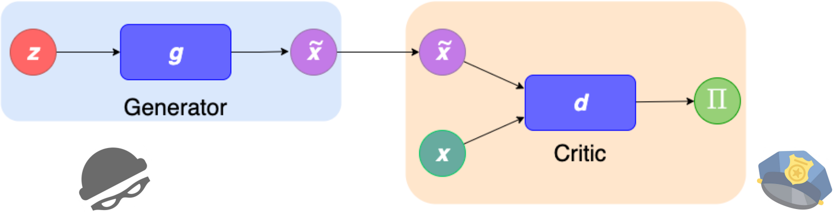
- Generates fake samples $\tilde{\boldsymbol{x}}$
- $\boldsymbol{g} : \Omega_Z \to \Omega_X$.
- Latent variable $\boldsymbol{z} \in \Omega_Z \subset \mathbb{R}^{N_z}$.
- $\boldsymbol{z} \sim P_Z$ simple distribution, e.g. Gaussian.
- $N_z \ll N_x$.

Critic network $d(.;\phi)$:

- Distinguishes fake samples from real
- $d : \Omega_X \to \mathbb{R}$.
- $\boldsymbol{x} \sim P_X$.
- $d(\boldsymbol{x})$ large for $\boldsymbol{x} \sim P_X$, small otherwise.

Generative adversarial network (GAN)

Designed by Goodfellow et al. (2014) to learn and sample from a target P_X.



Two networks with some suitable architectures.

Generator network $\boldsymbol{g}(.;\boldsymbol{\theta})$:

- Generates fake samples $\tilde{\boldsymbol{x}}$
- $\boldsymbol{g} : \Omega_Z \to \Omega_X$.
- Latent variable $\boldsymbol{z} \in \Omega_Z \subset \mathbb{R}^{N_Z}$.
- $\boldsymbol{z} \sim P_Z$ simple distribution, e.g. Gaussian.
- $N_z \ll N_x$.

Critic network $d(.;\phi)$:

- Distinguishes fake samples from real
- $d : \Omega_X \to \mathbb{R}$.
- $\boldsymbol{x} \sim P_X$.
- $d(\boldsymbol{x})$ large for $\boldsymbol{x} \sim P_X$, small otherwise.

For a metric $\mathcal{M}$ on $\mathcal{P}(\Omega_X)$, define the loss

$$\Pi(\boldsymbol{g}, d) := \Pi(\boldsymbol{\theta}, \phi) = \mathcal{M}(P_X, g_{\#}P_Z).$$

Solve the MinMax problem

$$(\boldsymbol{g}^*, d^*) = \arg\min_{\boldsymbol{g}} \arg\max_{d} \Pi(\boldsymbol{g}, d) \quad \longrightarrow \quad \text{Adversarial Training}$$

Proposed by Arjovsky et al. (2017), using the Wasserstein-1 metric

$$W_1(P_1, P_2) = \inf_{\gamma \in J(P_1, P_2)} \mathop{\mathbb{E}}_{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \gamma} [\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|]$$

Using the Kantorovich-Rubinstein dual characterization, we have

$$W_1(P_1, P_2) = \sup_{\|f\|_{\text{Lip}} \leq 1} \left(\mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_1} [f(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_2} [f(\boldsymbol{x})] \right)$$

Wasserstein GAN

Proposed by Arjovsky et al. (2017), using the Wasserstein-1 metric

$$W_1(P_1, P_2) = \inf_{\gamma \in J(P_1, P_2)} \mathop{\mathbb{E}}_{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \gamma} [\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|]$$

Using the Kantorovich-Rubinstein dual characterization, we have

$$W_1(P_1, P_2) = \sup_{\|f\|_{\text{Lip}} \leq 1} \left(\mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_1} [f(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_2} [f(\boldsymbol{x})] \right)$$

Set the loss function as

$$\Pi(\boldsymbol{g}, d) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_X} [d(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_Z} [d(\boldsymbol{g}(\boldsymbol{z}))]$$

Under the constraint $\|d\|_{\text{Lip}} \leq 1$, find

$$d^*(\boldsymbol{g}) = \arg\max_d \Pi(\boldsymbol{g}, d) = W_1(P_X, \boldsymbol{g}_\# P_Z)$$

Wasserstein GAN

Proposed by Arjovsky et al. (2017), using the Wasserstein-1 metric

$$W_1(P_1, P_2) = \inf_{\gamma \in J(P_1, P_2)} \mathop{\mathbb{E}}_{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \gamma} [\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|]$$

Using the Kantorovich-Rubinstein dual characterization, we have

$$W_1(P_1, P_2) = \sup_{\|f\|_{\text{Lip}} \leq 1} \left(\mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_1} [f(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_2} [f(\boldsymbol{x})] \right)$$

Set the loss function as

$$\Pi(\boldsymbol{g}, d) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_X} [d(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_Z} [d(\boldsymbol{g}(\boldsymbol{z}))]$$

Under the constraint $\|d\|_{\text{Lip}} \leq 1$, find

$$d^*(\boldsymbol{g}) = \arg\max_{d} \Pi(\boldsymbol{g}, d) = W_1(P_X, \boldsymbol{g}_\# P_Z)$$

Thus, for the optimal generator $\boldsymbol{g}^*$

$$\boldsymbol{g}^* = \arg\min_{\boldsymbol{g}} W_1(P_X, \boldsymbol{g}_\# P_Z)$$

Proposed by Arjovsky et al. (2017), using the Wasserstein-1 metric

$$W_1(P_1, P_2) = \inf_{\gamma \in J(P_1, P_2)} \mathop{\mathbb{E}}_{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \gamma} [\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|]$$

Using the Kantorovich-Rubinstein dual characterization, we have

$$W_1(P_1, P_2) = \sup_{\|f\|_{\text{Lip}} \leq 1} \left(\mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_1} [f(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_2} [f(\boldsymbol{x})] \right)$$

Set the loss function as

$$\Pi(\boldsymbol{g}, d) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_X} [d(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_Z} [d(\boldsymbol{g}(\boldsymbol{z}))]$$

Under the constraint $\|d\|_{\text{Lip}} \leq 1$, find

$$d^*(\boldsymbol{g}) = \arg\max_{d} \Pi(\boldsymbol{g}, d) = W_1(P_X, \boldsymbol{g}_\# P_Z)$$

Thus, for the optimal generator $\boldsymbol{g}^*$

$$\boldsymbol{g}^* = \arg\min_{\boldsymbol{g}} W_1(P_X, \boldsymbol{g}_\# P_Z)$$

Finally, convergence in W_1 implies weak convergence of measures

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_X} [\ell(\boldsymbol{x})] = \mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_Z} [\ell(\boldsymbol{g}^*(\boldsymbol{z}))], \quad \forall\, \ell \in C_b(\Omega_X)$$

$\longrightarrow$ moments converge.

