Real hero behind many relevant works:
Wenbo Gong (MSR)

Make Stein's method "great again" for generative modelling?

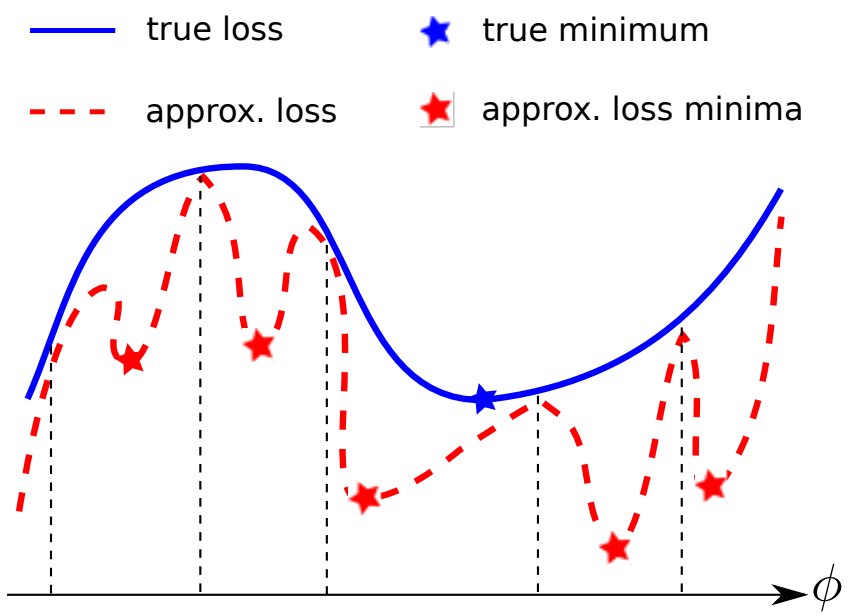
Yingzhen Li

yingzhen.li@imperial.ac.uk

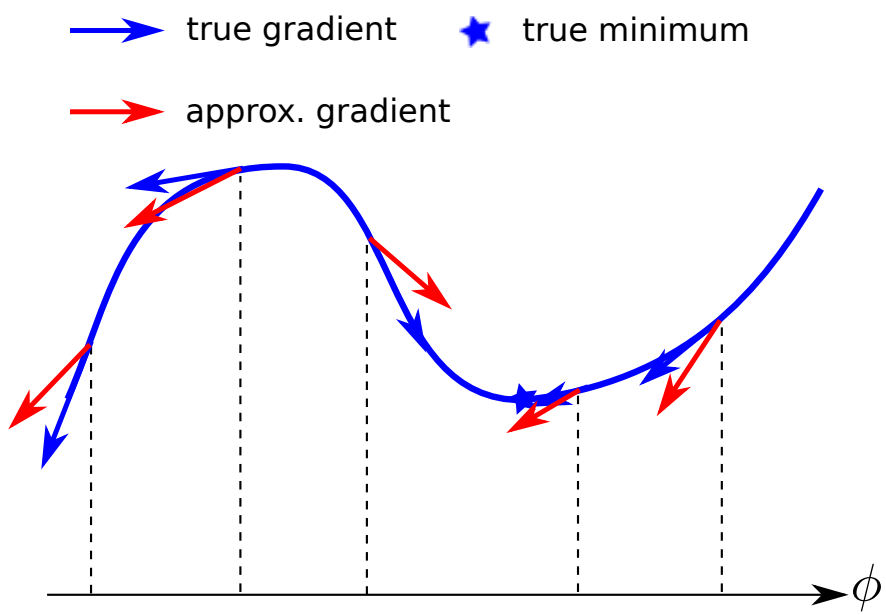
Training generative models with "gradient approximation"

$$\min_{\phi} L(\phi) \text{ with intractable } L(\phi)$$

Loss approximation:
Find $\tilde{L} \approx L$ then $\min_{\phi} \tilde{L}(\phi)$



Gradient approximation:
Find $\tilde{g}(\phi) \approx \nabla_{\phi} L$ then update $\phi \leftarrow \phi - \gamma \tilde{g}(\phi)$



Li and Turner. Gradient Estimators for Implicit Models. ICLR 2018

Example: applications in variational inference

Variational lower-bound: assume $\boldsymbol{z} \sim q_\phi \Leftrightarrow \epsilon \sim \pi(\epsilon), \boldsymbol{z} = \boldsymbol{f}_\phi(\epsilon, \boldsymbol{x})$

$$\begin{aligned}\mathcal{L}_{\mathrm{VI}}(q_\phi) &= \mathbb{E}_q\left[\log p(\boldsymbol{x}, \boldsymbol{z})\right] + \mathbb{H}[q_\phi(\boldsymbol{z}|\boldsymbol{x})] \\ &= \mathbb{E}_\pi\left[\log p(\boldsymbol{x}, \boldsymbol{f}_\phi(\epsilon, \boldsymbol{x}))\right] + \mathbb{H}[q_\phi(\boldsymbol{z}|\boldsymbol{x})] \quad // \textit{ reparam. trick}\end{aligned}$$

If you use gradient descent for optimisation, then you only need gradients!

The gradient of the variational lower-bound:

$$\nabla_\phi \mathcal{L}_{\mathrm{VI}}(q_\phi) = \mathbb{E}_\pi\left[\nabla_{\boldsymbol{f}} \log p(\boldsymbol{x}, \boldsymbol{f}_\phi(\epsilon, \boldsymbol{x}))^\mathsf{T} \nabla_\phi \boldsymbol{f}_\phi(\epsilon, \boldsymbol{x})\right] + \nabla_\phi \mathbb{H}[q_\phi(\boldsymbol{z}|\boldsymbol{x})]$$

The gradient of the entropy term:

$$\nabla_\phi \mathbb{H}[q_\phi(\boldsymbol{z}|\boldsymbol{x})] = -\mathbb{E}_\pi\left[\nabla_{\boldsymbol{f}} \log q(\boldsymbol{f}_\phi(\epsilon, \boldsymbol{x})|\boldsymbol{x})^\mathsf{T} \nabla_\phi \boldsymbol{f}_\phi(\epsilon, \boldsymbol{x})]\right] - \cancel{\mathbb{E}_q[\nabla_\phi \log q_\phi(\boldsymbol{z}|\boldsymbol{x})]}$$

this term is 0

It remains to approximate $\nabla_{\boldsymbol{z}} \log q(\boldsymbol{z}|\boldsymbol{x})$!

Li and Turner. Gradient Estimators for Implicit Models. ICLR 2018

Stein gradient estimator

Goal: approximate $\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})$ for a given distribution $q(\boldsymbol{x})$

Main idea: invert Stein's identity:

$$\mathbb{E}_q[\boldsymbol{h}(\boldsymbol{x})\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})^\top + \nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x})] = \mathbf{0}$$

1. Monte Carlo (MC) approximation to Stein's identity:

$$\frac{1}{K}\sum_{k=1}^{K} -\boldsymbol{h}(\boldsymbol{x}^k)\nabla_{\boldsymbol{x}^k} \log q(\boldsymbol{x}^k)^\top + \text{err} = \frac{1}{K}\sum_{k=1}^{K} \nabla_{\boldsymbol{x}^k}\boldsymbol{h}(\boldsymbol{x}^k), \quad \boldsymbol{x}^k \sim q(\boldsymbol{x}^k),$$

2. Rewrite the MC equations in matrix forms: denoting

$$\mathbf{H} = \left(\boldsymbol{h}(\boldsymbol{x}^1), \cdots, \boldsymbol{h}(\boldsymbol{x}^K)\right), \quad \overline{\nabla_{\boldsymbol{x}}\boldsymbol{h}} = \frac{1}{K}\sum_{k=1}^{K} \nabla_{\boldsymbol{x}^k}\boldsymbol{h}(\boldsymbol{x}^k),$$

$$\mathbf{G} := \left(\nabla_{\boldsymbol{x}^1} \log q(\boldsymbol{x}^1), \cdots, \nabla_{\boldsymbol{x}^K} \log q(\boldsymbol{x}^K)\right)^\top,$$

Then $-\frac{1}{K}\mathbf{H}\mathbf{G} + \text{err} = \overline{\nabla_{\boldsymbol{x}}\boldsymbol{h}}$.