In practice, at the discrete level

- Generate/obtain the finite dataset $\mathcal{S} = \{\boldsymbol{x}_i : \boldsymbol{x}_i \in \Omega_X,\ 1 \leq i \leq n\}$.
- Compute expectations using Monte Carlo

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_X}[d(\boldsymbol{x})] \approx \frac{1}{n}\sum_{i=1}^{n} d(\boldsymbol{x}_i), \quad \mathop{\mathbb{E}}_{\boldsymbol{z}\sim P_Z}[d(\boldsymbol{g}(\boldsymbol{z}))] \approx \frac{1}{n}\sum_{i=1,\boldsymbol{z}_i\sim P_Z}^{n} d(\boldsymbol{g}(\boldsymbol{z}_i))$$

- Iterative solve the MinMax problem:
  - Take N (typically $N \geq 4$) optimization steps for d
  - Take 1 optimization step for $\boldsymbol{g}$
- Add a gradient penalty term (Gulrajani, 2017) to constraint d to be 1-Lipschitz

$$\lambda\frac{1}{n}\sum_{j=1}^{n}(\|\nabla_{\boldsymbol{x}} d(\boldsymbol{x}_j)\| - 1)^2$$

Results by Karras et al. (2018) from NVIDIA.
CELEBA-HQ dataset, $N_z = 512$, $N_x = 1024 \times 1024 \times 3 = 3.14 \times 10^6$
$\longrightarrow$ dimension reduction!

Outline

"We call two problems inverses of one another if the formulation of each involves all or part of the solution of the other. Often, for historical reasons, one of the two problems has been studied extensively for sometime, while the other is newer and not so well understood. In such cases, the former is called the direct problem, while the latter is called the inverse problem."

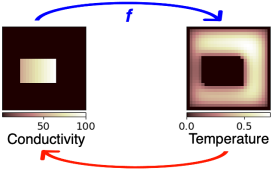
– Joseph Keller, 1976

Consider the elliptic PDE for the (steady-state) temperature field u with conductivity κ

$$\begin{aligned} -\nabla \cdot (\kappa \nabla u) &= b(\boldsymbol{\xi}), && \forall\, \boldsymbol{\xi} \in \Omega \\ u(\boldsymbol{\xi}) &= 0, && \forall\, \boldsymbol{\xi} \in \partial\Omega \end{aligned}$$

Direct: Given {PDE, b, κ} $\xrightarrow{f}$ u

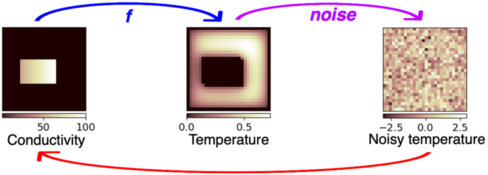
Inverse: Given {PDE, b, u} $\xrightarrow{f^{-1}}$ κ



Consider the elliptic PDE for the (steady-state) temperature field u with conductivity κ

$$-\nabla \cdot (\kappa \nabla u) = b(\boldsymbol{\xi}), \qquad \forall\, \boldsymbol{\xi} \in \Omega$$
$$u(\boldsymbol{\xi}) = 0, \qquad \forall\, \boldsymbol{\xi} \in \partial\Omega$$

Direct: Given {PDE, b, κ} $\xrightarrow{f}$ u Inverse: Given {PDE, b, u} $\xrightarrow{f^{-1}}$ κ



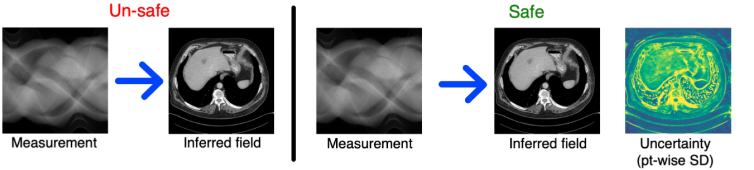
Challenges with inverse problems:

- Inverse map is not well posed.
- Noisy measurements from direct problem.
- Need to encode prior knowledge about inferred field.

Two approaches: regularization and Bayesian inference.

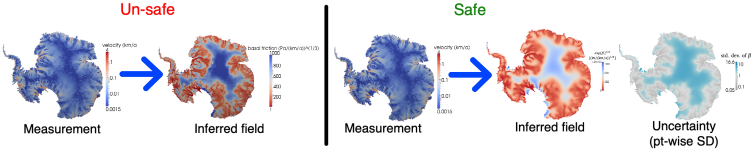
Uncertainty in inferred field critical for applications with high-stake decisions.

Example: Medical imaging to detect liver lesions



[Adler et al., 2018]

Example: Inferring basal sliding friction from surface ice velocity of Antarctic ice-shelf



[Issac et al., 2015]

Notations: We assume all quantities are discretized on some grid

- Parameter we wish to infer $\boldsymbol{x} \in \Omega_X \subset \mathbb{R}^{N_x}$ (e.g. κ on N_x grid points).
- Measured response from direct problem $\boldsymbol{y} \in \Omega_Y \subset \mathbb{R}^{N_y}$ (e.g. u on N_y grid points).
- Direct map $\boldsymbol{f} : \Omega_X \to \Omega_Y$ (e.g. discrete PDE solver). Sometimes,

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\eta} \quad \rightarrow \quad \text{(additive noise)}$$

  where $\boldsymbol{\eta}$ is noise with distribution P_η.
- Assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ are modelled using random variables X and Y.

Bayesian formulation

Bayes theorem gives us:

$$P_{X|Y}(\boldsymbol{x}|\boldsymbol{y}) = \frac{P_{Y|X}(\boldsymbol{y}|\boldsymbol{x})P_X(\boldsymbol{x})}{P_Y(\boldsymbol{y})}$$

We apply this to the inverse problem: given a measurement $\boldsymbol{y}$ and prior information, infer $\boldsymbol{x}$

Bayesian formulation

Bayes theorem gives us:

$$P_{X|Y}(\boldsymbol{x}|\boldsymbol{y}) = \frac{P_{Y|X}(\boldsymbol{y}|\boldsymbol{x})P_X(\boldsymbol{x})}{P_Y(\boldsymbol{y})}$$

We apply this to the inverse problem: given a measurement $\boldsymbol{y}$ and prior information, infer $\boldsymbol{x}$

- $P_X(\boldsymbol{x}) = P_X^{\text{prior}}(\boldsymbol{x})$: prior distribution, obtained from samples or other constraints.