Li and Turner. Gradient Estimators for Implicit Models. ICLR 2018

Stein gradient estimator

Goal: approximate $\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})$ for a given distribution $q(\boldsymbol{x})$

Main idea: invert Stein's identity:

$$\mathbb{E}_q[\boldsymbol{h}(\boldsymbol{x})\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x})^\top + \nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x})] = \mathbf{0}$$

Matrix form (MC): $-\frac{1}{K}\mathbf{H}\mathbf{G} + \text{err} = \overline{\nabla_{\boldsymbol{x}}\boldsymbol{h}}$.

3. Now solve a ridge regression problem:

$$\hat{\mathbf{G}}_V^{\text{Stein}} := \arg\min_{\hat{\mathbf{G}}\in\mathbb{R}^{K\times d}} ||\overline{\nabla_{\boldsymbol{x}}\boldsymbol{h}} + \frac{1}{K}\mathbf{H}\hat{\mathbf{G}}||_F^2 + \frac{\eta}{K^2}||\hat{\mathbf{G}}||_F^2,$$

Analytic solution: $\hat{\mathbf{G}}_V^{\text{Stein}} = -(\mathbf{K} + \eta\boldsymbol{I})^{-1}\langle\nabla, \mathbf{K}\rangle$,

with

$\mathbf{K} := \mathbf{H}^\top\mathbf{H}, \quad \mathbf{K}_{ij} = \mathcal{K}(\boldsymbol{x}^i, \boldsymbol{x}^j) := \boldsymbol{h}(\boldsymbol{x}^i)^\top\boldsymbol{h}(\boldsymbol{x}^j)$,

$\langle\nabla, \mathbf{K}\rangle := K\mathbf{H}^\top\overline{\nabla_{\boldsymbol{x}}\boldsymbol{h}}, \quad \langle\nabla, \mathbf{K}\rangle_{ij} = \sum_{k=1}^K \nabla_{x_j^k}\mathcal{K}(\boldsymbol{x}^i, \boldsymbol{x}^k)$.

Li and Turner. Gradient Estimators for Implicit Models. ICLR 2018

Stein gradient estimator

Kernelized Stein Discrepancy:

$$\mathcal{S}^2(q, \hat{q}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim q}\left[\hat{\boldsymbol{g}}(\boldsymbol{x})^\mathsf{T}\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'}\hat{\boldsymbol{g}}(\boldsymbol{x}') + \hat{\boldsymbol{g}}(\boldsymbol{x})^\mathsf{T}\nabla_{\boldsymbol{x}'}\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} + \nabla_{\boldsymbol{x}}\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'}^\mathsf{T}\hat{\boldsymbol{g}}(\boldsymbol{x}') + \mathrm{Tr}(\nabla_{\boldsymbol{x},\boldsymbol{x}'}\mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'})\right],$$

$$\boldsymbol{g}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}}\log q(\boldsymbol{x}), \quad \hat{\boldsymbol{g}}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}}\log \hat{q}(\boldsymbol{x}), \quad \mathcal{K}_{\boldsymbol{x}\boldsymbol{x}'} = \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}').$$

One can show that the V-statistic of KSD is

$$\mathcal{S}_V^2(q, \hat{q}) = \frac{1}{K^2}\mathrm{Tr}(\hat{\mathbf{G}}^\mathsf{T}\mathbf{K}\hat{\mathbf{G}} + 2\hat{\mathbf{G}}^\mathsf{T}\langle\nabla, \mathbf{K}\rangle) + C$$

This means

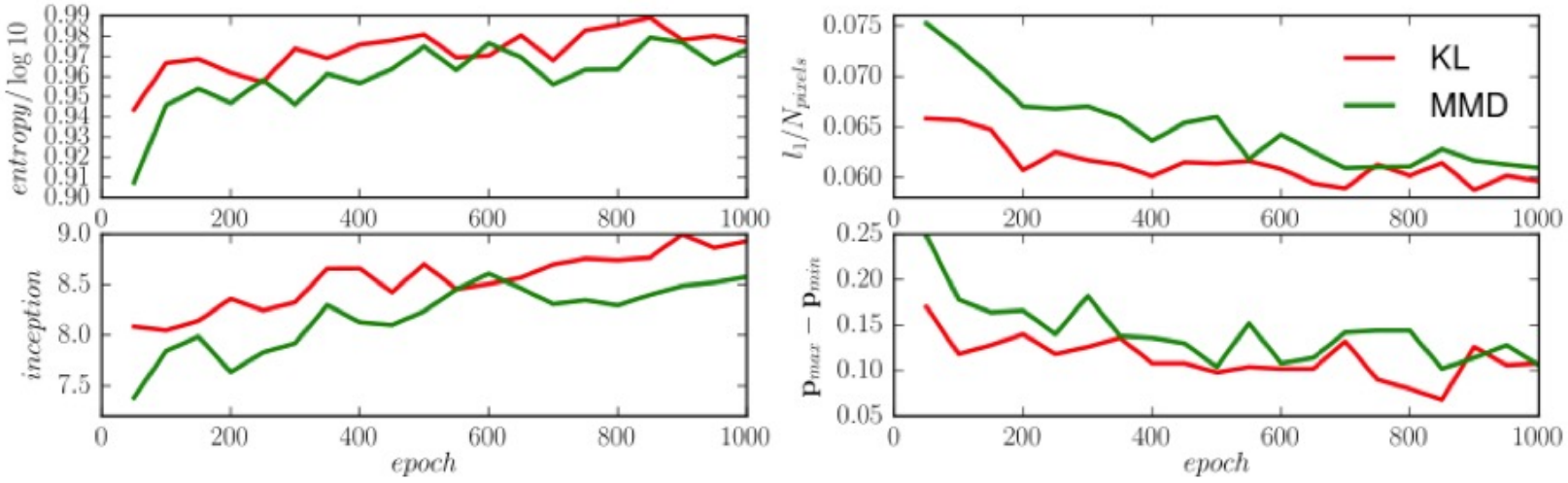
$$\hat{\mathbf{G}}_V^{Stein} = \underset{\hat{\mathbf{G}}\in\mathbb{R}^{K\times d}}{\arg\min} \quad \mathcal{S}_V^2(q, \hat{q}) + \frac{\eta}{K^2}||\hat{\mathbf{G}}||_F^2$$

Li and Turner. Gradient Estimators for Implicit Models. ICLR 2018

Stein gradient estimator

Example: Training an implicit generative model p_θ by minimising $KL[p_\theta \| p_{data}]$:

(defined like GAN's generator)



Li and Turner. Gradient Estimators for Implicit Models. ICLR 2018

Li. Approximate Gradient Descent for Training Implicit Generative Models. NeurIPS 2017 Bayesian Deep Learning Workshop.

The rise of score-based generative models

Yingzhen's idea of Stein gradient estimator (SGE) & summer visit to Tsinghua

Spectral SGE
(Shi et al. ICML 2018)

Learned Stein discrepancy
(Grathwohl et al. ICML 2020)

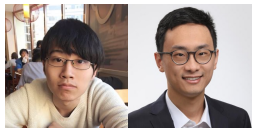
Stein

2017 2018 2019 2020 2021 2022 2023

Fisher & denoising

2019 2020 2021 2022 2023

Sliced score matching
(Song et al. UAI 2019)
Noise conditional score network
(Song and Ermon NeurIPS 2019)

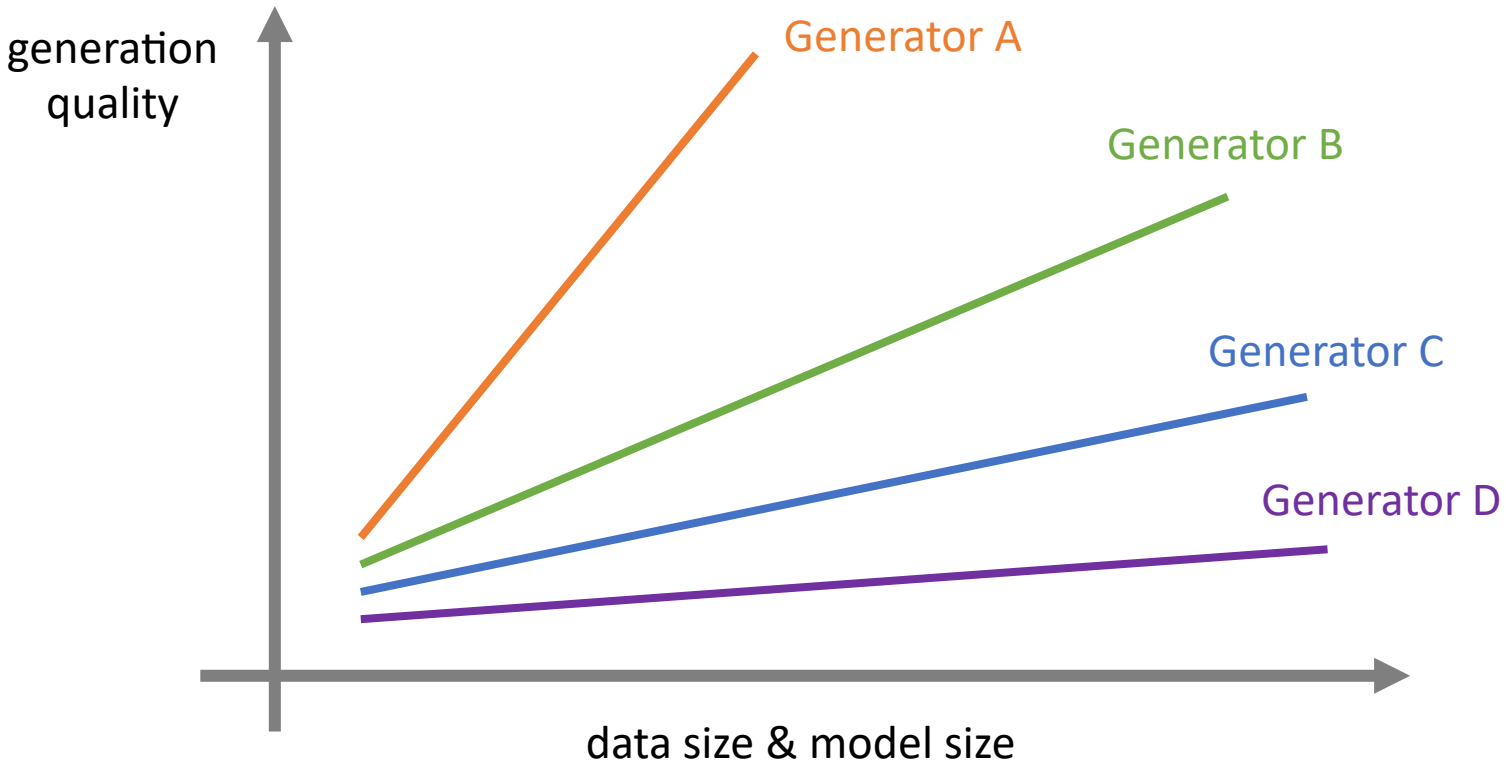


DDPM
(Ho et al. NeurIPS 2020)

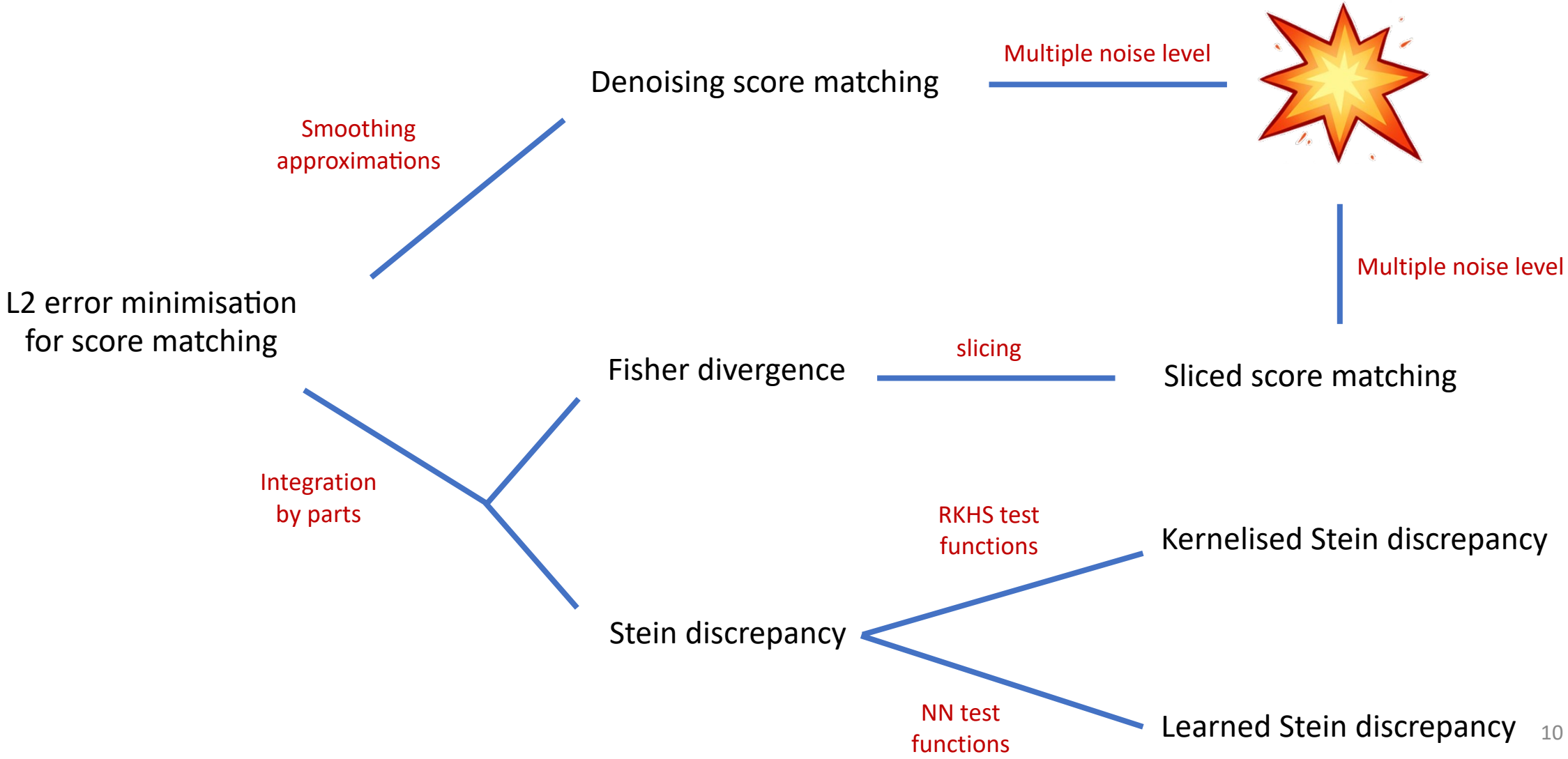
SDE framework
(Song et al. ICLR 2021)

Would Stein's method ever work for generative models?

What do I mean by "work": which one is the generator trained using Stein's method?