Bayesian formulation

Bayes theorem gives us:

$$P_{X|Y}(\boldsymbol{x}|\boldsymbol{y}) = \frac{P_{Y|X}(\boldsymbol{y}|\boldsymbol{x})P_X(\boldsymbol{x})}{P_Y(\boldsymbol{y})}$$

We apply this to the inverse problem: given a measurement $\boldsymbol{y}$ and prior information, infer $\boldsymbol{x}$

- $P_X(\boldsymbol{x}) = P_X^{\text{prior}}(\boldsymbol{x})$: prior distribution, obtained from samples or other constraints.
- $P_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) = P_Y^{\text{like}}(\boldsymbol{y}|\boldsymbol{x})$: the likelihood of observing the measurement $\boldsymbol{y}$ given $\boldsymbol{x}$. For additive noise

$$P_Y^{\text{like}}(\boldsymbol{y}|\boldsymbol{x}) = P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x})) \quad \rightarrow \quad \text{(embedding physics)}.$$

Bayesian formulation

Bayes theorem gives us:

$$P_{X|Y}(\boldsymbol{x}|\boldsymbol{y}) = \frac{P_{Y|X}(\boldsymbol{y}|\boldsymbol{x})P_X(\boldsymbol{x})}{P_Y(\boldsymbol{y})}$$

We apply this to the inverse problem: given a measurement $\boldsymbol{y}$ and prior information, infer $\boldsymbol{x}$

- $P_X(\boldsymbol{x}) = P_X^{\text{prior}}(\boldsymbol{x})$: prior distribution, obtained from samples or other constraints.
- $P_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) = P_Y^{\text{like}}(\boldsymbol{y}|\boldsymbol{x})$: the likelihood of observing the measurement $\boldsymbol{y}$ given $\boldsymbol{x}$. For additive noise

$$P_Y^{\text{like}}(\boldsymbol{y}|\boldsymbol{x}) = P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x})) \quad \rightarrow \quad \text{(embedding physics)}.$$

- $P_Y(\boldsymbol{y}) = Q$: the evidence/normalizing term

$$Q = \int P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}))P_X^{\text{prior}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \quad \rightarrow \quad \text{(hard to compute when } N_x \gg 1\text{)}.$$

Bayesian formulation

Bayes theorem gives us:

$$P_{X|Y}(\boldsymbol{x}|\boldsymbol{y}) = \frac{P_{Y|X}(\boldsymbol{y}|\boldsymbol{x})P_X(\boldsymbol{x})}{P_Y(\boldsymbol{y})}$$

We apply this to the inverse problem: given a measurement $\boldsymbol{y}$ and prior information, infer $\boldsymbol{x}$

- $P_X(\boldsymbol{x}) = P_X^{\text{prior}}(\boldsymbol{x})$: prior distribution, obtained from samples or other constraints.
- $P_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) = P_Y^{\text{like}}(\boldsymbol{y}|\boldsymbol{x})$: the likelihood of observing the measurement $\boldsymbol{y}$ given $\boldsymbol{x}$. For additive noise

  $$P_Y^{\text{like}}(\boldsymbol{y}|\boldsymbol{x}) = P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x})) \quad \rightarrow \quad \text{(embedding physics).}$$

- $P_Y(\boldsymbol{y}) = Q$: the evidence/normalizing term

  $$Q = \int P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}))P_X^{\text{prior}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \quad \rightarrow \quad \text{(hard to compute when } N_x \gg 1\text{).}$$

- $P_{X|Y}(\boldsymbol{x}|\boldsymbol{y}) = P_X^{\text{post}}(\boldsymbol{x}|\boldsymbol{y})$: the posterior distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$.

Bayes theorem gives us:

$$P_{X|Y}(\boldsymbol{x}|\boldsymbol{y}) = \frac{P_{Y|X}(\boldsymbol{y}|\boldsymbol{x})P_X(\boldsymbol{x})}{P_Y(\boldsymbol{y})}$$

We apply this to the inverse problem: given a measurement $\boldsymbol{y}$ and prior information, infer $\boldsymbol{x}$

- $P_X(\boldsymbol{x}) = P_X^{\text{prior}}(\boldsymbol{x})$: prior distribution, obtained from samples or other constraints.
- $P_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) = P_Y^{\text{like}}(\boldsymbol{y}|\boldsymbol{x})$: the likelihood of observing the measurement $\boldsymbol{y}$ given $\boldsymbol{x}$. For additive noise

$$P_Y^{\text{like}}(\boldsymbol{y}|\boldsymbol{x}) = P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x})) \quad \rightarrow \quad \text{(embedding physics)}.$$

- $P_Y(\boldsymbol{y}) = Q$: the evidence/normalizing term

$$Q = \int P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}))P_X^{\text{prior}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \quad \rightarrow \quad \text{(hard to compute when } N_x \gg 1\text{)}.$$

- $P_{X|Y}(\boldsymbol{x}|\boldsymbol{y}) = P_X^{\text{post}}(\boldsymbol{x}|\boldsymbol{y})$: the posterior distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$.

Bayesian formulation:

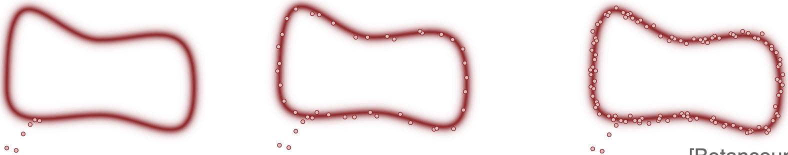
$$P_X^{\text{post}}(\boldsymbol{x}|\boldsymbol{y}) = \frac{P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}))P_X^{\text{prior}}(\boldsymbol{x})}{Q} \propto P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}))P_X^{\text{prior}}(\boldsymbol{x})$$

Posterior distribution

$$P_X^{\text{post}}(\boldsymbol{x}|\boldsymbol{y}) \propto P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}))P_X^{\text{prior}}(\boldsymbol{x})$$

Steps:

- Construct/obtain an explicit expression for P_X^{prior}.
- For a given $\boldsymbol{y}$, use Markov Chain Monte Carlo (MCMC) to sample from P_X^{post}.
  - Generate a Markov chain whose stationary distribution is P_X^{post}.
  - Need to burn the first part of the chain.



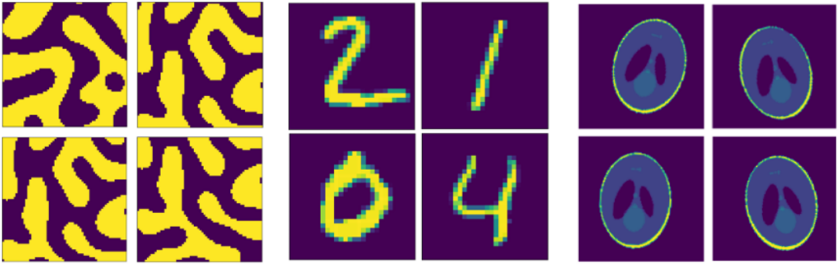
[Betancourt]

One could also use variational inference, which would find the best approximation of P_X^{post} among a parametrised family.

- MCMC is prohibitively expensive when N_x is large.
- Characterization of priors for complex data.

Typical Gaussian prior $P_X^{\text{prior}}(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{|\boldsymbol{x}|^2}{2\sigma^2}\right)$

However, prior knowledge may be samples like:



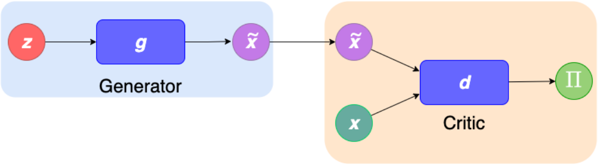
Representing this data in the form of a prior is hard!

Resolve both issues using GANs

Outline

Recall the WGAN

Learn and sample from a target P_X.



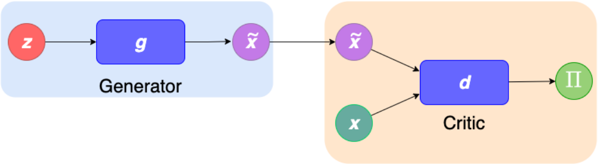
Generator network $\boldsymbol{g}(.;\boldsymbol{\theta})$:

- Generates fake samples $\tilde{\boldsymbol{x}}$
- $\boldsymbol{g} : \Omega_Z \to \Omega_X$.
- Latent variable $\boldsymbol{z} \in \Omega_Z \subset \mathbb{R}^{N_Z}$.
- $\boldsymbol{z} \sim P_Z$ simple distribution, e.g. Gaussian.
- $N_z \ll N_x$ (dimension reduction)

Critic network $d(.;\phi)$:

- Distinguishes fake samples from real
- $d : \Omega_X \to \mathbb{R}$.
- $\boldsymbol{x} \sim P_X$.
- $d(\boldsymbol{x})$ large for $\boldsymbol{x} \sim P_X$, small otherwise.

Learn and sample from a target P_X.



Generator network $\boldsymbol{g}(.;\boldsymbol{\theta})$:

- Generates fake samples $\tilde{\boldsymbol{x}}$
- $\boldsymbol{g}:\Omega_Z \to \Omega_X$.
- Latent variable $\boldsymbol{z} \in \Omega_Z \subset \mathbb{R}^{N_Z}$.
- $\boldsymbol{z} \sim P_Z$ simple distribution, e.g. Gaussian.
- $N_z \ll N_x$ (dimension reduction)

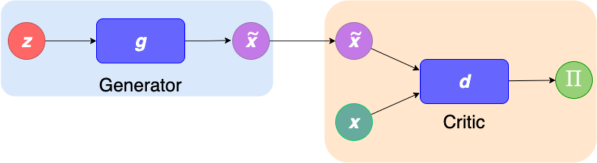
Critic network $d(.;\phi)$:

- Distinguishes fake samples from real
- $d:\Omega_X \to \mathbb{R}$.
- $\boldsymbol{x} \sim P_X$.
- $d(\boldsymbol{x})$ large for $\boldsymbol{x} \sim P_X$, small otherwise.

Solve the MinMax problem:

$$(\boldsymbol{g}^*, d^*) = \arg\min_{\boldsymbol{g}} \arg\max_{d} \Pi(\boldsymbol{g}, d) = \arg\min_{\boldsymbol{g}} \arg\max_{d} \left(\mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_X} [d(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{z}\sim P_Z} [d(\boldsymbol{g}(\boldsymbol{z}))] \right)$$

Learn and sample from a target P_X.



Generator network $\boldsymbol{g}(.;\boldsymbol{\theta})$:

- Generates fake samples $\tilde{\boldsymbol{x}}$
- $\boldsymbol{g}:\Omega_Z \rightarrow \Omega_X$.
- Latent variable $\boldsymbol{z} \in \Omega_Z \subset \mathbb{R}^{N_Z}$.
- $\boldsymbol{z} \sim P_Z$ simple distribution, e.g. Gaussian.
- $N_z \ll N_x$ (dimension reduction)

Critic network $d(.;\boldsymbol{\phi})$:

- Distinguishes fake samples from real
- $d:\Omega_X \rightarrow \mathbb{R}$.
- $\boldsymbol{x} \sim P_X$.
- $d(\boldsymbol{x})$ large for $\boldsymbol{x} \sim P_X$, small otherwise.

Solve the MinMax problem:

$$(\boldsymbol{g}^*, d^*) = \arg\min_{\boldsymbol{g}} \arg\max_{d} \Pi(\boldsymbol{g}, d) = \arg\min_{\boldsymbol{g}} \arg\max_{d} \left(\mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_X}[d(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{z}\sim P_Z}[d(\boldsymbol{g}(\boldsymbol{z}))] \right)$$

Convergence in $W_1 \implies$ weak convergence

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim P_X}[\ell(\boldsymbol{x})] = \mathop{\mathbb{E}}_{\boldsymbol{z}\sim P_Z}\left[\ell(\boldsymbol{g}^*(\boldsymbol{z}))\right], \quad \forall\, \ell \in C_b(\Omega_X)$$

$\longrightarrow$ moments converge.

Given:

- A set $\mathcal{S} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, where $\boldsymbol{x}_i \sim P_X^{\text{prior}}$.
- The direct map $\boldsymbol{f}(\boldsymbol{x})$ (exactly or approximately).
- The noise distribution P_η
- A noisy measurement $\boldsymbol{y}$

Goal: Determine P_X^{post} and evaluate statistics w.r.t. it.

• *GAN-based Priors for Uncertainty Quantification*, by Patel & Oberai, SIAM/ASA Journal on Uncertainty Quantification 9(3):1314-1343, 2021.

• *Solution of Physics-based Bayesian Inverse Problems with Deep Generative Priors*, by Patel, Ray & Oberai, arXiv:2107.02926, 2021.

GAN as prior

Step 1: Using $\mathcal{S}$, train a WGAN with generator $\boldsymbol{g}^*$.

Assume:

- $\boldsymbol{g}^*$ is the optimal generator satisfying the weak relation

$$\underset{\boldsymbol{x}\sim P_X^{\text{prior}}}{\mathbb{E}}[\ell(\boldsymbol{x})] = \underset{\boldsymbol{z}\sim P_Z}{\mathbb{E}}[\ell(\boldsymbol{g}^*(\boldsymbol{z}))], \quad \forall\, \ell \in C_b(\Omega_X).$$

- $\boldsymbol{f}$ and P_η are continuous.