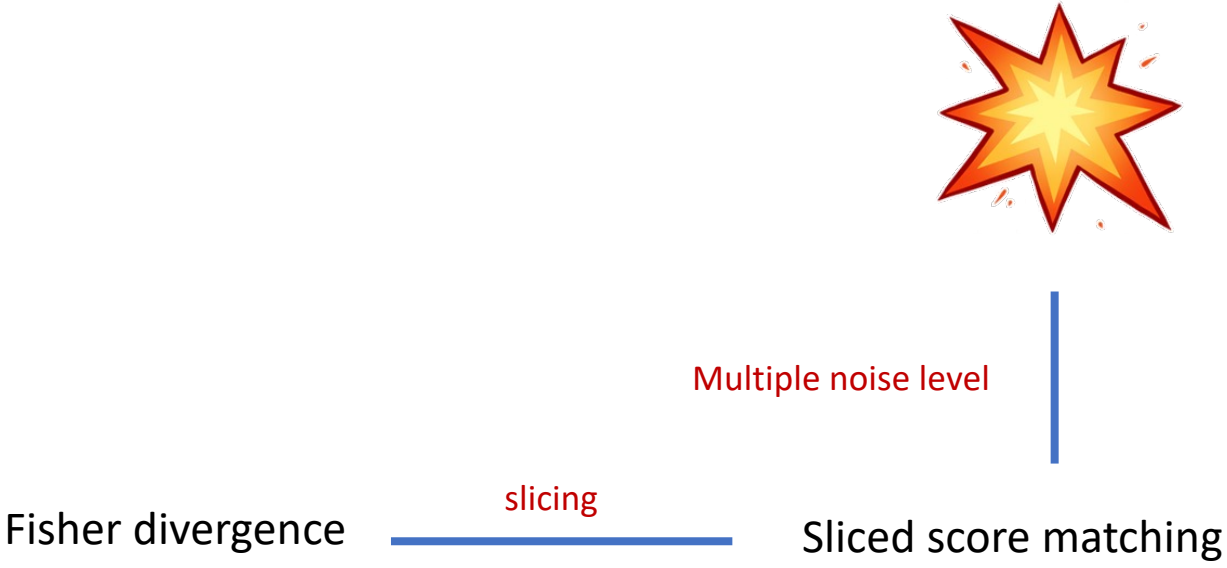
Spot the connections & differences



Solving common issues

- Score function is an $R^d \rightarrow R^d$ mapping
  - Issues in estimating it when d is large, e.g., $d \geq 100$

Solution: slicing



Sliced Stein Discrepancy

- Stein discrepancy:

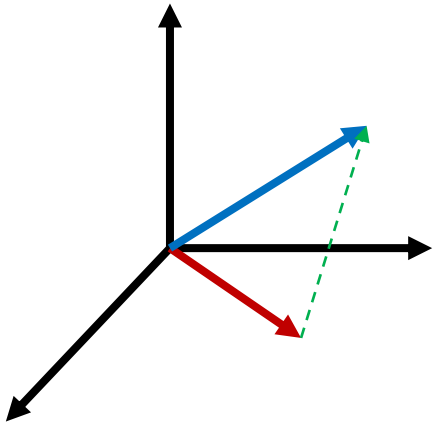
$$S(q,p) = \sup_{f \in F} E_{q(x)}[\nabla_x \log p(x)^\top f(x) + \nabla^\top f(x)],$$

- Curse-of-dimensionality problem:
  - $f \in F$ as L_2 integrable functions:
    - $f^*(x) \propto \nabla_x \log p(x) - \nabla_x \log q(x)$ (intractable)
    - Optimizing $f_\phi(x)$: $R^D \to R^D$ is challenging in high dimensions
  - $f \in F$ as functions in a unit ball of an RKHS:
    - Open question of kernel choice in high dimensions

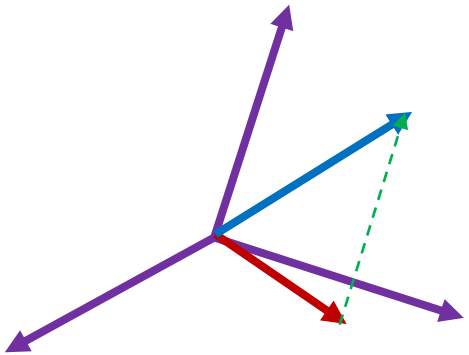
Gong et al. Sliced Kernelized Stein Discrepancy. ICLR 2021
Gong et al. Active Slices for Sliced Stein Discrepancy. ICML 2021

Sliced Stein Discrepancy

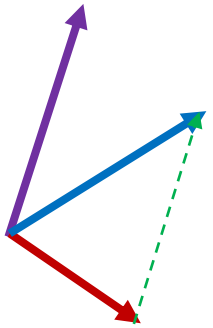
- 1st idea: find the projection direction where scores differ the most (on average)!



$\Rightarrow$



$\Rightarrow$



coordinate system with basis

$$I_d = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ & & \dots & \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

$$\nabla_x \log p(x) - \nabla_x \log q(x)$$

coordinate system with basis

$$O_r^\top = \begin{bmatrix} r_1^\top \\ \dots \\ r_d^\top \end{bmatrix}$$

$$O_r^\top (\nabla_x \log p(x) - \nabla_x \log q(x))$$

One best projection is enough!

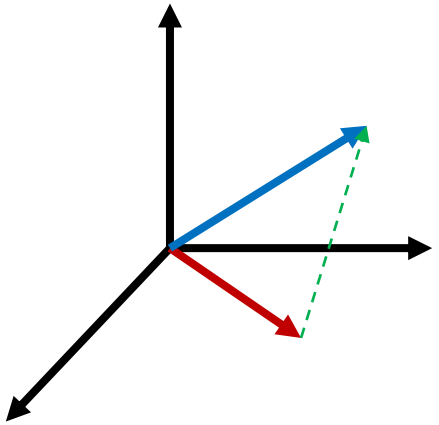
$$r_*$$

$$r_*^\top (\nabla_x \log p(x) - \nabla_x \log q(x))$$

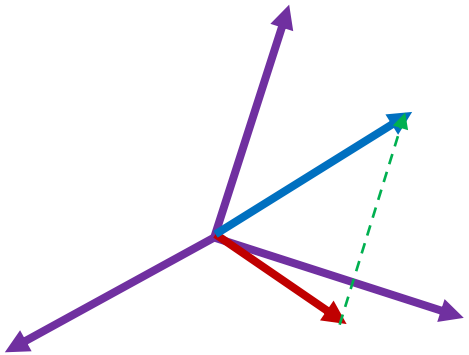
Gong et al. Sliced Kernelized Stein Discrepancy. ICLR 2021
Gong et al. Active Slices for Sliced Stein Discrepancy. ICML 2021

Sliced Stein Discrepancy

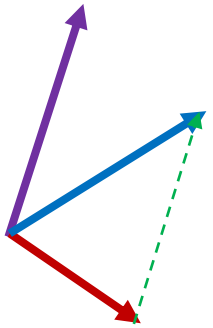
- 1st idea: find the projection direction where scores differ the most (on average)!



$\Rightarrow$



$\Rightarrow$



coordinate system with basis

$$I_d = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ & & \dots & \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

$$\sup_{f \in F} E_{q(x)}[\nabla_x \log p(x)^\top f(x) + \nabla^\top f(x)]$$

$$f: R^d \to R^d$$

coordinate system with basis

$$O_r^\top = \begin{bmatrix} r_1^\top \\ \dots \\ r_d^\top \end{bmatrix}$$

$$\sum_{r \in O_r} \sup_{f_r \in F_r} E_{q(x)}[r^\top \nabla_x \log p(x)\, f_r(x) + r^\top \nabla f_r(x)]$$

$$f_r: R^d \to R, \forall r \in O_r$$

One best projection is enough!

$$r_* = \sup_r S_r$$

$$:= S_r$$

$$\sup_r \sup_{f_r \in F_r} E_{q(x)}[r^\top \nabla_x \log p(x)\, f_r(x) + r^\top \nabla f_r(x)]$$

$$f_r: R^d \to R$$

Gong et al. Sliced Kernelized Stein Discrepancy. ICLR 2021
Gong et al. Active Slices for Sliced Stein Discrepancy. ICML 2021