Choose

$$\ell(\boldsymbol{x}) = \frac{1}{Q}\hat{\ell}(\boldsymbol{x})P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x})), \quad \hat{\ell} \in C_b(\Omega_X).$$

Then, we can get an expression for P_X^{post}

$$\underset{\boldsymbol{x}\sim P_X^{\text{prior}}}{\mathbb{E}}\left[\frac{1}{Q}\hat{\ell}(\boldsymbol{x})P_\eta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}))\right] = \underset{\boldsymbol{z}\sim P_Z}{\mathbb{E}}\left[\frac{1}{Q}\hat{\ell}(\boldsymbol{g}^*(\boldsymbol{z}))P_\eta\Big(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{g}^*(\boldsymbol{z}))\Big)\right]$$

$$\Longrightarrow \underset{\boldsymbol{x}\sim P_X^{\text{post}}}{\mathbb{E}}\left[\hat{\ell}(\boldsymbol{x})\right] = \underset{\boldsymbol{z}\sim P_Z^{\text{post}}}{\mathbb{E}}\left[\hat{\ell}(\boldsymbol{g}^*(\boldsymbol{z}))\right]$$

where

$$P_Z^{\text{post}}(\boldsymbol{z}|\boldsymbol{y}) = \frac{1}{Q}P_\eta\Big(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{g}^*(\boldsymbol{z}))\Big)P_Z(\boldsymbol{z}) \propto P_\eta\Big(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{g}^*(\boldsymbol{z}))\Big)P_Z(\boldsymbol{z})$$

Sampling $\boldsymbol{x}$ from $P_X^{\text{post}} \equiv$ sampling $\boldsymbol{z}$ from P_Z^{post} and evaluating $\boldsymbol{x} = \boldsymbol{g}^*(\boldsymbol{z})$.

Step 2: Generate an MCMC approximation $P_Z^{\text{mcmc}}(\boldsymbol{z}|\boldsymbol{y}) \approx P_Z^{\text{post}}(\boldsymbol{z}|\boldsymbol{y})$.

Step 3: Evaluate statistics using Monte Carlo

$$\mathbb{E}_{\boldsymbol{x}\sim P_X^{\text{post}}}[\ell(\boldsymbol{x})] \approx \frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} \ell(\boldsymbol{g}^*(\boldsymbol{z}))), \quad \boldsymbol{z} \sim P_Z^{\text{mcmc}}(\boldsymbol{z}|\boldsymbol{y}).$$

What do we gain?

- Ability to represent complex prior, if $\mathcal{S}$ is available.
- $N_z \ll N_x$ makes MCMC computational tractable.

Inferring thermal conductivity

Given u, find κ satisfying

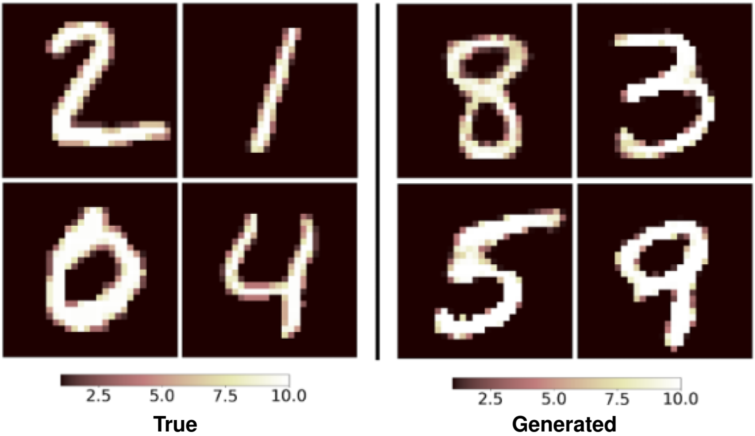
$$-\nabla \cdot (\kappa \nabla u) = b(\boldsymbol{\xi}), \qquad \forall\, \boldsymbol{\xi} \in \Omega \subset \mathbb{R}^2$$
$$u(\boldsymbol{\xi}) = 0, \qquad \forall\, \boldsymbol{\xi} \in \partial\Omega$$

Problem setup:

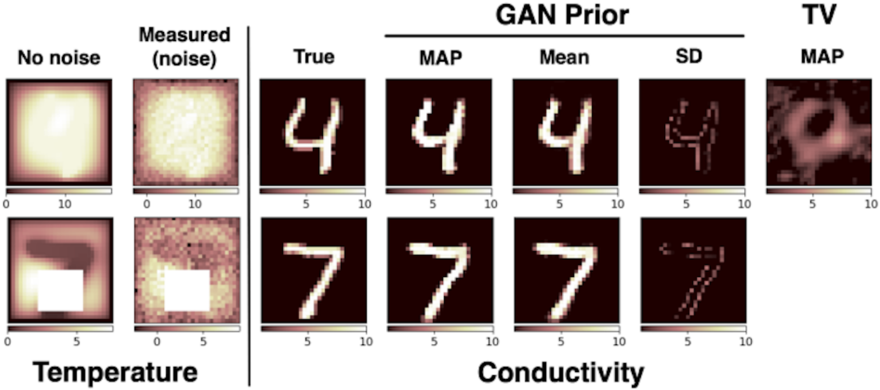
- Measurement $\boldsymbol{y}$, noisy temperature field u on a 2D grid.
- Infer $\boldsymbol{x}$, nodal values of conductivity κ.
- Non-linear forward map $\boldsymbol{f}$ solves the PDE. Implemented in Fenics.
- Noise is assumed to be Gaussian iid.

Inferring thermal conductivity (MNIST)

Assume that κ is given by MNIST digits ($N_x = 784$, $N_Z = 100$)



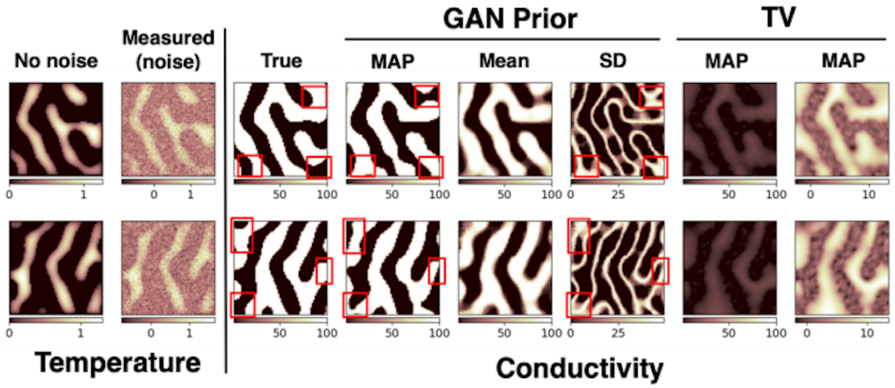
Solving the inference problem on test data



Inferring thermal conductivity (microstructure)

Microstructure profile given by Cahn-Hilliard ($N_x = 4096, N_Z = 100$)

Solving the inference problem on test data



Inverse Radon transform (CT)

Find the tissue density $\rho : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ given the line Radon transforms

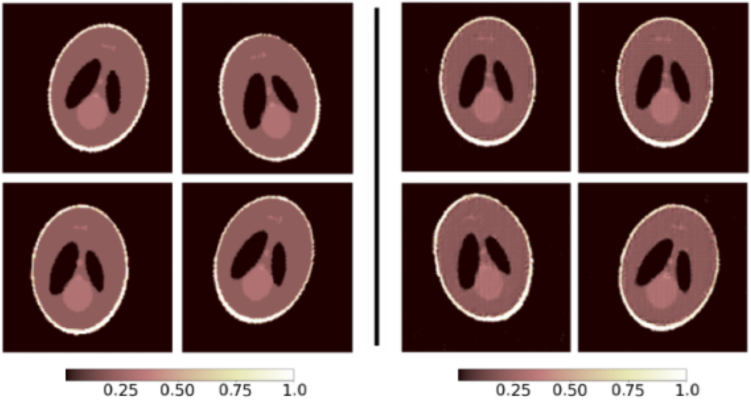
$$\mathcal{R}_{t,\psi} = \int_{\gamma_{t,\psi}} \rho \mathrm{d}\gamma$$

where $\gamma_{t,\psi}$ is the line at an angle ψ and at a signed-distance of t from the center of Ω.

Problem setup:

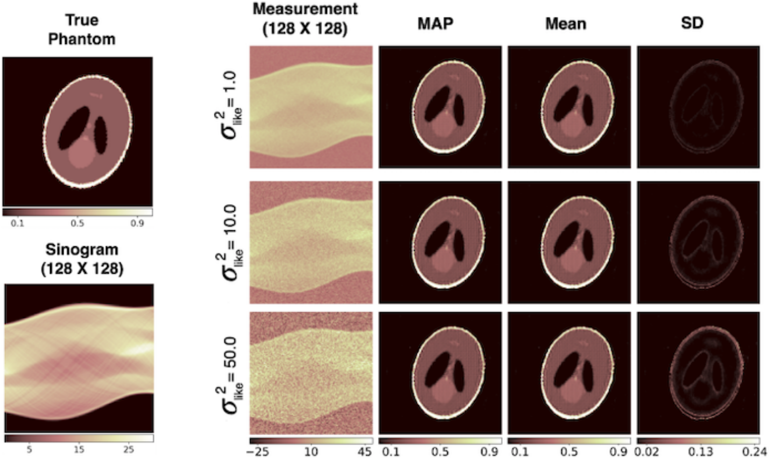
- Infer $\boldsymbol{x}$, nodal values of ρ.
- Linear forward map $\boldsymbol{f}$, Radon transform.
- Measurement $\boldsymbol{y}$, noisy Radon transforms on a set of lines.
- Noise is assumed to be Gaussian iid.

ρ given by perturbed Shepp-Logan phantoms ($N_x = 16384$, $N_Z = 100$)



True **Generated**

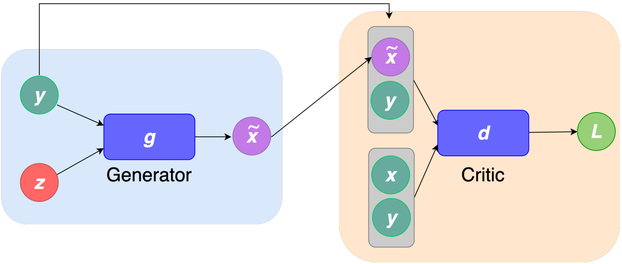
Solving the inference problem on test data



Outline

Conditional WGANs

Learning distributions conditioned on another field. Based on work by Adler et al. (2018) & Almahairi et al. (2018).



Generator network:

- $\boldsymbol{g} : \Omega_Z \times \Omega_Y \to \Omega_X$.
- $\boldsymbol{z} \sim P_Z$, $N_z \ll N_x$.
- $(\boldsymbol{x}, \boldsymbol{y}) \sim P_{XY}$

Critic network:

- $d : \Omega_X \times \Omega_Y \to \mathbb{R}$.
- $d(\boldsymbol{x}, \boldsymbol{y})$ large for real $\boldsymbol{x}$, small otherwise.

Conditional WGANs

- Objective function

$$L(\boldsymbol{g}, d) = \underset{\substack{(\boldsymbol{x},\boldsymbol{y})\sim P_{XY} \\ \boldsymbol{z}\sim P_Z}}{\mathbb{E}} \left[d(\boldsymbol{x}, \boldsymbol{y}) - d\big(\boldsymbol{g}(\boldsymbol{z}, \boldsymbol{y}), \boldsymbol{y}\big)\right]$$

- $\boldsymbol{g}$ and d determined (with constraint $\|d\|_{\text{Lip}} \leq 1$) through

$$(\boldsymbol{g}^*, d^*) = \arg\max_{d} \ \arg\min_{\boldsymbol{g}} L(\boldsymbol{g}, d)$$

- For the optimal generator $\boldsymbol{g}^*$ and given $\boldsymbol{y}$

$$\boldsymbol{g}^*(., \boldsymbol{y}) = \arg\min_{\boldsymbol{g}} W_1(P_{X|Y}, \boldsymbol{g}_{\#}(., \boldsymbol{y})P_Z)$$

- Convergence in W_1 implies weak convergence

$$\underset{\boldsymbol{x}\sim P_{X|Y}}{\mathbb{E}} [\ell(\boldsymbol{x})] = \underset{\boldsymbol{z}\sim P_Z}{\mathbb{E}} [\ell(\boldsymbol{g}(\boldsymbol{z}, \boldsymbol{y}))], \quad \forall\, \ell \in C_b(\Omega_X).$$

Given:

- A set $\mathcal{S} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, where $\boldsymbol{x}_i \sim P_X^{\text{prior}}$ and $\boldsymbol{y}_i \sim P_{Y|X}$.
- A noisy measurement $\boldsymbol{y}$

Goal: Determine P_X^{post} and evaluate statistics wrt it.

Step 1: Using $\mathcal{S}$, train a WGAN with generator $\boldsymbol{g}^*(\boldsymbol{z}, \boldsymbol{y})$.

Using Bayes and weak convergence of conditional WGAN for a given $\boldsymbol{y}$

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_X^{\text{post}}} [\ell(\boldsymbol{x})] = \mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_Z} [\ell(\boldsymbol{g}^*(\boldsymbol{z}, \boldsymbol{y}))], \quad \forall\, \ell \in C_b(\Omega_X)$$

Sampling $\boldsymbol{x}$ from $P_X^{\text{post}} \equiv$ sampling $\boldsymbol{z}$ from P_Z and evaluating $\boldsymbol{x} = \boldsymbol{g}^*(\boldsymbol{z}, \boldsymbol{y})$.