Sliced Stein Discrepancy

- 2nd idea: apply Radon transform

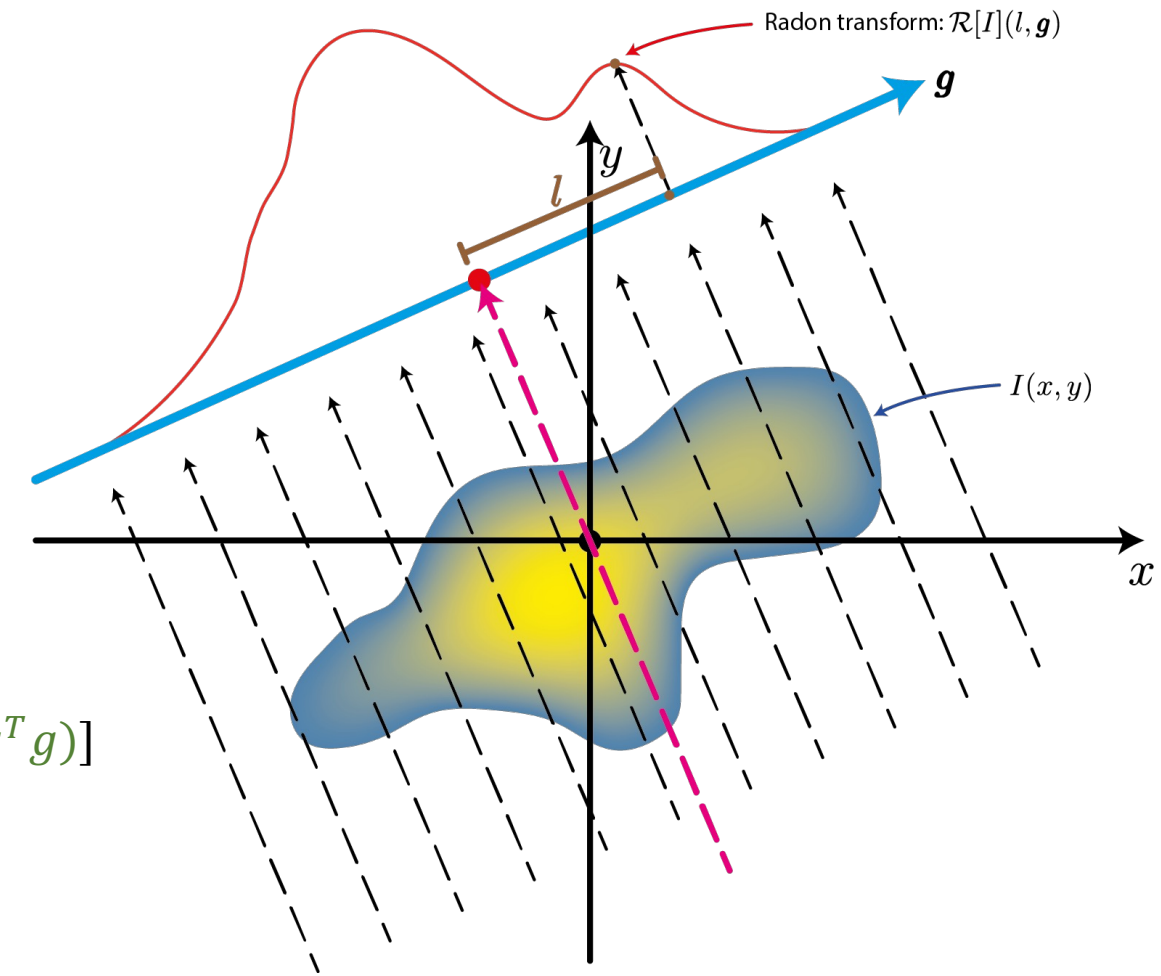
$$f_r: R^d \to R \quad \overset{\text{Radon transform}}{\Leftrightarrow} \quad f_{rg}: R \to R, \forall g \in S^{d-1}$$

- … again, pick best g

Resulting Stein discrepancy: maxSSD-rg

$$\sup_{r,g,f_{rg} \in F_{rg}} E_{q(x)}[r^\top \nabla_x \log p(x)\, f_{rg}(x^T g) + r^T g \nabla_{\boldsymbol{x}^T \boldsymbol{g}} f_{rg}(x^T g)]$$

$$f_{rg}: R \to R$$



Gong et al. Sliced Kernelized Stein Discrepancy. ICLR 2021
Gong et al. Active Slices for Sliced Stein Discrepancy. ICML 2021

ICA Model Learning

ICA generative process:

$$\boldsymbol{z} \sim Lap(0,1),\ \ \boldsymbol{x} = \boldsymbol{W}\boldsymbol{z}, \quad \log p(\boldsymbol{x}) = \log p_z\left(\boldsymbol{W}^{-\mathbf{1}}\boldsymbol{x}\right) + C$$

With normalizing constant C.

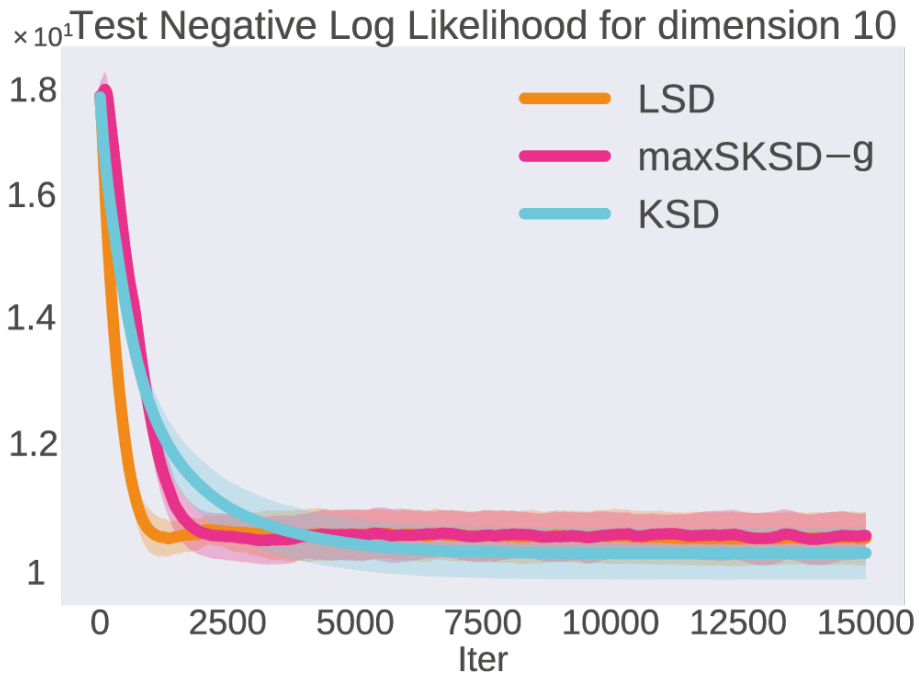
Training/evaluation setup:

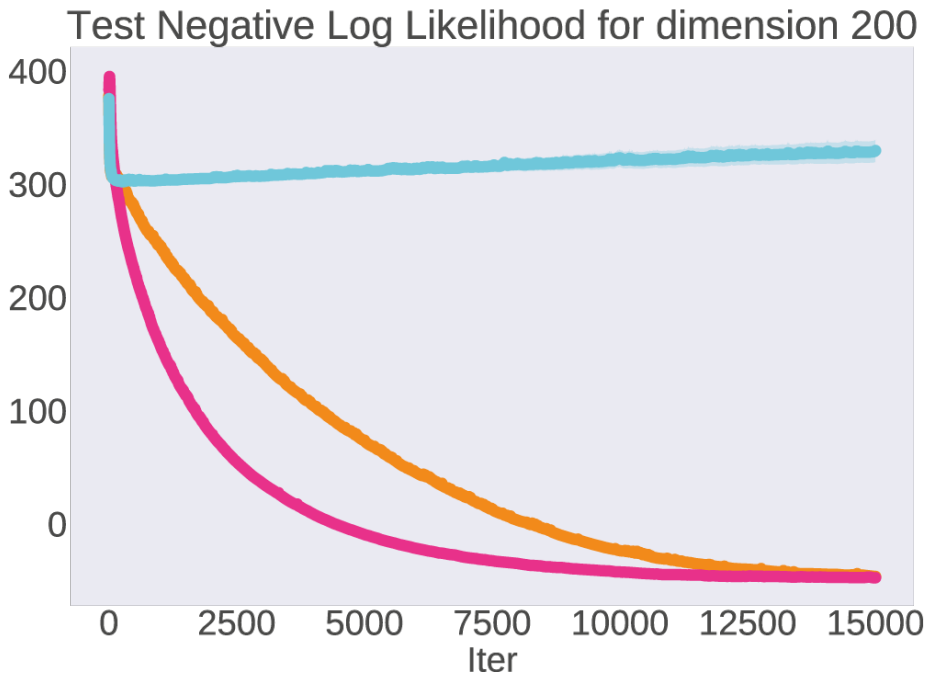
- Generate dataset $\boldsymbol{x}$ by ICA with ground truth $\boldsymbol{W}_t$
- Model $p(\boldsymbol{x})$ is an ICA with random initialized $\boldsymbol{W}$.
- Train $p(\boldsymbol{x})$ to match dataset $\boldsymbol{x}$.
- Evaluate with **log likelihood** on test data**.**