Step 2: Evaluate statistics using Monte Carlo

$$\mathbb{E}_{\boldsymbol{x}\sim P_X^{\text{post}}}[\ell(\boldsymbol{x})] \approx \frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} \ell(\boldsymbol{g}^*(\boldsymbol{z}, \boldsymbol{y}))), \quad \boldsymbol{z} \sim P_Z.$$

What do we gain?

- Ability to represent complex prior, if $\mathcal{S}$ is available.
- $N_z \ll N_x$.
- Sampling from a GAN is very simple.

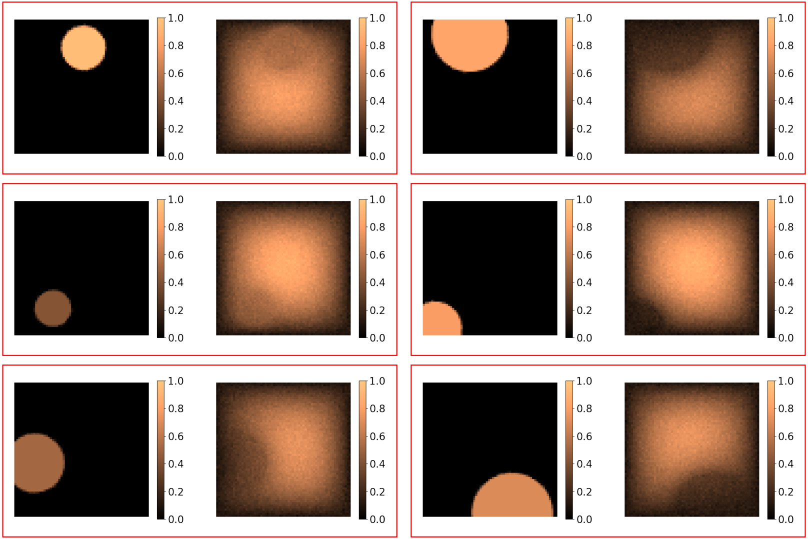
Given u, find κ satisfying

$$-\nabla \cdot (\kappa \nabla u) = 10, \qquad \forall\, \boldsymbol{\xi} \in \Omega \subset \mathbb{R}^2$$
$$u(\boldsymbol{\xi}) = 0, \qquad \forall\, \boldsymbol{\xi} \in \partial\Omega$$

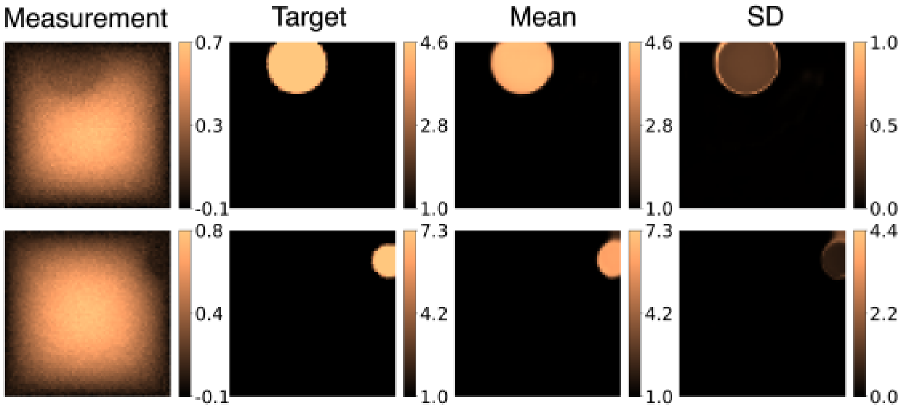
Problem setup:

- Infer $\boldsymbol{x}$, nodal values of conductivity κ.
- Measurement $\boldsymbol{y}$, noisy temperature field u on a 2D grid.
- Generate $\mathcal{S}$ by sampling $\boldsymbol{x} \sim P_X^{\text{prior}}$ and evaluating $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \eta$.
- Train WGAN on $\mathcal{S}$

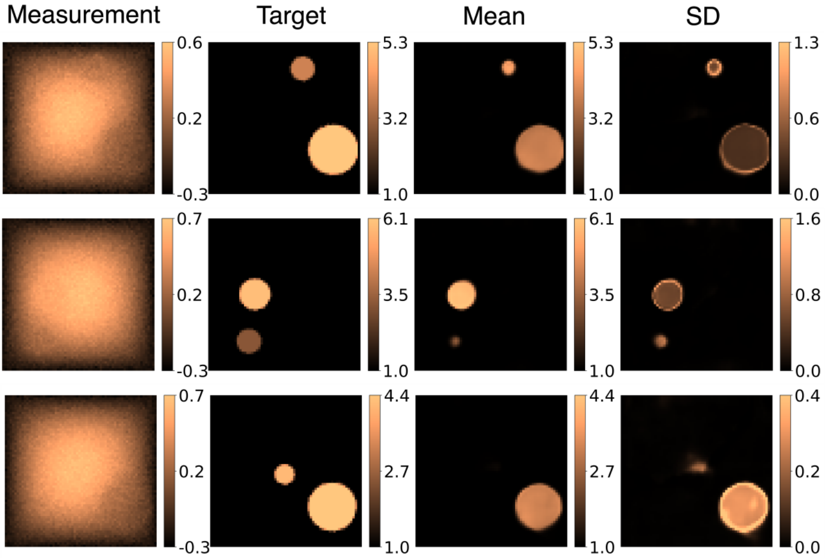
Assume κ is given by circular inclusions ($N_x = N_y = 4096$, $N_Z = 50$)