Gong et al. Sliced Kernelized Stein Discrepancy. ICLR 2021
Gong et al. Active Slices for Sliced Stein Discrepancy. ICML 2021

ICA Model Learning

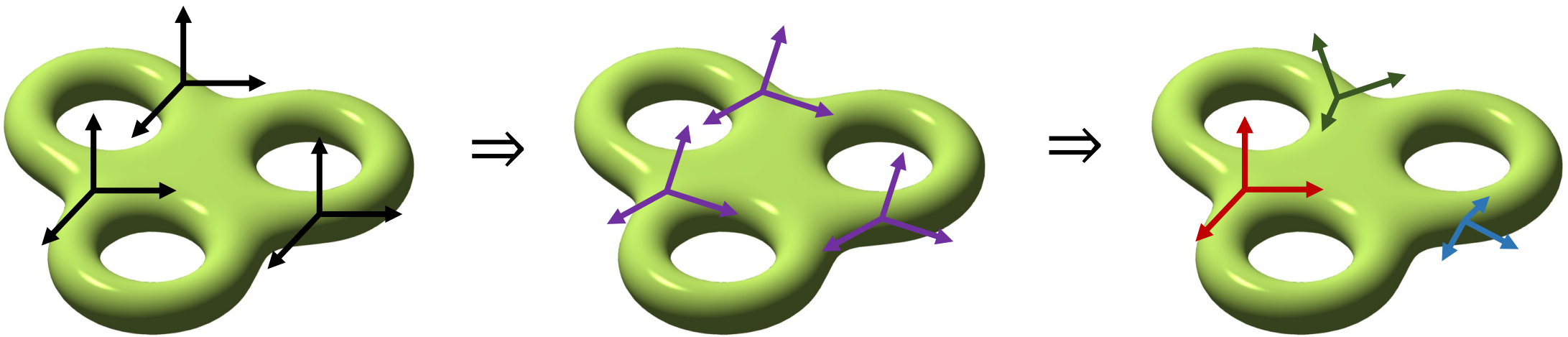
Method	Dimension						
	$D = 10$	$D = 20$	$D = 40$	$D = 60$	$D = 80$	$D = 100$	$D = 200$
KSD	**-10.23**	-15.98	-34.50	-56.87	-86.09	-116.51	-329.49
LSD	-10.42	-14.54	**-17.16**	**-15.05**	-12.39	-5.49	46.63
maxSKSD−g	-10.45	**-14.50**	-17.28	-15.70	**-11.91**	**-4.21**	**47.72**





Gong et al. Sliced Kernelized Stein Discrepancy. ICLR 2021
Gong et al. Active Slices for Sliced Stein Discrepancy. ICML 2021

From Global to Local Coordinates



Stein discrepancy (SD)
Global Coordinate system I_d

Sliced Stein discrepancy (SSD)
Global Coordinate system O_r

Diffusion score matching & SD
Local Coordinate system

Barp et al. Minimum Stein Discrepancy Estimators. NeurIPS 2019
Gong and Li. Interpreting Diffusion Score Matching using Normalizing Flows. ICML 2021 INNF+ workshop

Solving common issues

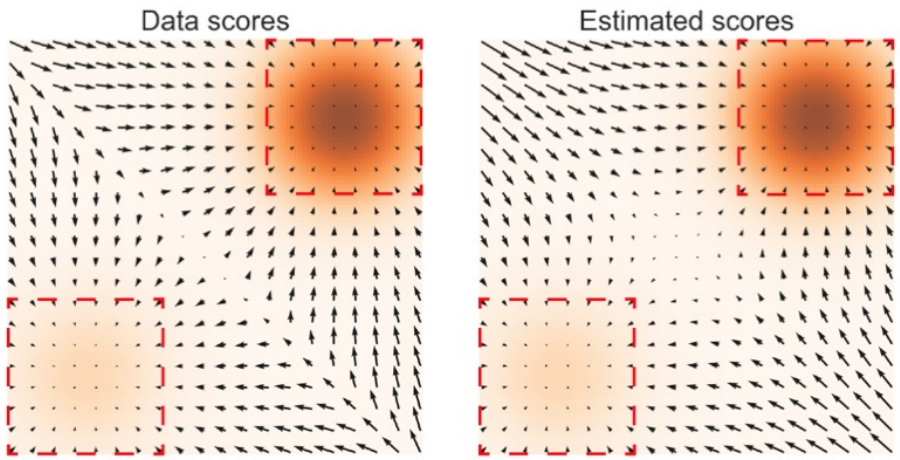
- Disconnected modes

Solution: smoothing approximations

Denoising score matching — Multiple noise level — 

Multiple noise level

Fisher divergence — slicing — Sliced score matching



Example:

$$p(x) = \pi_1 p_1(x) + (1 - \pi_1) p_2(x)$$

$$\nabla \log p(x) = \frac{\pi_1 p_1(x)}{\pi_1 p_1(x) + (1 - \pi_1) p_2(x)} \nabla \log p_1(x) + \frac{\pi_2 p_2(x)}{\pi_1 p_1(x) + (1 - \pi_1) p_2(x)} \nabla \log p_2(x)$$

Song and Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. NeurIPS 2019

Adding Gaussian noise…

Adding noise & Tempering: "temperature-matched" KSD:

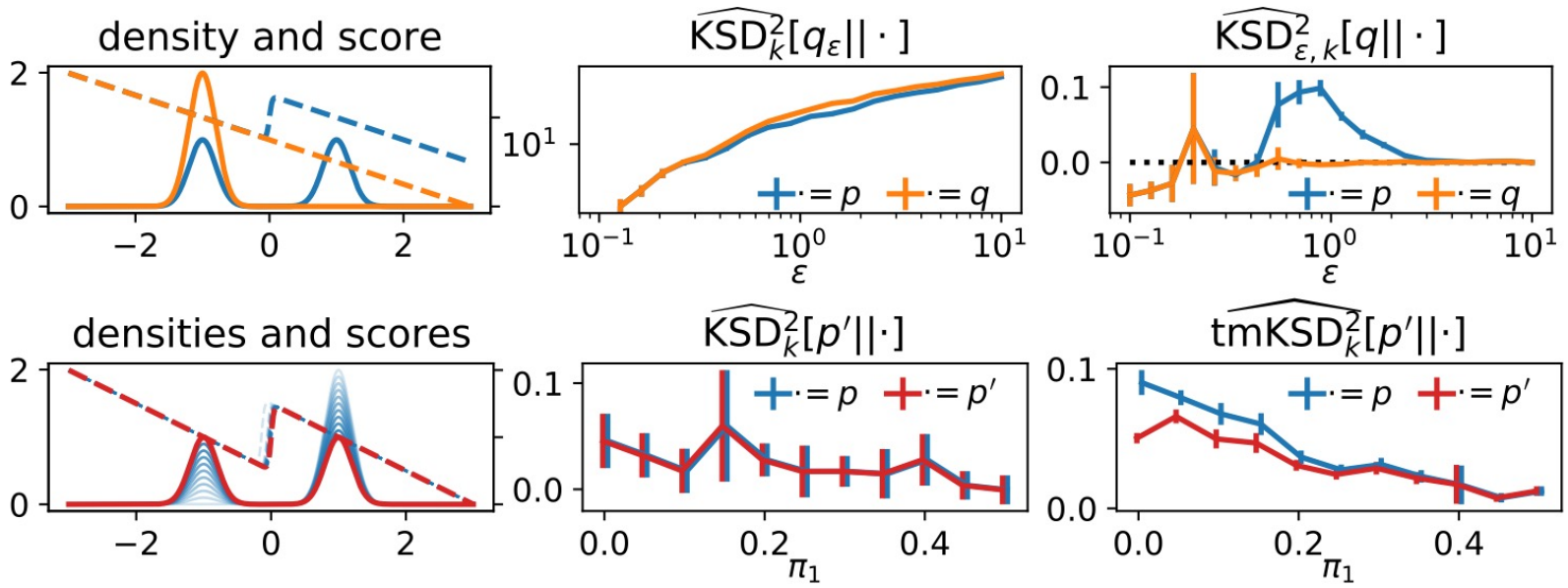
$$\mathrm{tmKSD}_k^2\,[q\|p] := \max_{\epsilon} \mathrm{KSD}_{\mathrm{k},\epsilon}^2\,[q\|p]$$

$$\mathrm{KSD}_{\mathrm{k},\epsilon}^2[q\|p] := \mathrm{KSD}_k^2[q_\epsilon\|\tilde{p}^{\beta^*(\epsilon)}], \quad \text{where} \quad \beta^*(\epsilon) = \arg\min_{\beta\in(0,1]} \mathrm{KSD}_k^2[q_\epsilon\|\tilde{p}^\beta]$$

$$q_\epsilon(x) := \int q(x')\mathcal{N}(x|x',\epsilon^2)\mathrm{d}x' \text{ where } \epsilon > 0, \quad \tilde{p}^\beta(x) \propto \big(p(x)\big)^\beta \text{ where } \beta \in (0,1]$$