Solving the inference problem



Inferring thermal conductivity



Given $u(\xi, T)$, find u_0

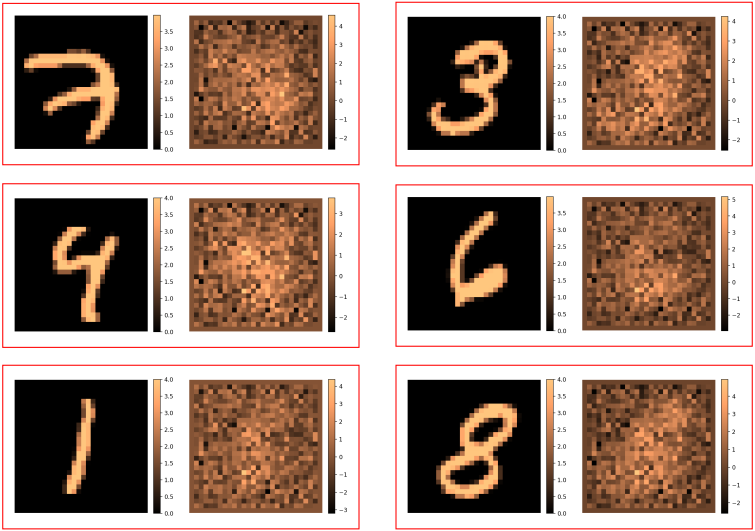
$$\begin{aligned} \frac{\partial u}{\partial t} - \nabla \cdot (2\nabla u) = 0, \qquad & \forall\, (\boldsymbol{\xi}, t) \in \Omega \times (0,1) \\ u(\boldsymbol{\xi}, 0) = u_0(\boldsymbol{\xi}), \qquad & \forall\, \boldsymbol{\xi} \in \Omega \\ u(\boldsymbol{\xi}, t) = 0, \qquad & \forall\, (\boldsymbol{\xi}, t) \in \partial\Omega \times (0,1) \end{aligned}$$

Severely ill-posed problem!

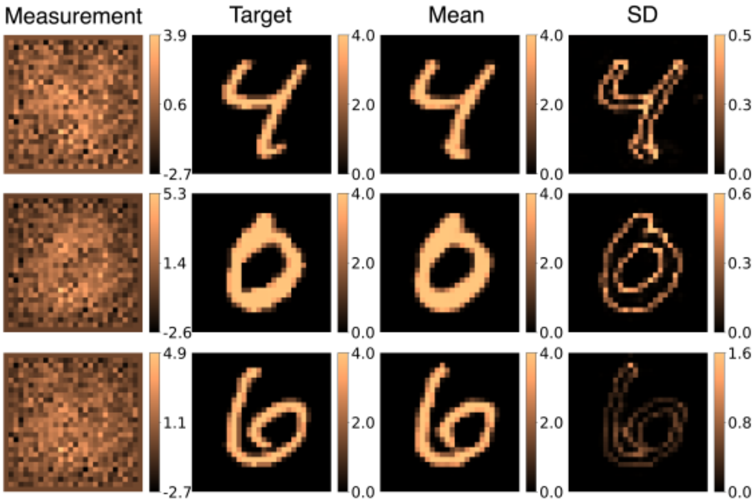
Problem setup:

- Infer $\boldsymbol{x}$, initial temperature field u on a 2D grid.
- Measurement $\boldsymbol{y}$, noisy temperature field u on a 2D grid.
- Generate $\mathcal{S}$ by sampling $\boldsymbol{x} \sim P_X^{\text{prior}}$ and evaluating $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \eta$.
- Train WGAN on $\mathcal{S}$

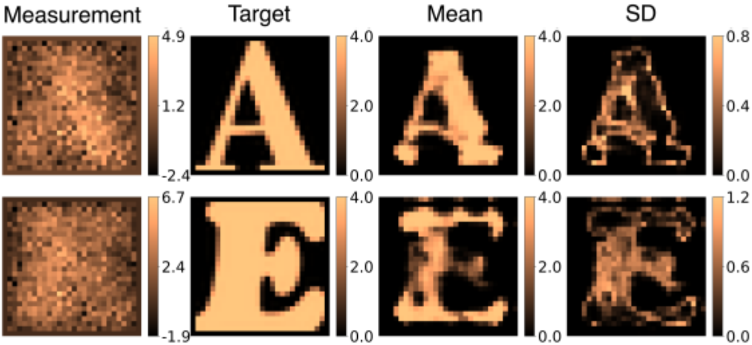
Assume u_0 is given by MNIST ($N_x = N_y = 784, N_Z = 100$)



Solving the inference problem



Solving the inference problem



Comparing the two approaches

	GAN as prior	GAN as posterior
Data generation	$\boldsymbol{x} \sim P_X^{\text{prior}}$	$\boldsymbol{x} \sim P_X^{\text{prior}}$, $\boldsymbol{y} \sim P_{Y\|X}$
Forward model	Need $\boldsymbol{f}$ and $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$	Possibly need $\boldsymbol{f}$ to generate data
Sampling	GAN and MCMC	Only GAN
Generalizability	Hard to control	Better control

Final remarks

- Neural networks are good universal approximators.
- GANs can be used to learn distributions from data and generate new samples.
- Using GANs to overcome challenges with Bayesian inference:
  - GANs as priors.
  - GANS as posterior.
- Ability to capture complex prior information.
- Dimensional reduction using latent space.
- Generate point estimates to quantify uncertainty in inferred field.
- There are many, many other variants of GANs.
- GANs are not the only generative algorithms – Variational Autoencoders (VAEs), normalizing flows, Deep Boltzman Machines (DBMs),etc.

- I. Goodfellow, Y. Bengio, A. Courville.
  Deep Learning, MIT Press.
  http://www.deeplearningbook.org (2016).
- N. Thuerey, P. Holl, M. Mueller, P. Schnell, F. Trost, K. Um.
  Physics-based Deep Learning.
  https://physicsbaseddeeplearning.org (2021).
- A. Pinkus.
  Approximation theory of the MLP model in neural networks.
  Acta Numerica, Vol. 8, 143–195, 1999.
- P. Kidger,T. Lyons
  Universal Approximation with Deep Narrow Networks.
  PMLR,125, 2306–2327, 2020.
- D. Yarotsky, A. Zhevnerchuk
  The phase diagram of approximation rates for deep neural networks.
  arXiv:1906.09477, 2021.

References

J. B. Keller.
Inverse Problems.
The American Mathematical Monthly, 83:107–118, 1976.

I. J. Goodfellow, J. P. -Abadie, M. Mirza, B. Xu, D. W. -Farley, S. Ozair, A. Courville, Y. Bengio
Generative Adversarial Networks.
Advances in Neural Information Processing Systems, 2672–2680, 2014.

M. Arjovsky, S. Chintala, L. Bottou
Wasserstein Generative Adversarial Networks.
PMLR, 70:214-223, 2017

T. Karras, T. Aila, S. Laine, J. Lehtinen
Progressive Growing of GANs for Improved Quality, Stability, and Variation.
arXiv:1710.10196, 2018

J. Adler, O. Öktem
Deep Bayesian Inversion.
arXiv:1811.05910, 2018

References

A. Almahairi, S. Rajeswar, A. Sordoni, P. Bachman, A. Courville
Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data.
PMLR, 80:195-204, 2018

D. Patel, A. A. Oberai
GAN-based Priors for Uncertainty Quantification
arXiv:2003.12597, 2020

D. Patel, D. Ray, A. A. Oberai
Solution of Physics-based Bayesian Inverse Problems with Deep Generative Priors
arXiv:2107.02926, 2021