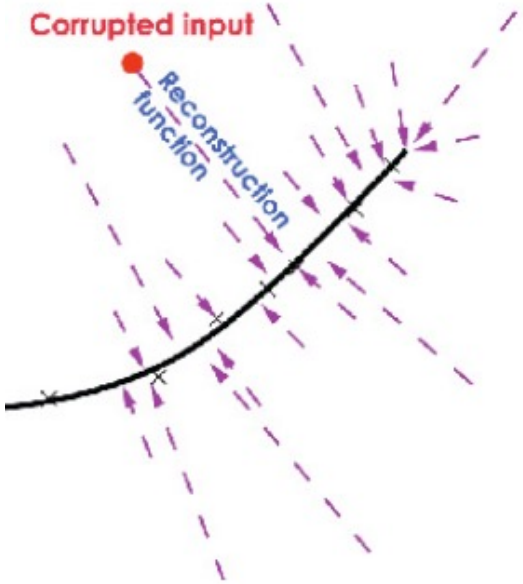
Example:

$$p(x) = \pi_1 p_1(x) + (1 - \pi_1)p_2(x)$$
$$p'(x) = \pi_1' p_1(x) + (1 - \pi_1')p_2(x)$$
$$q(x) = p_1(x)$$



Wenliang and Kanagawa. Blindness of score-based methods to isolated components and mixing proportions. arXiv:2008.10087

Adding Gaussian noise…



Denoising Autoencoder objective:

$$L(r_\sigma) = E_{p(x)p_\sigma(x'|x)}[\|r_\sigma(x') - x\|_2^2]$$

Performing second-order Taylor expansion:

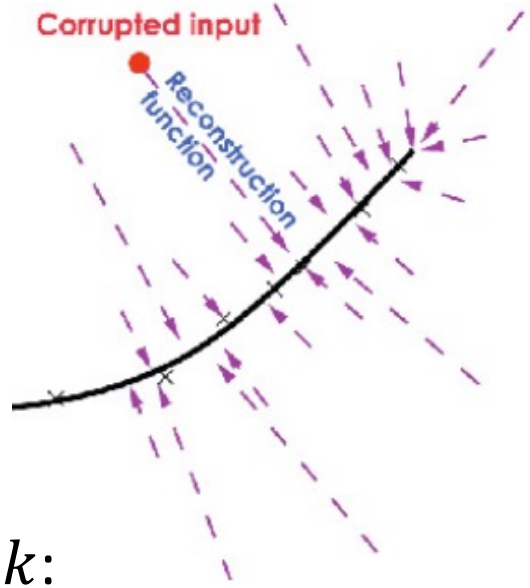
$$r_\sigma(x + \sigma\epsilon) = r_\sigma(x) + \sigma\nabla_x r_\sigma(x)\epsilon + O(\sigma^2\epsilon^2)$$

Rewriting the DAE objective:

$$L(r_\sigma) = E_{p(x)}\left[\|r_\sigma(x) - x\|_2^2 + \sigma^2\left\|\nabla_x r_\sigma(x)\right\|_F^2\right] + O(\sigma^2)$$

Alain and Bengio (2014). What regularized auto-encoders learn from the data-generating distribution. JMLR.
Unpublished result with Wenbo Gong.

Adding Gaussian noise…



Denoising Autoencoder objective when σ is small:

$$L(r_\sigma) = E_{p(x)}\left[\left\|r_\sigma(x) - x\right\|_2^2 + \sigma^2\left\|\nabla_x r_\sigma(x)\right\|_F^2\right]$$

Consider finding the functional gradient in an RKHS H^d with kernel k:

$$F(f) \coloneqq L(r_\sigma + f), \quad \nabla_f F(f) \coloneqq \lim_{\epsilon \to 0} \frac{F(f+\epsilon g) - F(f)}{\epsilon}$$

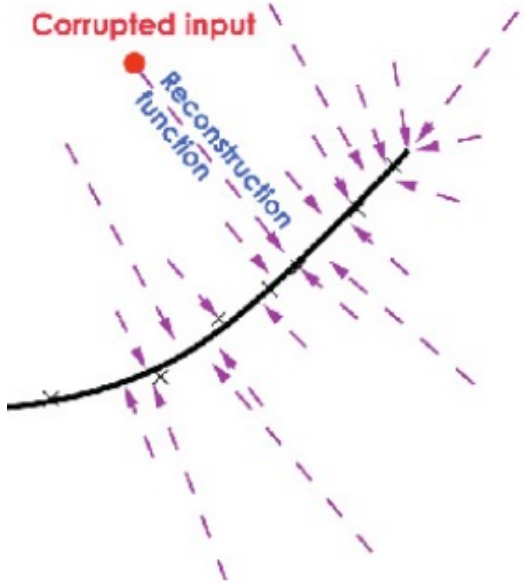
We can show:

$$\nabla_f F(f)|_{f=0} = 2\sigma^2 E_{p(x)}[(r_\sigma(x) - x)k(x,\cdot) + \nabla_x k(x,\cdot)]$$

Define $q(x)$ such that $\nabla_x \log q(x) = \frac{r_\sigma(x) - x}{\sigma^2}$ $\quad \Rightarrow \left\|\nabla_f F(f)|_{f=0}\right\|_{H^d}^2 = 4\sigma^4 KSD(p,q)$

Unpublished result with Wenbo Gong.

Adding Gaussian noise…



Denoising Autoencoder objective when σ is small:

$$L(r_\sigma) = E_{p(x)}\left[\left\|r_\sigma(x) - x\right\|_2^2 + \sigma^2\left\|\nabla_x r_\sigma(x)\right\|_F^2\right]$$

Functional gradient in an RKHS H^d with kernel k:

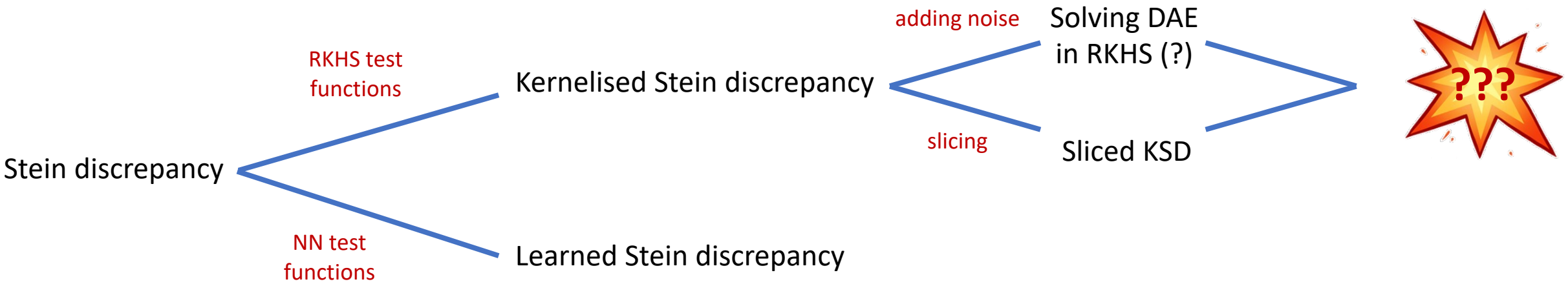
$$\nabla_f F(f)|_{f=0} = 2\sigma^2 E_{p(x)}[(r_\sigma(x) - x)k(x,\cdot) + \nabla_x k(x,\cdot)]$$

With $\nabla_x \log q(x) = \frac{r_\sigma(x)-x}{\sigma^2}$, $\left\|\nabla_f F(f)|_{f=0}\right\|_{H^d}^2 = 4\sigma^4 KSD(p,q)$

- Can try to find DAE updates in an RKHS following the functional gradient
- KSD captures the RKHS norm of the functional gradient
- Finding optimal DAE $r_\sigma(\cdot) \Leftrightarrow KSD(p,q) = 0$

Unpublished result with Wenbo Gong.

Already enough?



Big problem: finding a suitable test function family!

Solving specific issues?

- Kernel methods ever suitable?
- Answer (?): Deep Kernel learning

$$\min_{\theta} \max_{\phi} KSD_{\phi}(p_{\theta}, p_{data})?$$

$$K_{\phi}(x, y) = K_{RBF}(f_{\phi}(x), f_{\phi}(y))$$

MMD-GAN generated CelebA images

Wilson et al. Deep Kernel Learning. AISTATS 2016
Sutherland et al. Generative Models and Model Criticism via Optimized Maximum Mean Discrepancy. ICLR 2017
Li et al. MMD GAN: Towards Deeper Understanding of Moment Matching Network. NeurIPS 2017

Solving specific issues?

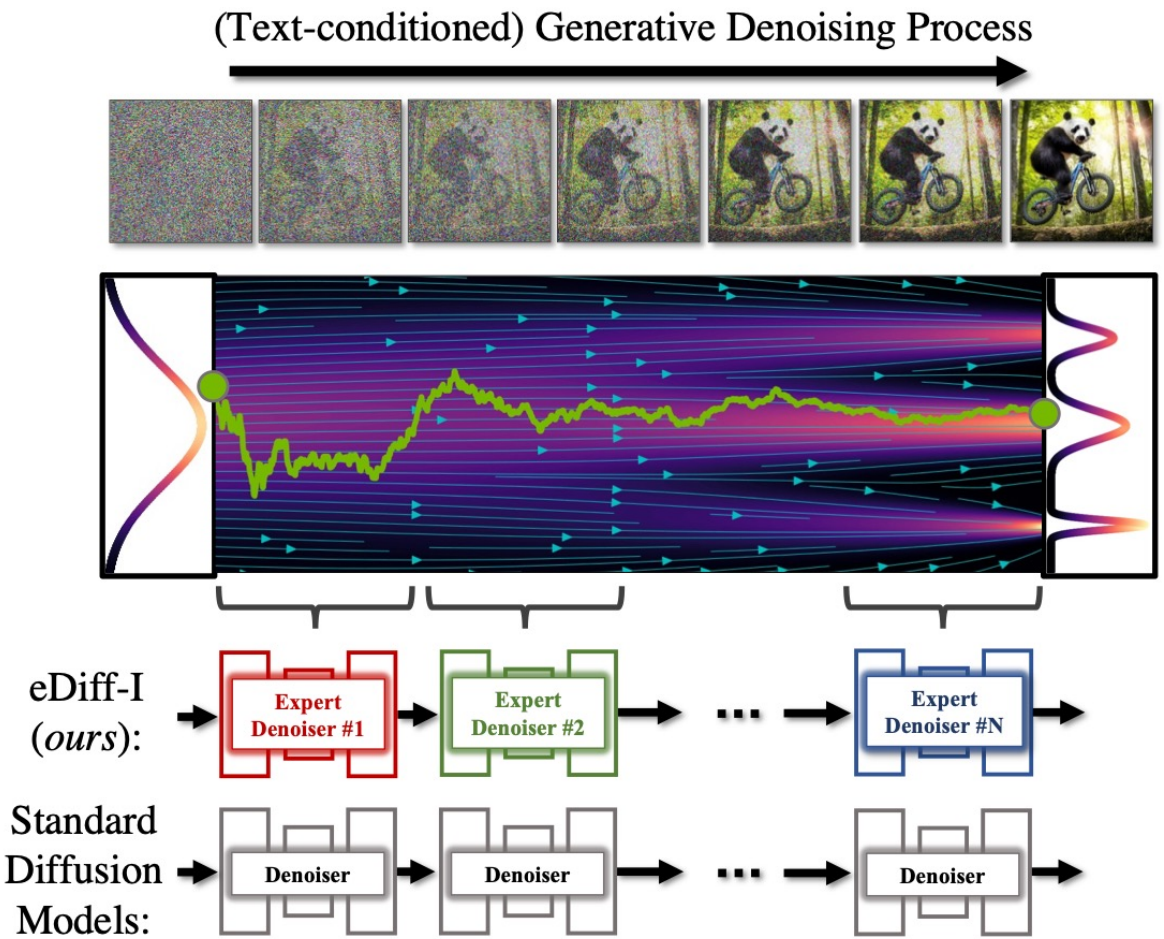
- Adversarial training for multi-dimensional test functions?
- Developments for GANs: lots of stabilisation tricks
  - Label smoothing
  - Regularising the norm of the discriminator function
  - Neural network normalisation layers
  - Designs of learning rate schedules for the generator & discriminator
  - …

Are these tricks directly transferable to Learned Stein discrepancies?

Salimans et al. Improved Techniques for Training GANs. NIPS 2016
Gulrajani et al. Improved Training of Wasserstein GANs. NIPS 2017
Miyato et al. Spectral Normalization for Generative Adversarial Networks. ICLR 2018
Heusel et al. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NIPS 2017

An open question

- Does estimation accuracy of the score matters a lot always?
  - Important for later stages

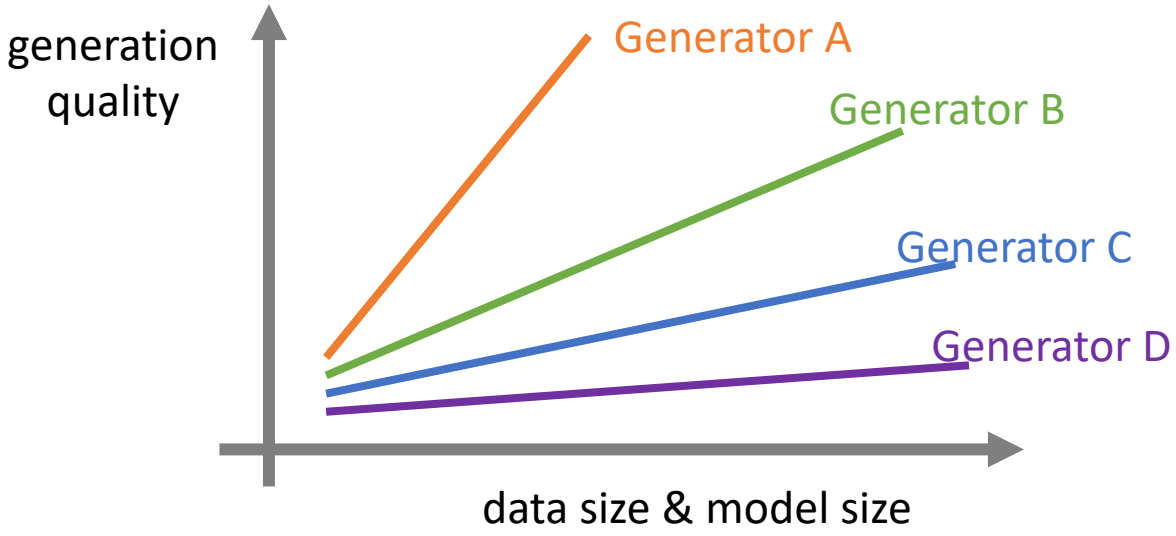


Song et al. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR 2021
Balaji et al. eDiff-I: Text-to-Image Diffusion Models with Ensemble of Expert Denoisers. arXiv:2211.01324

THANK YOU!

Questions? Ask now, or email:
yingzhen.li@imperial.ac.uk

Which one corresponds to Stein's method?



Thanks to my awesome collaborators:

Wenbo Gong

Richard E Turner

Jose Miguel Hernandez-Lobato

...and the future MRes student.

(for believing in me in this topic)